Dubroca_Jean_François_1_notebook_012025

March 4, 2025

[2]:
```
#Importation de la librairie Pandas
import pandas as pd
import matplotlib.pyplot as plt
```

1.2 - Chargement des fichiers Excel

[4]:
```
#Importation du fichier population.csv
population = pd.read_csv('population.csv')

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

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

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

Etape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier population

[7]:
```
#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)
```

[8]:
```
#Consulter le nombre de colonnes
print('Le tableau comporte {} colonnes'.format(population.shape[1]))

#La nature des données dans chacune des colonnes
print (population.dtypes)

#Le nombre de valeurs présentes dans chacune des colonnes
print ('La colonne Zone contient {} valeurs'.format(population['Zone'].
 ↪shape[0]))
```

```
print ('La colonne Année contient {} valeurs'.format(population['Année'].
 ↪shape[0]))
print ('La colonne Valeurs contient {} valeurs'.format(population['Valeur'].
 ↪shape[0]))
```

```
Le tableau comporte 3 colonnes
Zone       object
Année       int64
Valeur    float64
dtype: object
La colonne Zone contient 1416 valeurs
La colonne Année contient 1416 valeurs
La colonne Valeurs contient 1416 valeurs
```

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

[9]:

	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

[10]:
```
#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
population.head(3)
```

[10]:

	Zone	Année	Valeur
0	Afghanistan	2013	32269589.0
1	Afghanistan	2014	33370794.0
2	Afghanistan	2015	34413603.0

[11]:
```
#changement du nom de la colonne Valeur par Population
population.rename (columns = {'Valeur' : 'Population'}, inplace = True)
population.head(3)
```

[11]:

	Zone	Année	Population
0	Afghanistan	2013	32269589.0
1	Afghanistan	2014	33370794.0
2	Afghanistan	2015	34413603.0

[12]:
```
#Affichage les 5 premières lignes de la table pour voir les modifications
population.head()
```

[12]:
```
          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
```

[13]:
```
# Détermination des années couvertes par ce fichier
population['Année'].unique().tolist()
```

[13]:
```
[2013, 2014, 2015, 2016, 2017, 2018]
```

2.2 - Analyse exploratoire du fichier disponibilité alimentaire

[15]:
```
#Afficher les dimensions du dataset
print ('Le dataset dispo_alimentaire contient {} lignes'.
 ↪format(dispo_alimentaire.shape[0]),
       ('et {} colonnes'.format(dispo_alimentaire.shape[1])))
```

```
Le dataset dispo_alimentaire contient 15605 lignes et 18 colonnes
```

[16]:
```
#Consulter le nombre de colonnes
print ('Le dataset dispo_alimentaire contient {} colonnes'.
 ↪format(dispo_alimentaire.shape[1]))
```

```
Le dataset dispo_alimentaire contient 18 colonnes
```

[17]:
```
# Nombre de pays représentés
pays = dispo_alimentaire['Zone'].nunique()
print(f'Il y a {pays} pays représentés')
```

```
Il y a 174 pays représentés
```

[18]:
```
#Affichage les 5 premières lignes de la table
dispo_alimentaire.head()
```

[18]:
```
          Zone                Produit   Origine  Aliments pour animaux  \
0  Afghanistan       Abats Comestible   animale                    NaN
1  Afghanistan        Agrumes, Autres  vegetale                    NaN
2  Afghanistan  Aliments pour enfants  vegetale                    NaN
3  Afghanistan                Ananas  vegetale                    NaN
4  Afghanistan               Bananes  vegetale                    NaN

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

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

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

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

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

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

[19]:
```
#remplacement des NaN dans le dataset par des 0
dispo_alimentaire.fillna(0, inplace = True)
dispo_alimentaire.head(3)
```

[19]:
```
          Zone                Produit   Origine  Aliments pour animaux  \
0  Afghanistan       Abats Comestible   animale                    0.0
1  Afghanistan        Agrumes, Autres  vegetale                    0.0
2  Afghanistan  Aliments pour enfants  vegetale                    0.0

   Autres Utilisations  Disponibilité alimentaire (Kcal/personne/jour)  \
0                  0.0                                             5.0
```

```
1                  0.0                                                1.0
2                  0.0                                                1.0

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

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

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

   Disponibilité intérieure  Exportations - Quantité  Importations - Quantité  \
0                      53.0                       0.0                      0.0
1                      41.0                       2.0                     40.0
2                       2.0                       0.0                      2.0

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

[20]:
```
#multiplication de toutes les lignes contenant des milliers de tonnes en Kg
dispo_alimentaire.iloc[:, 9:] *= 1000000
dispo_alimentaire.head(3)

# après une erreur de manipulation, je n'ai pas su revenir en arrière,
#j'ai essayé plusieurs essais, ce qui explique au final la division pae 1
```

[20]:
```
          Zone                Produit   Origine  Aliments pour animaux  \
0  Afghanistan       Abats Comestible   animale                    0.0
1  Afghanistan        Agrumes, Autres  vegetale                    0.0
2  Afghanistan  Aliments pour enfants  vegetale                    0.0

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

   Disponibilité alimentaire en quantité (kg/personne/an)  \
0                                               1.72
```

```
1                                                1.29
2                                                0.06

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

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

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

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

[21]:
```
#Affichage les 5 premières lignes de la table
dispo_alimentaire.head()
```

[21]:
```
          Zone                Produit   Origine  Aliments pour animaux  \
0  Afghanistan       Abats Comestible   animale                    0.0
1  Afghanistan        Agrumes, Autres  vegetale                    0.0
2  Afghanistan  Aliments pour enfants  vegetale                    0.0
3  Afghanistan                Ananas  vegetale                    0.0
4  Afghanistan               Bananes  vegetale                    0.0

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

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

```
   Disponibilité de matière grasse en quantité (g/personne/jour)  \
0                                                       0.20
1                                                       0.01
2                                                       0.01
3                                                       0.00
4                                                       0.02

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

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

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

2.3 - Analyse exploratoire du fichier aide alimentaire

[23]:
```
#Afficher les dimensions du dataset
print ('Le dataset aide alimentaire contient {} lignes'.format(aide_alimentaire.
 ↪shape[0]),
('et {} colonnes'.format(aide_alimentaire.shape[1])))
```

```
Le dataset aide alimentaire contient 1475 lignes et 4 colonnes
```

[24]:
```
#Consulter le nombre de colonnes
print ('Le dataset aide alimentaire contient {} colonnes'.
 ↪format(aide_alimentaire.shape[1]))
```

```
Le dataset aide alimentaire contient 4 colonnes
```

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

[25]:
```
  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
```

[26]:
```
# Années d'études
aide_alimentaire['Année'].unique()
```

[26]:
```
array([2013, 2014, 2015, 2016])
```

[27]:
```
#changement du nom de la colonne Pays bénéficiaire par Zone
aide_alimentaire.rename(columns = {'Pays bénéficiaire': 'Zone'}, inplace = True)
aide_alimentaire.head(3)
```

[27]:
```
          Zone  Année              Produit  Valeur
0  Afghanistan   2013  Autres non-céréales     682
1  Afghanistan   2014  Autres non-céréales     335
2  Afghanistan   2013         Blé et Farin   39224
```

[28]:
```
#Multiplication de la colonne Aide_alimentaire qui contient des tonnes par 1000␣
 ↪pour avoir des kg
aide_alimentaire['Valeur'] = aide_alimentaire['Valeur'] * 1000
aide_alimentaire.head(3)
```

[28]:
```
          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
```

[29]:
```
#Affichage les 5 premières lignes de la table
aide_alimentaire.head()
```

[29]:
```
          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

[31]:
```
#Afficher les dimensions du dataset
print ('Le dataset sous_nutrition contient {} lignes'.format(sous_nutrition.
 ↪shape[0]),
'et {} colonnes'.format(sous_nutrition.shape[1]))
```

```
Le dataset sous_nutrition contient 1218 lignes et 3 colonnes
```

[32]:
```
#Consulter le nombre de colonnes
print ('Le fichier sous_nutrition contient {} colonnes'.format(sous_nutrition.
 ↪shape[1]))
```

```
Le fichier sous_nutrition contient 3 colonnes
```

[33]:
```
#Afficher les 5 premières lignes de la table
sous_nutrition.head()
```

[33]:

	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

[34]:
```
# Année d'étude
sous_nutrition['Année'].unique()
```

[34]:
```
array(['2012-2014', '2013-2015', '2014-2016', '2015-2017', '2016-2018',
       '2017-2019'], dtype=object)
```

[35]:
```
#Conversion de la colonne sous nutrition en numérique
#sous_nutrition['Valeur'] = pd.to_numeric(sous_nutrition['Valeur'])
# Exemple donné qui ne fonctionne pas.
```

[36]:
```
#Conversion de la colonne (avec l'argument errors=coerce qui permet de␣
 ↪convertir automatiquement
#les lignes qui ne sont pas des nombres en NaN)
#Puis remplacement des NaN en 0
sous_nutrition['Valeur'] = pd.to_numeric(sous_nutrition['Valeur'], errors =␣
 ↪'coerce')
sous_nutrition['Valeur'] = sous_nutrition['Valeur'].fillna(0)
sous_nutrition.dtypes
```

[36]:
```
Zone       object
Année      object
Valeur    float64
dtype: object
```

[37]:
```
#changement du nom de la colonne Valeur par sous_nutrition
sous_nutrition.rename(columns = {'Valeur' : 'Sous_nutrition'}, inplace = True)
sous_nutrition.head(3)
```

[37]:

	Zone	Année	Sous_nutrition
0	Afghanistan	2012-2014	8.6
1	Afghanistan	2013-2015	8.8
2	Afghanistan	2014-2016	8.9

[38]:
```
#Multiplication de la colonne sous_nutrition par 1000000
sous_nutrition['Sous_nutrition'] = sous_nutrition['Sous_nutrition'] * 1000000
```

[39]:
```
#Afficher les 5 premières lignes de la table
sous_nutrition.head()
```

[39]:

	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

[41]:
```
# Il faut tout d'abord faire une jointure entre la table population et la table sous nutrition, en ciblant l'année 2017
population_sous_nutrition_2017 = (pd.merge(sous_nutrition.loc[sous_nutrition['Année'] == '2016-2018', ['Zone', 'Sous_nutrition']],
                                           population.loc[population['Année'] == 2017, :], on = 'Zone', how = 'left')
                                  )

population_sous_nutrition_2017.head(3)
```

[41]:

	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

[42]:
```
#Affichage du dataset
population_sous_nutrition_2017
```

[42]:

	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 x 4 columns]

[43]:
```
#Calcul et affichage du nombre de personnes en état de sous nutrition
total_sous_nutrition = population_sous_nutrition_2017['Sous_nutrition'].sum()
print ('Le nombre total de personnes en état de sous nutrition dans le monde␣
 ↪pour l\'année 2017 est de')
print(total_sous_nutrition / 1000000,'millions de personnes')
```

```
Le nombre total de personnes en état de sous nutrition dans le monde pour
l'année 2017 est de
535.7 millions de personnes
```

[306]:
```
print(f' Population mondiale en 2017:␣
 ↪{round(population_sous_nutrition_2017['Population'].sum()/1000000000, 2)}␣
 ↪milliard d\'habitants')
```

```
 Population mondiale en 2017: 7.54 milliard d'habitants
```

3.2 - Nombre théorique de personne qui pourrait être nourries

[46]:
```
#Combien mange en moyenne un être humain ? Source => https://www.vidal.fr/sante/
 ↪nutrition/equilibre-alimentaire-adulte/
 ↪recommandations-nutritionnelles-adulte.html
print ('Pour un homme adulte, l'apport conseillé en énergie est, en moyenne, de␣
 ↪2 400 à 2 600 calories par jour, selon l'activité. Pour une femme adulte, il␣
 ↪est de 1 800 à 2 200 calories. Nous retiendrons ici 2200 calories.')
```

```
Pour un homme adulte, l'apport conseillé en énergie est, en moyenne, de 2 400 à
2 600 calories par jour, selon l'activité. Pour une femme adulte, il est de 1
800 à 2 200 calories. Nous retiendrons ici 2200 calories.
```

[47]:
```
#On commence par faire une jointure entre le data frame population et␣
 ↪Dispo_alimentaire afin d'ajouter dans ce dernier la population
# Le fichier dispo alimentaire n'est valable que pour 2017
population_2017 = population.loc[population['Année'] == 2017, :]
dispo_alimentaire_population = pd.merge(dispo_alimentaire, population_2017,␣
 ↪on='Zone', how='left')
```

[48]:
```
#Affichage du nouveau dataframe
dispo_alimentaire_population.head(3)
```

[48]:

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

```
   Disponibilité alimentaire en quantité (kg/personne/an)  \
0                                                1.72
1                                                1.29
2                                                0.06

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

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

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

   Nourriture     Pertes  Production  Semences  Traitement  \
0  53000000.0        0.0  53000000.0       0.0         0.0
1  39000000.0  2000000.0   3000000.0       0.0         0.0
2   2000000.0        0.0         0.0       0.0         0.0

   Variation de stock   Année  Population
0                 0.0  2017.0  36296113.0
1                 0.0  2017.0  36296113.0
2                 0.0  2017.0  36296113.0
```

[49]:
```
#Création de la colonne dispo_kcal avec calcul des kcal disponibles mondialement
# Pour calculer la colonne dispo_kcal, j'ai multiplié la colonne 'disponibilité␣
 ↪alimentaire (kcal/personne/jour'
# par la population

dispo_alimentaire_population['dispo_kcal']=(dispo_alimentaire_population['Disponibilité␣
 ↪alimentaire (Kcal/personne/jour)']*

 ↪dispo_alimentaire_population['Population'] * 365
                                             )



dispo_alimentaire_population.head(3)
```

[49]:
```
          Zone                Produit   Origine  Aliments pour animaux  \
0  Afghanistan       Abats Comestible   animale                    0.0
1  Afghanistan        Agrumes, Autres  vegetale                    0.0
```

```
2  Afghanistan  Aliments pour enfants  vegetale                    0.0

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

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

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

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

   Disponibilité intérieure  …  Importations - Quantité  Nourriture  \
0               53000000.0  …                     0.0  53000000.0
1               41000000.0  …              40000000.0  39000000.0
2                2000000.0  …               2000000.0   2000000.0

      Pertes  Production  Semences  Traitement  Variation de stock   Année  \
0        0.0  53000000.0       0.0         0.0                 0.0  2017.0
1  2000000.0   3000000.0       0.0         0.0                 0.0  2017.0
2        0.0         0.0       0.0         0.0                 0.0  2017.0

   Population    dispo_kcal
0  36296113.0  6.624041e+10
1  36296113.0  1.324808e+10
2  36296113.0  1.324808e+10

[3 rows x 21 columns]
```

[50]:
```
#Calcul du nombre d'humains pouvant être nourris

# Cacul du nombre total de calories stocké dans une variable
kcal_total = dispo_alimentaire_population['dispo_kcal'].sum()

# Affichage de la réponse
print ('Considérant un apport journalier de 2200 calories par jour et par␣
 ↪personne, en 2017 nous disposions de', kcal_total,'calories,')
```

```
print ('nous pouvions donc nourir', round(kcal_total/(2200*365)/1000000000, 2),␣
 ↪'milliards de personnes')
```

```
Considérant un apport journalier de 2200 calories par jour et par personne, en
2017 nous disposions de 7635429388975815.0 calories,
nous pouvions donc nourir 9.51 milliards de personnes
```

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

[52]:
```
#Transfert des données avec les végétaux dans un nouveau dataframe
dispo_vegetaux = dispo_alimentaire_population.
 ↪loc[dispo_alimentaire_population['Origine'] == 'vegetale', :]
dispo_vegetaux.head(3)
```

[52]:

	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	…	Importations - Quantité	Nourriture	Pertes	Production	Semences	Traitement	Variation de stock	Année
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	0.02	41000000.0	…	40000000.0	39000000.0	2000000.0	3000000.0	0.0	0.0	0.0	2017.0
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	0.03	2000000.0	…	2000000.0	2000000.0	0.0	0.0	0.0	0.0	0.0	2017.0
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	0.00	0.0	…	0.0	0.0						

```
3        0.0          0.0        0.0          0.0                    0.0  2017.0

   Population    dispo_kcal
1  36296113.0  1.324808e+10
2  36296113.0  1.324808e+10
3  36296113.0  0.000000e+00

[3 rows x 21 columns]
```

[53]:
```
#Calcul du nombre de kcal disponible pour les végétaux
kcal_vegetaux = dispo_vegetaux['dispo_kcal'].sum()
print ('Les végétaux représentent', kcal_vegetaux, 'calories en 2017')
```

```
Les végétaux représentent 6300178937197865.0 calories en 2017
```

[54]:
```
#Calcul du nombre d'humains pouvant être nourris avec les végétaux
print ('Considérant 2200 calories nécessaires par jour et par personne, nous␣
 ↪pouvions nourrir', round(kcal_vegetaux/(2200 * 365)/1000000000,0))
print ('milliards de personnes avec les végétaux en 2017')
```

```
Considérant 2200 calories nécessaires par jour et par personne, nous pouvions
nourrir 8.0
milliards de personnes avec les végétaux en 2017
```

3.4 - Utilisation de la disponibilité intérieure

[56]:
```
#Calcul de la disponibilité totale

# Création d'une variable pour stocker la somme de 'disponibilté intérieure'
dispo_int_totale = dispo_alimentaire_population['Disponibilité intérieure'].
 ↪sum()

# Affichage de la réponse
print ('En 2017, la disponibilité intérieure était de', dispo_int_totale/
 ↪1000000000,'milliards de tonnes')
```

```
En 2017, la disponibilité intérieure était de 9848.994 milliards de tonnes
```

[57]:
```
# Nous ne connaissons pas l'unité de la colonne Aliments pour animaux, j'ai␣
 ↪supposé qu'elle était exprimée en milliers de tonnes
#comme les autres que nous avons transformé en kg.

# Expression des colonnes 'Aliments pour animaux' et 'Autres utilisations' en␣
 ↪kg (exprimée en milliers de tonnes)
dispo_alimentaire_population['Aliments pour animaux'] *= 1000000
dispo_alimentaire_population['Autres Utilisations'] *= 1000000
```

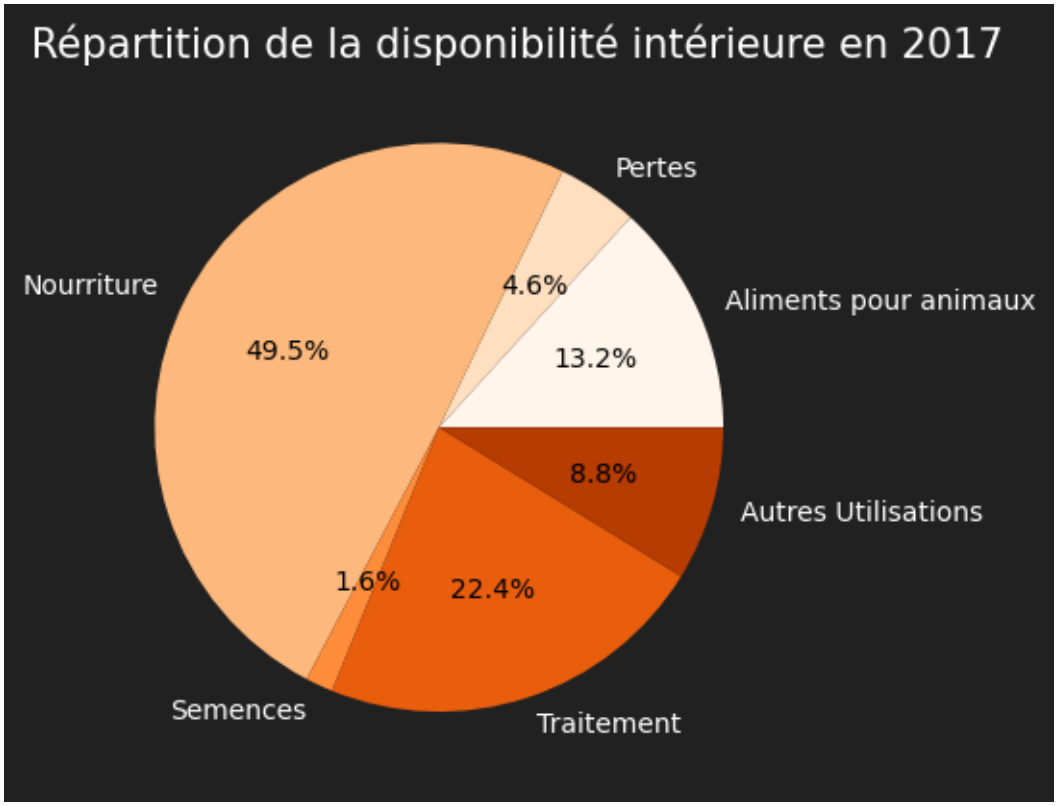
[310]:
```
#création d'une boucle for pour afficher les différentes valeurs en fonction␣
 ↪des colonnes aliments pour animaux, pertes, nourritures,
```

```
#Disponibilité intérieure = Semences + Pertes + Nourriture + Aliments pour␣
 ↪animaux + Traitement + Autres utilisations

# Création d'une variable qui contient le nom des colonnes
colonnes = ['Aliments pour animaux', 'Pertes', 'Nourriture', 'Semences',␣
 ↪'Traitement', 'Autres Utilisations']

# Création d'une variable pour stocker les valeurs en vue de faire un piechart
valeurs = []
label = []

print()
print('-' * 98)
# Création de la boucle for
for colonne in colonnes:
    somme = dispo_alimentaire_population[colonne].sum()
    valeurs.append(somme)
    label.append(colonne)
    pct_dispo_int = round(somme / dispo_int_totale *100, 2)
    if colonne == 'Pertes':
        print ('En 2017, les', colonne, 'représentent', somme/1000000000,␣
 ↪'millions de tonnes (', pct_dispo_int,'%)')
    else:
        print ('En 2017, la disponibilité en', colonne ,'était de', somme/
 ↪1000000000,'millions de tonnes (', pct_dispo_int,'%)')
print('-' *98)
print()
# Création de la figure et de l'axe
fig, ax = plt.subplots()

# Modification de la couleur de fond
fig.patch.set_facecolor('#212121')

# Personnalisation des couleurs du graphe
cmap = plt.get_cmap('Oranges') # Récupère la palette Oranges srockée dans cmap
colors = [cmap(i / len(valeurs)) for i in range(len(valeurs))]
# colors = [cmap(i / 3) for i in range(3)] si l'on ne souhaite générer que 3␣
 ↪couleurs
# cmap est un colormap qui génère une couleur en fonction d'un nombre compris␣
 ↪entre 0 et 1
# i / len(valeurs) convertit l'indice i en une valeur normalisée entre 0 et 1␣
 ↪pour une progression uniforme des couleurs
# cmap() retourne la couleur correspondante dans la palette choisie
# for i in range(len(sizes)) boucle sur chaque élément de valeurs
# cmap(i / len(sizes)) retourne une couleur pour chaque valeur de i
```

```
# Création du graphique
wedges, texts, autotexts = ax.pie(x=valeurs, labels=label, autopct='%.1f%%',␣
 ↪colors=colors)
# wedges contient les secteurs du piechart
# texts contient les labels des catégories
# autotexts contient les pourcentages

# Modification de la couleur des légendes
# On utilise une boucle parce que set_color ne peut s'appliquer à une lites, et␣
 ↪texs est une liste.
for text in texts:
    text.set_color('white')

plt.title('Répartition de la disponibilité intérieure en 2017', color='white',␣
 ↪fontsize=15, x=0.61)
plt.show
```

```
--------------------------------------------------------------------------------
------------------
En 2017, la disponibilité en Aliments pour animaux était de 1304.245 millions de
tonnes ( 13.24 %)
En 2017, les Pertes représentent 453.698 millions de tonnes ( 4.61 %)
En 2017, la disponibilité en Nourriture était de 4876.258 millions de tonnes (
49.51 %)
En 2017, la disponibilité en Semences était de 154.681 millions de tonnes ( 1.57
%)
En 2017, la disponibilité en Traitement était de 2204.687 millions de tonnes (
22.38 %)
En 2017, la disponibilité en Autres Utilisations était de 865.023 millions de
tonnes ( 8.78 %)
--------------------------------------------------------------------------------
------------------
```

[310]: <function matplotlib.pyplot.show(close=None, block=None)>



3.5 - Utilisation des céréales

[60]:
```
#Création d'une liste avec toutes les variables
dispo_alimentaire_population['Produit'].unique().tolist()
```

[60]:
```
['Abats Comestible',
 'Agrumes, Autres',
 'Aliments pour enfants',
 'Ananas',
 'Bananes',
 'Beurre, Ghee',
 'Bière',
 'Blé',
 'Boissons Alcooliques',
 'Café',
 'Coco (Incl Coprah)',
 'Crème',
 'Céréales, Autres',
 'Dattes',
 'Edulcorants Autres',
 'Feve de Cacao',
 'Fruits, Autres',
```

```
'Graines de coton',
'Graines de tournesol',
'Graisses Animales Crue',
'Huil Plantes Oleif Autr',
'Huile Graines de Coton',
"Huile d'Arachide",
"Huile d'Olive",
'Huile de Colza&Moutarde',
'Huile de Palme',
'Huile de Soja',
'Huile de Sésame',
'Huile de Tournesol',
'Lait - Excl Beurre',
'Légumes, Autres',
'Légumineuses Autres',
'Maïs',
'Miel',
'Millet',
'Miscellanees',
'Noix',
'Oeufs',
'Olives',
'Oranges, Mandarines',
'Orge',
'Plantes Oleiferes, Autre',
'Poissons Eau Douce',
'Poivre',
'Pommes',
'Pommes de Terre',
'Raisin',
'Riz (Eq Blanchi)',
'Sucre Eq Brut',
'Sucre, betterave',
'Sucre, canne',
'Sésame',
'Thé',
'Tomates',
"Viande d'Ovins/Caprins",
'Viande de Bovins',
'Viande de Volailles',
'Viande, Autre',
'Vin',
'Épices, Autres',
'Alcool, non Comestible',
'Animaux Aquatiques Autre',
'Arachides Decortiquees',
'Avoine',
```

```
 'Bananes plantains',
 'Boissons Fermentés',
 'Cephalopodes',
 'Citrons & Limes',
 'Crustacés',
 'Girofles',
 'Graines Colza/Moutarde',
 'Haricots',
 'Huile de Coco',
 'Huile de Germe de Maïs',
 'Huile de Palmistes',
 'Huiles de Foie de Poisso',
 'Huiles de Poissons',
 'Ignames',
 'Manioc',
 'Mollusques, Autres',
 'Oignons',
 'Palmistes',
 'Pamplemousse',
 'Patates douces',
 'Perciform',
 'Piments',
 'Plantes Aquatiques',
 'Pois',
 'Poissons Marins, Autres',
 'Poissons Pelagiques',
 'Racines nda',
 'Seigle',
 'Soja',
 'Sorgho',
 'Viande de Suides',
 'Huile de Son de Riz',
 'Sucre non centrifugé',
 'Viande de Anim Aquatiq']
```

[61]:
```
# Recherche de l'orthographe exact de céréale
dispo_alimentaire_population[dispo_alimentaire_population['Produit'].str.
 ↪contains('Céréal', na=False)].head(3)
```

[61]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)
12	Afghanistan	Céréales, Autres	vegetale	0.0	0.0	0.0
81	Afrique du Sud	Céréales, Autres	vegetale	8000000.0	0.0	1.0
176	Albanie	Céréales, Autres	vegetale	0.0		

```
176                 0.0                                                0.0

     Disponibilité alimentaire en quantité (kg/personne/an)  \
12                                                    0.00
81                                                    0.07
176                                                   0.08

     Disponibilité de matière grasse en quantité (g/personne/jour)  \
12                                                     0.0
81                                                     0.0
176                                                    0.0

     Disponibilité de protéines en quantité (g/personne/jour)  \
12                                                    0.00
81                                                    0.02
176                                                   0.01

     Disponibilité intérieure  …  Importations - Quantité  Nourriture  \
12                        0.0  …                      0.0         0.0
81                 12000000.0  …                3000000.0   4000000.0
176                       0.0  …                      0.0         0.0

     Pertes  Production  Semences  Traitement  Variation de stock   Année  \
12      0.0         0.0       0.0         0.0                 0.0  2017.0
81      0.0  19000000.0       0.0         0.0                 0.0  2017.0
176     0.0         0.0       0.0         0.0                 0.0  2017.0

     Population    dispo_kcal
12   36296113.0  0.000000e+00
81   57009756.0  2.080856e+10
176   2884169.0  0.000000e+00

[3 rows x 21 columns]
```

[62]:
```
#Création d'un dataframe avec les informations uniquement pour ces céréales

# Création d'une liste de céréales
cereales=['Céréales, Autres', 'Blé', 'Riz (Eq Blanchi)', 'Orge', 'Maïs',␣
 ↪'Seigle', 'Avoine', 'Millet', 'Sorgho']
dispo_cereale = dispo_alimentaire_population.
 ↪loc[dispo_alimentaire_population['Produit'].isin(cereales), :]
dispo_cereale.head(3)
```

[62]:
```
           Zone           Produit   Origine  Aliments pour animaux  \
7   Afghanistan              Blé  vegetale                    0.0
12  Afghanistan  Céréales, Autres  vegetale                    0.0
32  Afghanistan             Maïs  vegetale            200000000.0
```

```
    Autres Utilisations  Disponibilité alimentaire (Kcal/personne/jour)  \
7                   0.0                                           1369.0
12                  0.0                                              0.0
32                  0.0                                             21.0

    Disponibilité alimentaire en quantité (kg/personne/an)  \
7                                               160.23
12                                                0.00
32                                                2.50

    Disponibilité de matière grasse en quantité (g/personne/jour)  \
7                                                4.69
12                                               0.00
32                                               0.30

    Disponibilité de protéines en quantité (g/personne/jour)  \
7                                               36.91
12                                               0.00
32                                               0.56

    Disponibilité intérieure  …  Importations - Quantité    Nourriture  \
7               5.992000e+09  …             1.173000e+09  4.895000e+09
12              0.000000e+00  …             0.000000e+00  0.000000e+00
32              3.130000e+08  …             1.000000e+06  7.600000e+07

         Pertes    Production     Semences  Traitement  Variation de stock  \
7   775000000.0  5.169000e+09  322000000.0         0.0        -350000000.0
12          0.0  0.000000e+00          0.0         0.0                 0.0
32   31000000.0  3.120000e+08    5000000.0         0.0                 0.0

     Année  Population    dispo_kcal
7   2017.0  36296113.0  1.813662e+13
12  2017.0  36296113.0  0.000000e+00
32  2017.0  36296113.0  2.782097e+11

[3 rows x 21 columns]
```

[63]:
```
#Création d'un dataframe avec les informations uniquement pour ces céréales
# Vérifiactaion avec la table d'origine

# Création d'une liste de céréales
cereales=['Céréales, Autres', 'Blé', 'Riz (Eq Blanchi)', 'Orge', 'Maïs',␣
 ↪'Seigle', 'Avoine', 'Millet', 'Sorgho']
v_dispo_cereale = dispo_alimentaire.loc[dispo_alimentaire['Produit'].
 ↪isin(cereales), :].copy()
```

```
# Expression de la colonne 'Aliments pour animaux' en kg (exprimée en milliers␣
 ↪de tonnes)
v_dispo_cereale['Aliments pour animaux'] *= 1000000

v_dispo_cereale.head(3)
```

[63]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)	Disponibilité intérieure	Exportations - Quantité	Importations - Quantité	Nourriture	Pertes	Production	Semences	Traitement	Variation de stock
7	Afghanistan	Blé	vegetale	0.0	0.0	1369.0	160.23	4.69	36.91	5.992000e+09	0.0	1.173000e+09	4.895000e+09	775000000.0	5.169000e+09	322000000.0	0.0	-350000000.0
12	Afghanistan	Céréales, Autres	vegetale	0.0	0.0	0.0	0.00	0.00	0.00	0.000000e+00	0.0	0.000000e+00	0.000000e+00	0.0	0.000000e+00	0.0	0.0	0.0
32	Afghanistan	Maïs	vegetale	200000000.0	0.0	21.0	2.50	0.30	0.56	3.130000e+08	0.0	1.000000e+06	7.600000e+07	31000000.0	3.120000e+08	5000000.0	0.0	0.0

[64]:
```
#Affichage de la proportion d'alimentation animale par rapport au nombre␣
 ↪d'occurence
round(dispo_alimentaire_population['Origine'].value_counts(normalize =␣
 ↪True)*100, 1)
```

[64]:
```
Origine
vegetale    76.2
animale     23.8
Name: proportion, dtype: float64
```

[65]:
```
#Affichage de la proportion d'alimentation d'origine animale ou végétale par␣
 ↪rapport à la production
proportion_animale = dispo_alimentaire_population['Production'].
 ↪groupby(dispo_alimentaire_population['Origine']).sum()
round(proportion_animale / dispo_alimentaire_population['Production'].
 ↪sum()*100, 2)
```

[65]:
```
Origine
animale     13.71
vegetale    86.29
Name: Production, dtype: float64
```

[66]:
```
#Affichage de la proportion d'alimentation pour les animaux et pour les humains␣
 ↪par rapport à la disponibilité intérieure
proportion_animal = round(dispo_cereale['Aliments pour animaux'].sum() /␣
 ↪dispo_cereale['Disponibilité intérieure'].sum() * 100, 1)
proportion_humaine = round(dispo_cereale['Nourriture'].sum() /␣
 ↪dispo_cereale['Disponibilité intérieure'].sum() * 100, 1)
print(f'Les céréales alimentent pour {proportion_animal} % les animaux et pour␣
 ↪{proportion_humaine} % les humains ')
```

```
Les céréales alimentent pour 36.3 % les animaux et pour 42.8 % les humains
```

[67]:
```
#Affichage de la proportion d'alimentation animale par rapport à la production
# Vérification avec la table d'origine
v_proportion_animal = round(v_dispo_cereale['Aliments pour animaux'].sum() /␣
 ↪v_dispo_cereale['Disponibilité intérieure'].sum() * 100, 1)
v_proportion_humaine = round(v_dispo_cereale['Nourriture'].sum() /␣
 ↪v_dispo_cereale['Disponibilité intérieure'].sum() * 100, 1)
print(f'Les céréales alimentent pour {v_proportion_animal} % les animaux et␣
 ↪pour {v_proportion_humaine} % les humains ')
```

```
Les céréales alimentent pour 36.3 % les animaux et pour 42.8 % les humains
```

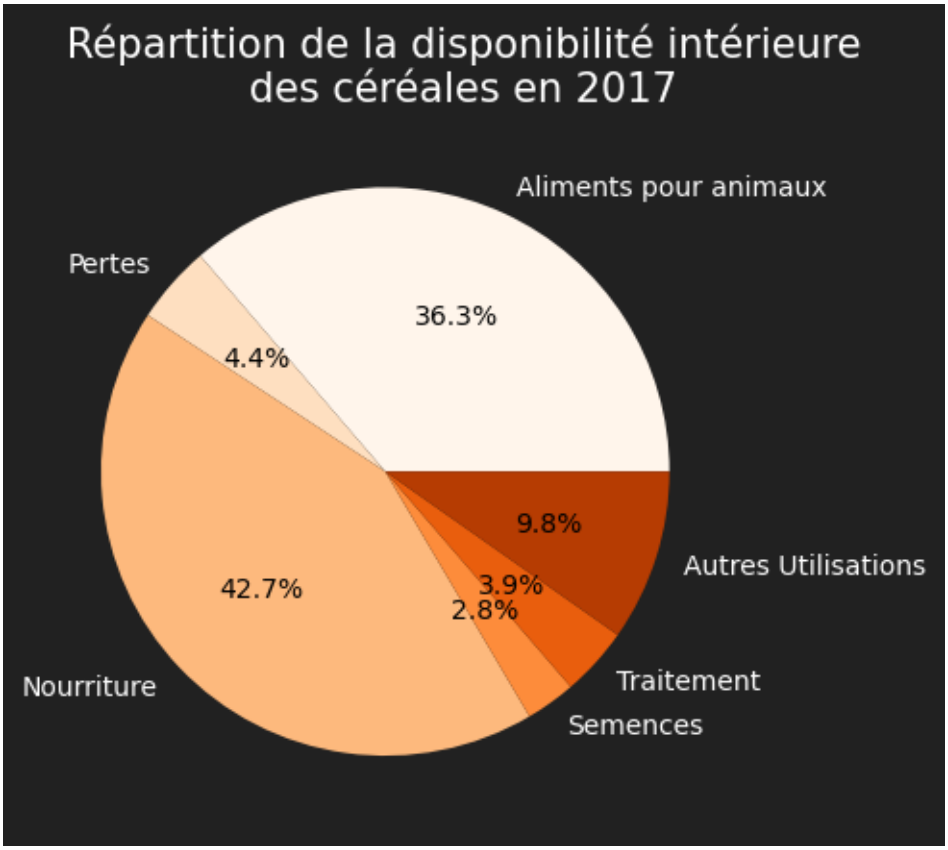
[312]:
```
# Proportion de l'utilisation des céréales pour humains et pour animaux
#Création de deux listes pour récupérer les valeurs des variables colonnes et␣
 ↪somme
valeurs_cereales=[]
```

```
label_cereales=[]

# Disponibilité intérieure totale pour les céréales
dispo_cereale_totale = dispo_cereale['Disponibilité intérieure'].sum()

print('-' * 93)
print('En 2017 et pour les céréales:\n')
for colonne in colonnes:  # colonnes est défini au chapitre précédent colonnes
 ↪= ['Aliments pour animaux', 'Pertes', 'Nourriture'
 ↪
    somme = dispo_cereale[colonne].sum()
 ↪  #, 'Semences', 'Traitement', 'Autres Utilisations']
    pct = round(somme / dispo_cereale_totale * 100, 2)
    valeurs_cereales.append(somme)
    label_cereales.append(colonne)
    if colonne == 'Pertes':
        print(' - les', colonne, 'représentaient', somme/ 1000000000,'millions
 ↪de tonnes (', pct,'%)')
    else:
        print(' - la disponibilté en', colonne,'était de', somme/
 ↪1000000000,'millions de tonnes (', pct, '%)')

print('-' * 93)
print()

# Création d'un piechart

# Création de la figure et de l'axe
fig, ax = plt.subplots()

# Modification de la couleur de fond
fig.patch.set_facecolor('#212121')

# Personnalisation des couleurs du graphe
cmap = plt.get_cmap('Oranges') # Récupère la palette orange srockée dans cmap
colors = [cmap(i / len(valeurs_cereales)) for i in range(len(valeurs_cereales))]
# colors = [cmap(i / 3) for i in range(3)] si l'on ne souhaite générer que 3
 ↪couleurs
# cmap est un clormap qui génère une couleur en fonction d'un nombre compris
 ↪entre 0 et 1
# i / len(valeurs) convertit l'indice i en une valeur normalisée entre 0 et 1
 ↪pour une progression uniforme des couleurs
# cmap() retourne la couleur correspondante dans la palette choisie
# for i in range(len(sizes)) boucle sur chaque élément de sizes
# cmap(i / len(sizes)) retourne une couleur pour chaque valeur de i
```

```
# Création du graphique
wedges, texts, autotexts = ax.pie(x=valeurs_cereales, labels=label_cereales,␣
 ↪autopct='%.1f%%', colors=colors)
# wedges contient les secteurs du piechart
# texts contient les labels des catégories
# autotexts contient les pourcentages

# Modification de la couleur des légendes
# On utilise une boucle parce que set_color ne peut s'appliquer à une lites, et␣
 ↪texs est une liste.
for text in texts:
    text.set_color('white')

plt.title(f'''Répartition de la disponibilité intérieure
des céréales en 2017''', color='white', fontsize=15, x=0.61)
plt.show
```

```
--------------------------------------------------------------------------------
-------------
En 2017 et pour les céréales:

 - la disponibilté en Aliments pour animaux était de 873.535 millions de tonnes
( 36.29 %)
 - les Pertes représentaient 107.12 millions de tonnes ( 4.45 %)
 - la disponibilté en Nourriture était de 1029.01 millions de tonnes ( 42.75 %)
 - la disponibilté en Semences était de 68.538 millions de tonnes ( 2.85 %)
 - la disponibilté en Traitement était de 94.589 millions de tonnes ( 3.93 %)
 - la disponibilté en Autres Utilisations était de 234.787 millions de tonnes (
9.75 %)
--------------------------------------------------------------------------------
-------------
```

[312]: <function matplotlib.pyplot.show(close=None, block=None)>



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

[70]:
```
population.head(3)
```

[70]:

	Zone	Année	Population
0	Afghanistan	2013	32269589.0
1	Afghanistan	2014	33370794.0
2	Afghanistan	2015	34413603.0

[71]:
```
sous_nutrition.head(3)
```

[71]:

	Zone	Année	Sous_nutrition
0	Afghanistan	2012-2014	8600000.0
1	Afghanistan	2013-2015	8800000.0
2	Afghanistan	2014-2016	8900000.0

[72]:
```
# Création de la colonne proportion par pays

# Jointure des tables  sous nutrtiion et population pour l'année 2017
sous_nutrition_population_2017 = (pd.merge(sous_nutrition.
 ↪loc[sous_nutrition['Année']== '2016-2018', ['Zone', 'Sous_nutrition']],
```

```
                                                population.loc[population['Année']␣
 ↪== 2017, :] , on='Zone', how='inner')
                                    )

# Création de la colonne proportion
sous_nutrition_population_2017['Proportion %'] =␣
 ↪(round(sous_nutrition_population_2017['Sous_nutrition'] /
                                                          ␣
 ↪sous_nutrition_population_2017['Population']*100, 2)
                                                  )

sous_nutrition_population_2017.head(3)
```

[72]:

	Zone	Sous_nutrition	Année	Population	Proportion %
0	Afghanistan	10500000.0	2017	36296113.0	28.93
1	Afrique du Sud	3100000.0	2017	57009756.0	5.44
2	Albanie	100000.0	2017	2884169.0	3.47

[73]:
```
#affichage après trie des 10 pires pays
sous_nutrition_10_pays = sous_nutrition_population_2017.sort_values('Proportion␣
 ↪%', ascending = False).head(10)
sous_nutrition_10_pays
```

[73]:

	Zone	Sous_nutrition	Année	Population	Proportion %
78	Haïti	5300000.0	2017	10982366.0	48.26
157	République populaire démocratique de Corée	12000000.0	2017	25429825.0	47.19
108	Madagascar	10500000.0	2017	25570512.0	41.06
103	Libéria	1800000.0	2017	4702226.0	38.28
100	Lesotho	800000.0	2017	2091534.0	38.25
183	Tchad	5700000.0	2017	15016753.0	37.96
161	Rwanda	4200000.0	2017	11980961.0	35.06
121	Mozambique	9400000.0	2017	28649018.0	32.81
186	Timor-Leste	400000.0	2017	1243258.0	32.17
0	Afghanistan	10500000.0	2017	36296113.0	28.93

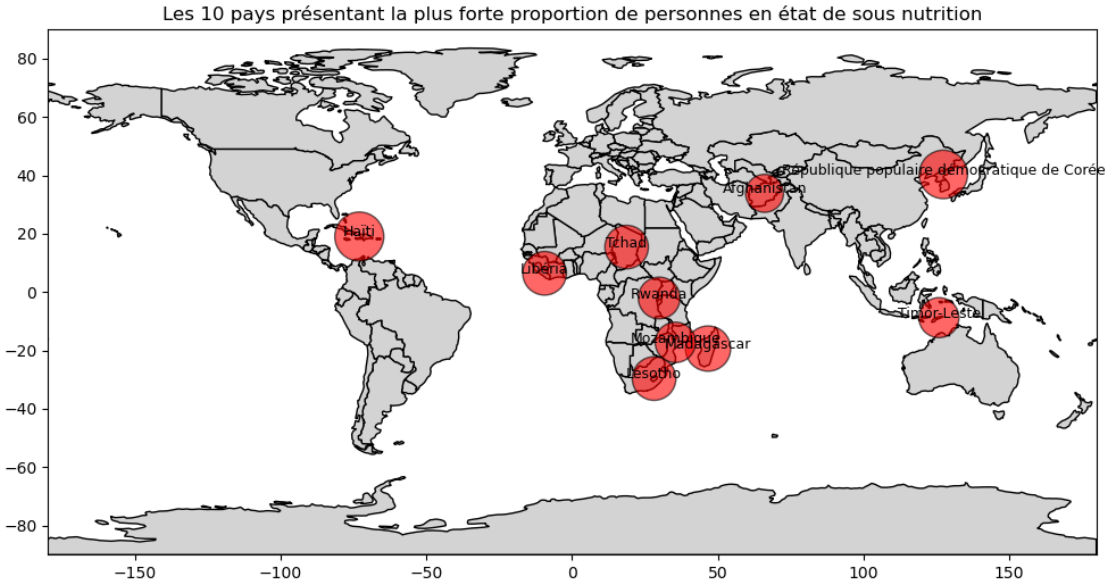
[74]:
```
# Calcul de la proportion de personne sous alimentées en 2017
sous_nutrition_totale=(round(sous_nutrition_population_2017['Sous_nutrition'].
 ↪sum() /
                             sous_nutrition_population_2017['Population'].
 ↪sum()*100, 0)
                       )
print (sous_nutrition_totale, '% de la population mondiale est en état de sous␣
 ↪nutrition en 2017')
```

```
7.0 % de la population mondiale est en état de sous nutrition en 2017
```

[75]:
```
# Création d'une carte pour afficher les pays

# Import de la librairie geopandas
import geopandas as gpd
import geodatasets

# Chargement des pays du monde
url = "https://naturalearth.s3.amazonaws.com/110m_cultural/
 ↪ne_110m_admin_0_countries.zip"
world = gpd.read_file(url)

# Récupération des coordonnées des pays
coordonnees = world[['NAME', 'geometry']].copy()

# Extraire les coordonnées
coordonnees['latitude'] = coordonnees['geometry'].centroid.y
coordonnees['longitude'] = coordonnees['geometry'].centroid.x

# Création de la table sous_nutrition_10_pays jointe avec la table coordonnées
sous_nutrition_carte = pd.merge(sous_nutrition_10_pays, coordonnees,␣
 ↪left_on='Zone', right_on='NAME', how='left')

# Correction des NaN
sous_nutrition_carte.loc[sous_nutrition_carte['Zone'] == 'Haïti', 'latitude'] =␣
 ↪19.054426
sous_nutrition_carte.loc[sous_nutrition_carte['Zone'] == 'Haïti', 'longitude']␣
 ↪= -73.04597
sous_nutrition_carte.loc[sous_nutrition_carte['Zone'] == 'République populaire␣
 ↪démocratique de Corée', 'latitude'] = 40.339693
sous_nutrition_carte.loc[sous_nutrition_carte['Zone'] == 'République populaire␣
 ↪démocratique de Corée', 'longitude'] = 127.495377
sous_nutrition_carte.loc[sous_nutrition_carte['Zone'] == 'Libéria', 'latitude']␣
 ↪= 6.452424
sous_nutrition_carte.loc[sous_nutrition_carte['Zone'] == 'Libéria',␣
 ↪'longitude'] = -9.428598
```

```
sous_nutrition_carte.loc[sous_nutrition_carte['Zone'] == 'Tchad', 'latitude'] =␣
 ↪15.445719
sous_nutrition_carte.loc[sous_nutrition_carte['Zone'] == 'Tchad', 'longitude']␣
 ↪= 18.738068

# Créatiun de la figure
fig, ax = plt.subplots(figsize = (12,6))

# Dessiner la carte du monde
world.plot(ax = ax, color = 'lightgrey', edgecolor = 'black')

# Obtenir les coordonnées des pays
pays = world[world['NAME'] == 'Pays']

# Ajouter des points porportionnels aux valeurs
plt.scatter(sous_nutrition_carte['longitude'],␣
 ↪sous_nutrition_carte['latitude'], s = sous_nutrition_carte['Proportion %'] *␣
 ↪20,
            color = 'red', alpha = 0.6, edgecolors='black', zorder=3)

# Ajouter des labels
for i, row in sous_nutrition_carte.iterrows():
    plt.text(row['longitude'], row['latitude'], row['Zone'], fontsize=9,␣
 ↪ha='center', zorder=4)

# Ajuster les limites de la carte
ax.set_xlim([-180, 180])
ax.set_ylim([-90, 90])

# Titre
plt.title('Les 10 pays présentant la plus forte proportion de personnes en état␣
 ↪de sous nutrition')

plt.show()
```

```
/var/folders/41/p4j85gx56_z0npf_rntd6ht00000gn/T/ipykernel_25377/319914634.py:15
: UserWarning: Geometry is in a geographic CRS. Results from 'centroid' are
likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometries to a
projected CRS before this operation.

  coordonnees['latitude'] = coordonnees['geometry'].centroid.y
/var/folders/41/p4j85gx56_z0npf_rntd6ht00000gn/T/ipykernel_25377/319914634.py:16
: UserWarning: Geometry is in a geographic CRS. Results from 'centroid' are
likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometries to a
projected CRS before this operation.

  coordonnees['longitude'] = coordonnees['geometry'].centroid.x
```



[76]:
```
# Affichage d'une liste des pays et des proportions pour intégrer à la présentation
liste_proportion_pays = sous_nutrition_carte[['Zone','Proportion %']].values.tolist()
print(liste_proportion_pays)
```

```
[['Haïti', 48.26], ['République populaire démocratique de Corée', 47.19],
['Madagascar', 41.06], ['Libéria', 38.28], ['Lesotho', 38.25], ['Tchad', 37.96],
['Rwanda', 35.06], ['Mozambique', 32.81], ['Timor-Leste', 32.17],
['Afghanistan', 28.93]]
```

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

[78]:
```
aide_alimentaire.head(3)
```

[78]:

	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

[79]:
```
# Calcul du total de l'aide alimentaire par pays

# Remplacement du nom de la colonne Valeur par aide alimentaire
aide_alimentaire.rename(columns = {'Valeur':'Aide_alimentaire'}, inplace=True)

# Calcul de l'aide par pays
aide_alimentaire_pays = aide_alimentaire.groupby(aide_alimentaire['Zone']).agg({'Aide_alimentaire':'sum'}).reset_index()
```

```
aide_alimentaire_pays.head(3)
```

[79]:

	Zone	Aide_alimentaire
0	Afghanistan	185452000
1	Algérie	81114000
2	Angola	5014000

[80]:
```
#affichage après trie des 10 pays qui ont bénéficié le plus de l'aide␣
 ↪alimentaire
aide_alimentaire_pays_trie = aide_alimentaire_pays.
 ↪sort_values('Aide_alimentaire', ascending=False).head(10)

# Affichage des valeurs en millions de tonnes
aide_alimentaire_pays_trie['Aide_alimentaire'] /= 1000000

# Précision de l'unité
aide_alimentaire_pays_trie.rename(columns = {'Aide_alimentaire':␣
 ↪'Aide_alimentaire (millions de tonnes)'}, inplace=True)

# Remplacement de 'République démocratique du Congo' par 'Congo' pour une␣
 ↪meilleure visibilté sur le graphe
aide_alimentaire_pays_trie.loc[aide_alimentaire_pays_trie['Zone'] ==␣
 ↪'République démocratique du Congo', 'Zone'] = 'Congo'


aide_alimentaire_pays_trie
```

[80]:

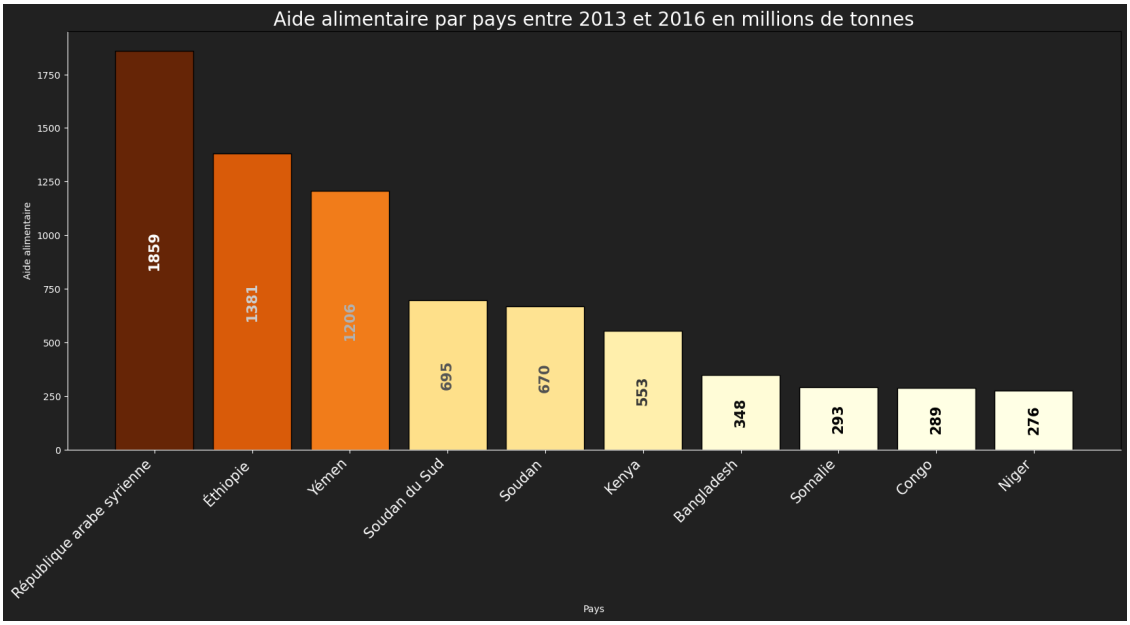
	Zone	Aide_alimentaire (millions de tonnes)
50	République arabe syrienne	1858.943
75	Éthiopie	1381.294
70	Yémen	1206.484
61	Soudan du Sud	695.248
60	Soudan	669.784
30	Kenya	552.836
3	Bangladesh	348.188
59	Somalie	292.678
53	Congo	288.502
43	Niger	276.344

[81]:
```
# import de matplotlib.cm pour créer un dégradé de couleur
import matplotlib.cm as cm

# Appliquer un dégradé de couleur
norm = plt.Normalize(min(aide_alimentaire_pays_trie['Aide_alimentaire (millions␣
 ↪de tonnes)']),
```

```
                    max(aide_alimentaire_pays_trie['Aide_alimentaire (millions␣
 ↪de tonnes)']))

# Appliquer un dégradé aux barres
colors = cm.YlOrBr(norm(aide_alimentaire_pays_trie['Aide_alimentaire (millions␣
 ↪de tonnes)']))

# Appliquer un dégradé aux valeurs sur les barres
text_colors = cm.Greys_r(norm(aide_alimentaire_pays_trie['Aide_alimentaire␣
 ↪(millions de tonnes)']))

# fig et ax permettent de personnaliser facilement le graphe
fig, ax = plt.subplots(figsize=(20,8))

# Personnalisation du graphe
fig.patch.set_facecolor('#212121')  # Modifie de la couleur autour du graphe
ax.set_facecolor('#212121')  # Modifie de la couleur du fond du graphe
ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe des x
ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
ax.tick_params(axis='y', colors='white')  # Modifie la couleur des valeurs de␣
 ↪l'axe des y
ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des x
ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des y
#ax.title.set_color('White')  # Modifie la couleur du titre
ax.set_title('Aide alimentaire par pays entre 2013 et 2016 en millions de␣
 ↪tonnes',fontsize=20, color='white')


# Création d'un diagramme en barre
bars = ax.bar(height = aide_alimentaire_pays_trie['Aide_alimentaire (millions␣
 ↪de tonnes)'], x = aide_alimentaire_pays_trie['Zone'],
              color=colors, edgecolor='black')

# inscrire les valeurs sur les barres
for bar, text_color in zip(bars, text_colors):
    yval = bar.get_height()  # Récupère la valeur de la barre
    plt.text(bar.get_x() + bar.get_width() / 2, yval / 2, round(yval, ),
             ha = 'center', va = 'center', fontsize = 15, color = text_color,␣
 ↪fontweight='bold', rotation = 90)

# bar.get_x() + bar.get_width() / 2  centre le texte sur la barre
# yval / 2 place le texte au milieur de la barre
# ha = 'center'  centre le texte horizontalement
# va = 'center' place le texte au centre de la barre
```

```
plt.xticks(rotation = 45, ha = 'right')  # Rotation des noms des pays
plt.xlabel('Pays')
plt.ylabel('Aide alimentaire')

plt.show
```

[81]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



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

[83]:
```
#Création d'un dataframe avec la zone, l'année et l'aide alimentaire puis␣
 ↪groupby sur zone et année
aide_alimentaire_periode = aide_alimentaire.groupby(['Zone', 'Année']).
 ↪agg({'Aide_alimentaire':'sum'}).reset_index()

aide_alimentaire_periode.head(3)
```

[83]:

	Zone	Année	Aide_alimentaire
0	Afghanistan	2013	128238000
1	Afghanistan	2014	57214000
2	Algérie	2013	35234000

[84]:
```
# Création d'une liste contenant les 5 pays qui ont le plus bénéficiées de␣
 ↪l'aide alimentaire
# Détermination des 5 pays ayant reçu le plus d'aide alimentaire entre 2013 et␣
 ↪2016
```

```
# Sélection des années entre 2013 et 2016
aide_alimentaire_2013_2016 = aide_alimentaire_periode.
 ↪loc[aide_alimentaire_periode['Année'].between (2013, 2016), :]
# Je n'avais pas besoin de faire ça, les années du df sont toutes comprises␣
 ↪entre 2103 et 2106,
# mais c'était cool d'apprendre between, alors je le garde.

# Somme de l'aide alimentaire perçue par pays entre 2013 et 2016
aide_alimentaire_2013_2016_total = aide_alimentaire_2013_2016.groupby(['Zone']).
 ↪agg({'Aide_alimentaire':'sum'}).reset_index()

# Tri des 5 pays ayant bénéficié le plus d'aide alimentaire et création de la␣
 ↪liste
liste_5_pays=aide_alimentaire_2013_2016_total.sort_values('Aide_alimentaire',␣
 ↪ascending=False).head(5)['Zone'].tolist()

print(liste_5_pays)
```

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

[85]:
```
#On filtre sur le dataframe avec notre liste
aide_alimentaire_periode_trie = aide_alimentaire_periode.
 ↪loc[aide_alimentaire_periode['Zone'].isin(liste_5_pays), :]
```

[86]:
```
# Affichage des pays avec l'aide alimentaire par année
aide_alimentaire_periode_trie
```

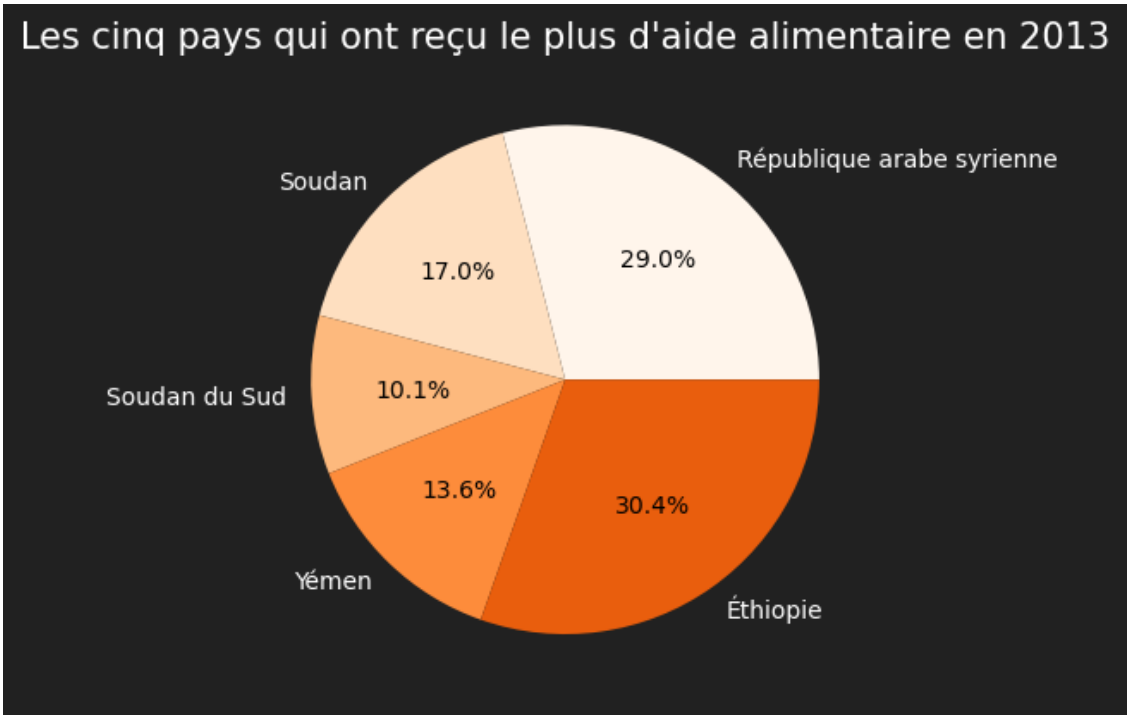
[86]:

	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

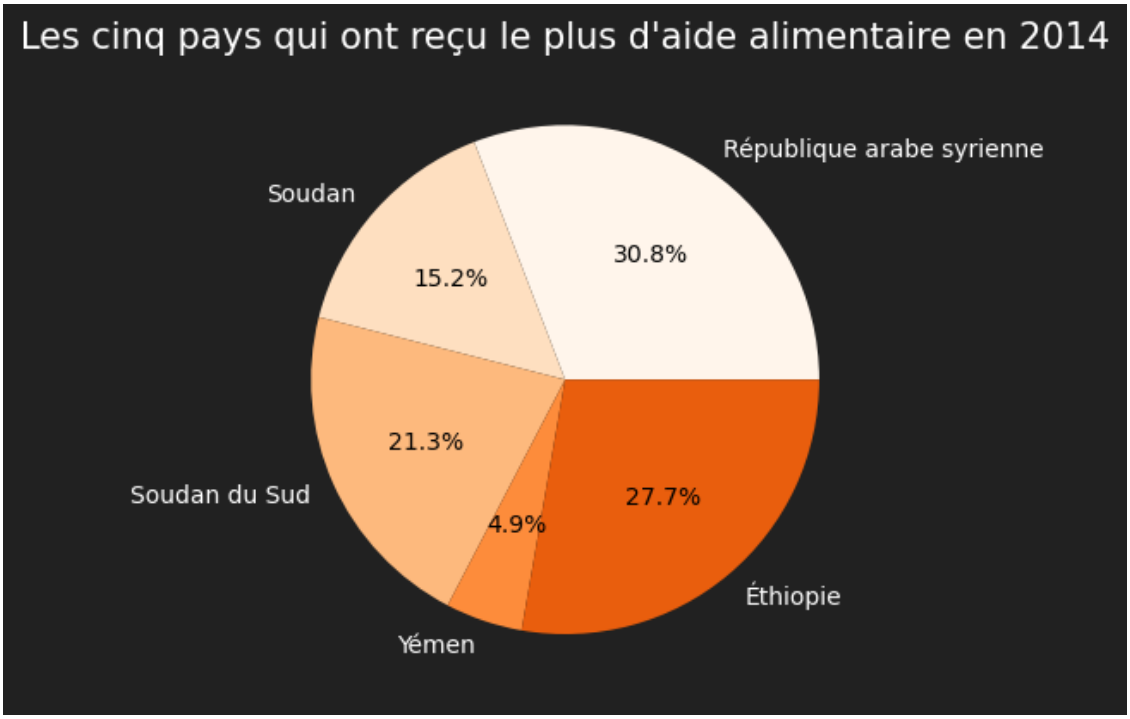
[314]:
```
# Création de tables distinctes par année pour faire des graphhes

# Récupération dans une liste des valeurs uniques de la colonne Année
annees = aide_alimentaire_periode_trie['Année'].unique().tolist()

# Création d'un dictionnaire pour stoker les dataframes
aide_alimentaire_par_annee = {}

# Boucle for pour la création des dataframe
for annee in annees:
    aide_alimentaire_par_annee[annee] = aide_alimentaire_periode_trie.
 ↪loc[aide_alimentaire_periode_trie['Année'] == annee, :]

# Création d'une boucle for pour la génération de graphiques
for annee in annees:
    plt.figure(figsize=(6,4))  # Permet de générer un graphe distinct à chaque␣
 ↪boucle
    # Création de la figure et de l'axe
    fig, ax = plt.subplots()

# Modification de la couleur de fond
    fig.patch.set_facecolor('#212121')

# Personnalisation des couleurs du graphe
    cmap = plt.get_cmap('Oranges') # Récupère la palette viridis srockée dans␣
 ↪cmap
    colors = [cmap(i / len(valeurs)) for i in range(len(valeurs))]
# colors = [cmap(i / 3) for i in range(3)] si l'on ne souhaite générer que 3␣
 ↪couleurs
# cmap est un clormap qui génère une couleur en fonction d'un nombre compris␣
 ↪entre 0 et 1
# i / len(valeurs) convertit l'indice i en une valeur normalisée entre 0 et 1␣
 ↪pour une progression uniforme des couleurs
# cmap() retourne la couleur correspondante dans la palette choisie
# for i in range(len(sizes)) boucle sur chaque élément de sizes
# cmap(i / len(sizes)) retourne une couleur pour chaque valeur de i

# Création du graphique
    wedges, texts, autotexts = (ax.
 ↪pie(x=aide_alimentaire_par_annee[annee]['Aide_alimentaire'],
                                    ␣
 ↪labels=aide_alimentaire_par_annee[annee]['Zone'],
                                      autopct='%.1f%%', colors=colors)
                                )
# wedges contient les secteurs du piechart
# texts contient les labels des catégories
```

```
# autotexts contient les pourcentages

# Modification de la couleur des légendes
# On utilise une boucle parce que set_color ne peut s'appliquer à une lites, et texs est une liste.
    for text in texts:
        text.set_color('white')

    #plt.pie(x=aide_alimentaire_par_annee[annee]['Aide_alimentaire'], labels=aide_alimentaire_par_annee[annee]['Zone'], autopct='%.2f%%')
    plt.title(f'Les cinq pays qui ont reçu le plus d\'aide alimentaire en {annee}', color='white', fontsize=15)
    plt.show
```

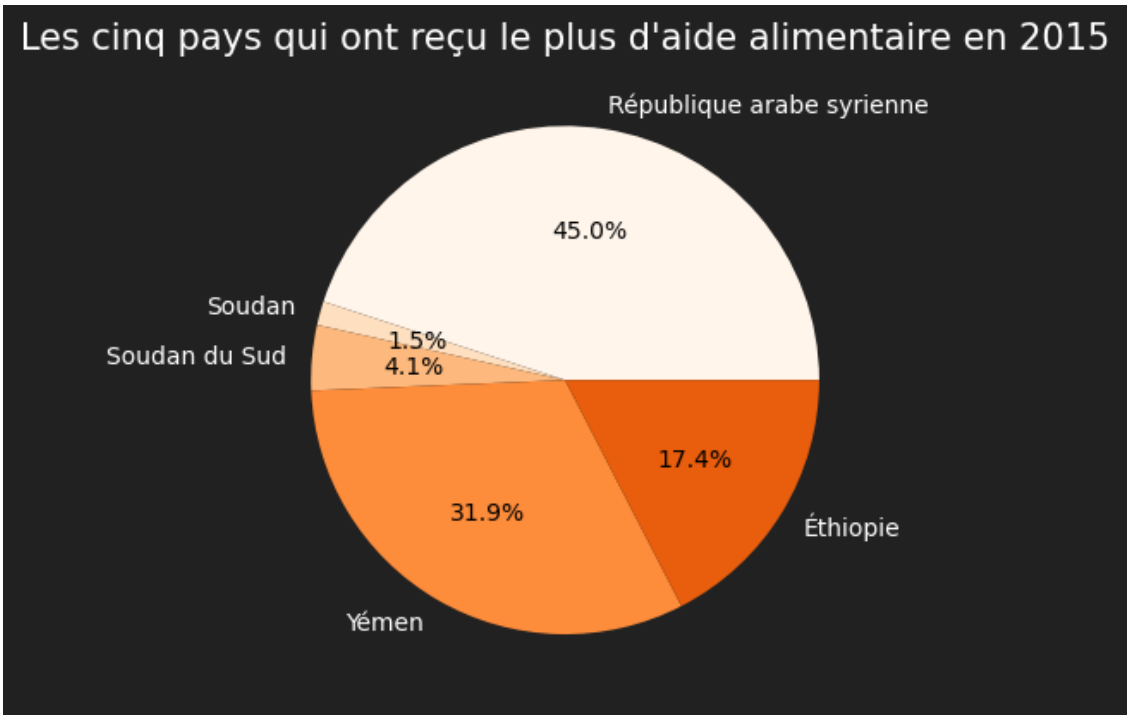
<Figure size 600x400 with 0 Axes>



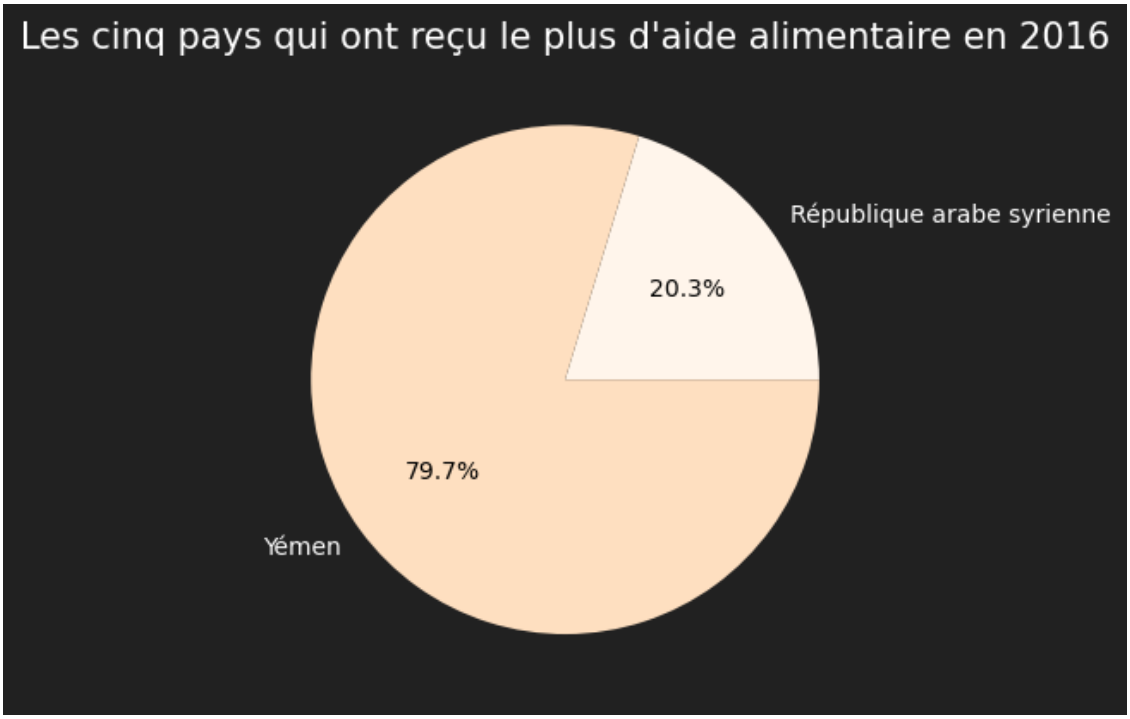
<Figure size 600x400 with 0 Axes>




```
<Figure size 600x400 with 0 Axes>
```



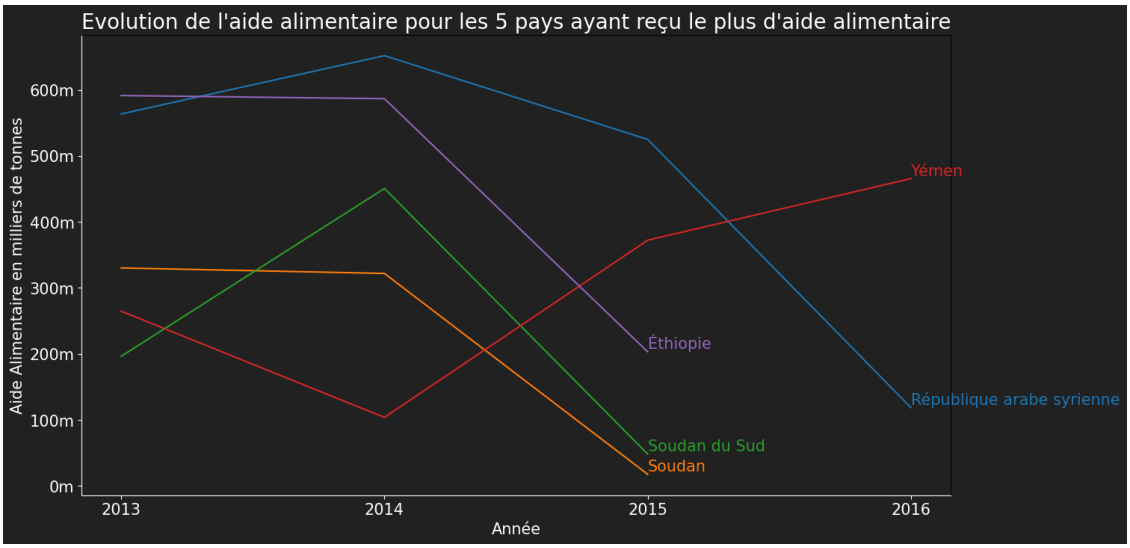
```
<Figure size 600x400 with 0 Axes>
```



[88]:
```
# Création de courbes pour chaque pays

# Import de MaxNLocator pour le forçage des années en entiers
from matplotlib.ticker import MaxNLocator

# Import de FuncFormatter pour forcer l'affichage des ordonnées en milliers de␣
 ↪tonnes
from matplotlib.ticker import FuncFormatter

# Création d'une liste des pays
pays_unique = aide_alimentaire_periode_trie['Zone'].unique()

fig, ax = plt.subplots(figsize=(15,8))

# Boucle pour générer les courbes
for pays in pays_unique:
    pays_data = aide_alimentaire_periode_trie.
 ↪loc[aide_alimentaire_periode_trie['Zone'] == pays]
    line, = plt.plot(pays_data['Année'], pays_data['Aide_alimentaire'],␣
 ↪label=pays)  # Récupère les données pour chaque courbe
    # La virgule permet de décomposer la liste retournée par plt.plot
```

```
    # Persoonnalisation du graphe
    fig.patch.set_facecolor('#212121')
    ax.set_facecolor('#212121')
    ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe␣
 ↪des x
    ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
    ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la␣
 ↪couleur des valeurs de l'axe des x
    ax.tick_params(axis='y', labelsize=15, colors='white')  # Modifie la␣
 ↪couleur des valeurs de l'axe des y
    ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe␣
 ↪des x
    ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe␣
 ↪des y
    ax.xaxis.label.set_size(15)  # Modifie la taille du label de l'axe des x
    ax.yaxis.label.set_size(15)  # Modifie la taille de l'axe des y
    plt.text(pays_data['Année'].iloc[-1],  # On prend la dernière année pour␣
 ↪positionner le texte en fin de courbe
             pays_data['Aide_alimentaire'].iloc[-1],  # On prend la dernière␣
 ↪valeur d'Aide alimentaire pour positionnr le texte en y
             pays,  # On récupère la valeur de pays
             color=line.get_color(),  # On récupère la couleur de la courbe pour␣
 ↪l'appliquer au texte
             fontsize=15,  # Taille du texte
             verticalalignment='bottom')  # On aligne le texte sous la position␣
 ↪donnée
# plt.text permet d'ajouter du texte à une position spécifique dans le graphique


# Afficher le nom des axes, la légende et le titre
plt.xlabel('Année')
plt.ylabel('Aide Alimentaire en milliers de tonnes')
plt.title('Evolution de l\'aide alimentaire pour les 5 pays ayant reçu le plus␣
 ↪d\'aide alimentaire', color='white', fontsize=20)

# Forcer l'affichage des années comme des entiers. Le premier afficahge affiche␣
 ↪les années comme des déciamux
#avec des moitiés d'années comme 2013,5
plt.gca().xaxis.set_major_locator(MaxNLocator(integer=True))

# Forcer l'affichage des ordonnées en milliers de tonnes
def scale_y_ticks(y, pos):
    return f"{int(y / 1000000)}m"
ax.yaxis.set_major_formatter(FuncFormatter(scale_y_ticks))

plt.show
```

[88]: <function matplotlib.pyplot.show(close=None, block=None)>



3.9 - Pays avec le moins de disponibilité par habitant

[90]:
```
#Calcul de la disponibilité en kcal par personne par jour par pays
dispo_alimentaire_kcal = (dispo_alimentaire_population.groupby('Zone').
                         agg({'Disponibilité alimentaire (Kcal/personne/jour)':
 ↪'sum'}).reset_index()
                        )

dispo_alimentaire_kcal.head(3)
```

[90]:

	Zone	Disponibilité alimentaire (Kcal/personne/jour)
0	Afghanistan	2087.0
1	Afrique du Sud	3020.0
2	Albanie	3188.0

[91]:
```
#Affichage des 10 pays qui ont le moins de dispo alimentaire par personne
dispo_alimentaire_kcal_trie_croiss = (dispo_alimentaire_kcal.
 ↪sort_values('Disponibilité alimentaire (Kcal/personne/jour)',

 ↪ascending=True).head(10)
                                     )

# Modification des noms des colonnes pour la présentation
dispo_alimentaire_kcal_trie_croiss.rename(columns = {'Zone':'Pays'},␣
 ↪inplace=True)
```

```
# Modification du nom de la République populaire démocratique de Corée en␣
 ↪Corrée du Nord
dispo_alimentaire_kcal_trie_croiss.loc[
    dispo_alimentaire_kcal_trie_croiss['Pays'] == 'République populaire␣
 ↪démocratique de Corée', ['Pays']] = 'Corrée du Nord'

dispo_alimentaire_kcal_trie_croiss
```

[91]:

	Pays	Disponibilité alimentaire (Kcal/personne/jour)
128	République centrafricaine	1879.0
166	Zambie	1924.0
91	Madagascar	2056.0
0	Afghanistan	2087.0
65	Haïti	2089.0
133	Corrée du Nord	2093.0
151	Tchad	2109.0
167	Zimbabwe	2113.0
114	Ouganda	2126.0
154	Timor-Leste	2129.0

[92]:
```
# Vérification des résultats avec la table d'origine

# Aggrégation des disponibilités en kcal/personne/jour par pays
v_dispo_alimentaire_kcal = dispo_alimentaire.groupby('Zone').
 ↪agg({'Disponibilité alimentaire (Kcal/personne/jour)':'sum'}).reset_index()

# tri par ordre croissant
v_dispo_alimentaire_kcal = (v_dispo_alimentaire_kcal.sort_values('Disponibilité␣
 ↪alimentaire (Kcal/personne/jour)',
                                                                 ␣
 ↪          ascending=True)
                                        )

v_dispo_alimentaire_kcal.head(10)

# Les résultats sont identiques
```

[92]:

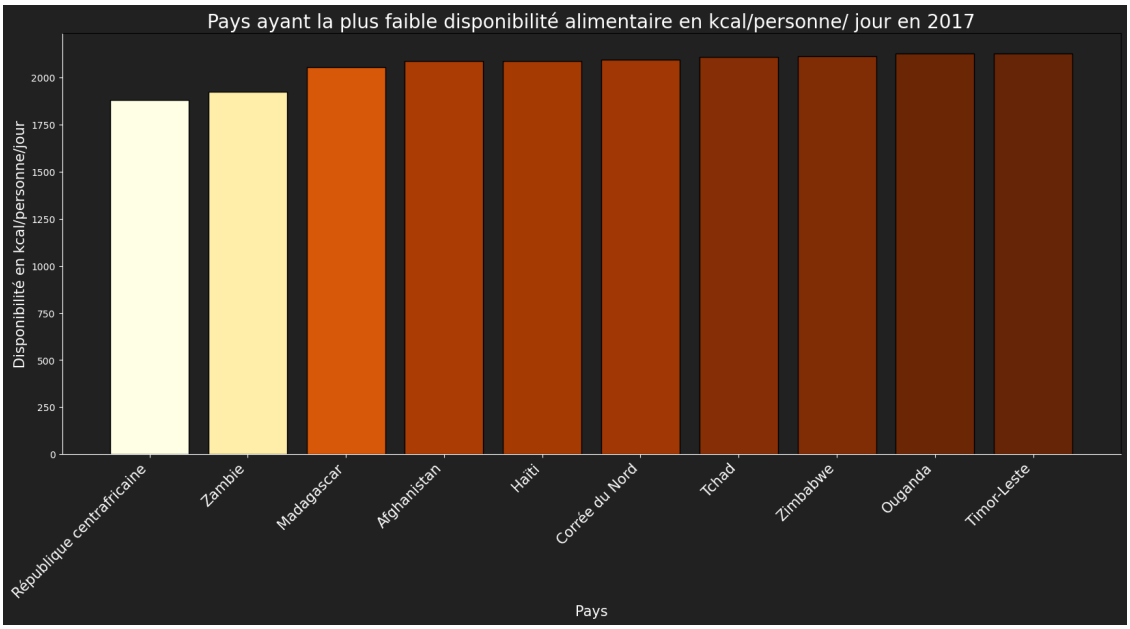
	Zone
128	République centrafricaine
166	Zambie
91	Madagascar
0	Afghanistan
65	Haïti
133	République populaire démocratique de Corée
151	Tchad
167	Zimbabwe
114	Ouganda

```
154                                  Timor-Leste

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

[93]:
```
# import de matplotlib.cm pour créer un dégradé de couleur
import matplotlib.cm as cm

# Appliquer un dégradé de couleur
norm = plt.Normalize(min(dispo_alimentaire_kcal_trie_croiss['Disponibilité␣
 ↪alimentaire (Kcal/personne/jour)']),
                     max(dispo_alimentaire_kcal_trie_croiss['Disponibilité␣
 ↪alimentaire (Kcal/personne/jour)']))

# Appliquer un dégradé aux barres
colors = cm.YlOrBr(norm(dispo_alimentaire_kcal_trie_croiss['Disponibilité␣
 ↪alimentaire (Kcal/personne/jour)']))

# Appliquer un dégradé aux valeurs sur les barres
text_colors = cm.Greys_r(norm(dispo_alimentaire_kcal_trie_croiss['Disponibilité␣
 ↪alimentaire (Kcal/personne/jour)']))

# fig et ax permettent de personnaliser plus facilement le graphe
fig, ax = plt.subplots(figsize=(20,8))

#  Personnalisation du graphique
fig.patch.set_facecolor('#212121')  # Modifie de la couleur autour du graphe
ax.set_facecolor('#212121')  # Modifie de la couleur du fond du graphe
ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe des x
ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
ax.tick_params(axis='y', colors='white')  # Modifie la couleur des valeurs de␣
 ↪l'axe des y
ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des x
ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des y
#ax.title.set_color('White')  # Modifie la couleur du titre
```

```
ax.set_title('Pays ayant la plus faible disponibilité alimentaire en kcal/
 ↪personne/ jour en 2017',fontsize=20, color='white')


# Création d'un diagramme en barre
bars = (ax.bar(height = dispo_alimentaire_kcal_trie_croiss['Disponibilité␣
 ↪alimentaire (Kcal/personne/jour)'],
               x = dispo_alimentaire_kcal_trie_croiss['Pays'],
               color=colors, edgecolor='black')
       )

plt.xticks(rotation = 45, ha = 'right')  # Rotation des noms des pays
plt.xlabel('Pays', fontsize=15)
plt.ylabel('Disponibilité en kcal/personne/jour', fontsize = 15)

plt.show
```

[93]: <function matplotlib.pyplot.show(close=None, block=None)>



3.10 - Pays avec le plus de disponibilité par habitant

[95]:
```
#Affichage des 10 pays qui ont le plus de dispo alimentaire par personne
dispo_alimentaire_kcal_trie_decroiss = (dispo_alimentaire_kcal.
 ↪sort_values('Disponibilité alimentaire (Kcal/personne/jour)',

                                                                    ␣
 ↪ascending=False).head(10)
                                        )
```

```
# Renommer les colonnes pour la présentation
dispo_alimentaire_kcal_trie_decroiss.rename(columns = {'Zone':'Pays'},␣
 ↪inplace=True)

dispo_alimentaire_kcal_trie_decroiss
```

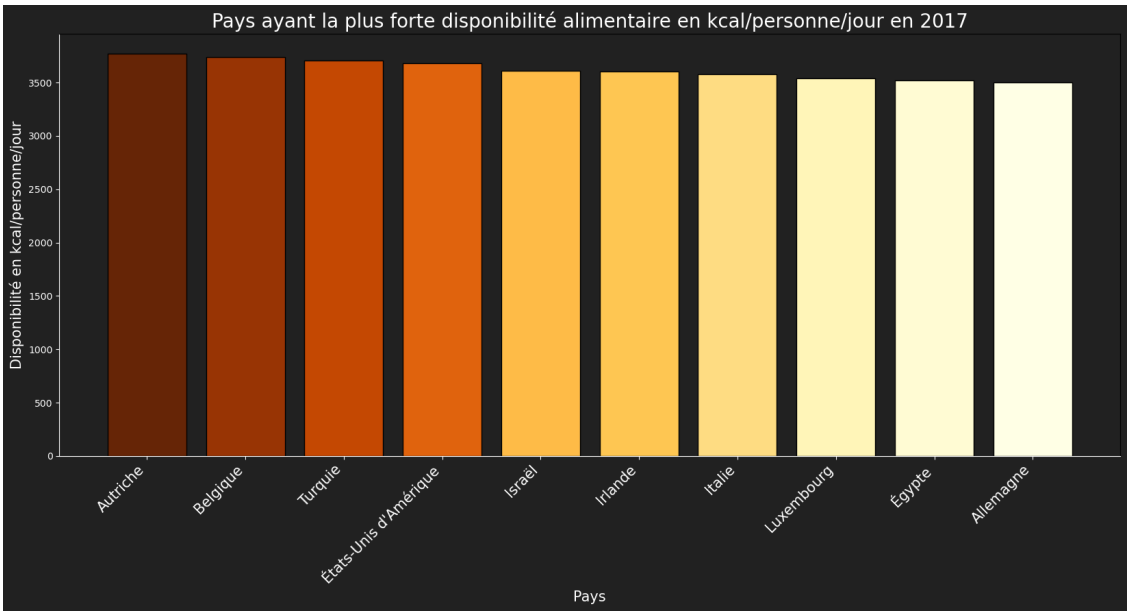
[95]:

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

[96]:
```
# import de matplotlib.cm pour créer un dégradé de couleur
import matplotlib.cm as cm

# Appliquer un dégradé de couleur
norm = plt.Normalize(min(dispo_alimentaire_kcal_trie_decroiss['Disponibilité␣
 ↪alimentaire (Kcal/personne/jour)']),
                     max(dispo_alimentaire_kcal_trie_decroiss['Disponibilité␣
 ↪alimentaire (Kcal/personne/jour)']))

# Appliquer un dégradé aux barres
colors = cm.YlOrBr(norm(dispo_alimentaire_kcal_trie_decroiss['Disponibilité␣
 ↪alimentaire (Kcal/personne/jour)']))

# Appliquer un dégradé aux valeurs sur les barres
text_colors = cm.
 ↪Greys_r(norm(dispo_alimentaire_kcal_trie_decroiss['Disponibilité alimentaire␣
 ↪(Kcal/personne/jour)']))

# fig et ax permettent de personnaliser plus facilement le graphe
fig, ax = plt.subplots(figsize=(20,8))

#  Personnalisation du graphique
fig.patch.set_facecolor('#212121')  # Modifie de la couleur autour du graphe
ax.set_facecolor('#212121')  # Modifie de la couleur du fond du graphe
ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe des x
ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
```

```
ax.tick_params(axis='y', colors='white')  # Modifie la couleur des valeurs de␣
 ↪l'axe des y
ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des x
ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des y
#ax.title.set_color('White')  # Modifie la couleur du titre
ax.set_title('Pays ayant la plus forte disponibilité alimentaire en kcal/
 ↪personne/jour en 2017',fontsize=20, color='white')


# Création d'un diagramme en barre
bars = (ax.bar(height = dispo_alimentaire_kcal_trie_decroiss['Disponibilité␣
 ↪alimentaire (Kcal/personne/jour)'],
                        x = dispo_alimentaire_kcal_trie_decroiss['Pays'],
                       color=colors, edgecolor='black')
       )


plt.xticks(rotation = 45, ha = 'right')  # Rotation des noms des pays
plt.xlabel('Pays', fontsize=15)
plt.ylabel('Disponibilité en kcal/personne/jour', fontsize=15)

plt.show
```

[96]: <function matplotlib.pyplot.show(close=None, block=None)>



3.11 - Exemple de la Thaïlande pour le Manioc

0.0.1 Création d'un dataframe avec uniquement la Thaïlande

[99]:
```
# Recherche de l'orthographe exact de Thaïlande

sous_nutrition[sous_nutrition['Zone'].str.contains('Thaï', na=False)].head(3)
```

[99]:

	Zone	Année	Sous_nutrition
1110	Thaïlande	2012-2014	6200000.0
1111	Thaïlande	2013-2015	6000000.0
1112	Thaïlande	2014-2016	5900000.0

[100]:
```
# Création du dataframe
sous_nutrition_thai=sous_nutrition.loc[sous_nutrition['Zone'] == 'Thaïlande', :]
sous_nutrition_thai.head(3)
```

[100]:

	Zone	Année	Sous_nutrition
1110	Thaïlande	2012-2014	6200000.0
1111	Thaïlande	2013-2015	6000000.0
1112	Thaïlande	2014-2016	5900000.0

0.0.2 Calcul de la sous nutrition en Thaïlande

[102]:
```
# Vérification de l'orthographe exacte de Thaïlande dans le DF population
population[population['Zone'].str.contains('Thaï', na=False)]
```

[102]:

	Zone	Année	Population
1308	Thaïlande	2013	68144518.0
1309	Thaïlande	2014	68438746.0
1310	Thaïlande	2015	68714511.0
1311	Thaïlande	2016	68971308.0
1312	Thaïlande	2017	69209810.0
1313	Thaïlande	2018	69428453.0

[103]:
```
# Recherche des valeurs uniques de la colonne Année de la table␣
↪sous_nutrition_thai
sous_nutrition_thai['Année'].unique().tolist()
```

[103]:
```
['2012-2014', '2013-2015', '2014-2016', '2015-2017', '2016-2018', '2017-2019']
```

[104]:
```
# Remplacement des Années par leurs valeurs médianes pour pouvoir joindre les␣
↪deux tables avec les années
sous_nutrition_thai.loc[sous_nutrition_thai['Année'] == '2012-2014', 'Année'] =␣
↪2013
sous_nutrition_thai.loc[sous_nutrition_thai['Année'] == '2013-2015', 'Année'] =␣
↪2014
sous_nutrition_thai.loc[sous_nutrition_thai['Année'] == '2014-2016', 'Année'] =␣
↪2015
```

```
sous_nutrition_thai.loc[sous_nutrition_thai['Année'] == '2015-2017', 'Année'] =␣
 ↪2016
sous_nutrition_thai.loc[sous_nutrition_thai['Année'] == '2016-2018', 'Année'] =␣
 ↪2017
sous_nutrition_thai.loc[sous_nutrition_thai['Année'] == '2017-2019', 'Année'] =␣
 ↪2018
sous_nutrition_thai['Année'].unique().tolist()
```

[104]: [2013, 2014, 2015, 2016, 2017, 2018]

[316]:
```
# Jointure interne des tables sous_nutrition_thai et population pour ne␣
 ↪conserver que les lignes qui correspondent
# à la Thaïlande

sous_nutrition_population_thai=pd.merge(sous_nutrition_thai, population,␣
 ↪on=['Zone', 'Année'], how='inner')
sous_nutrition_population_thai
```

[316]:

	Zone	Année	Sous_nutrition	Population
0	Thaïlande	2013	6200000.0	68144518.0
1	Thaïlande	2014	6000000.0	68438746.0
2	Thaïlande	2015	5900000.0	68714511.0
3	Thaïlande	2016	6000000.0	68971308.0
4	Thaïlande	2017	6200000.0	69209810.0
5	Thaïlande	2018	6500000.0	69428453.0

[324]:
```
# Détermination de la sous nutrition  et de la poopulation moyenne entre 2013␣
 ↪et 2018

sous_nutrition_thai_moyenne =␣
 ↪round(sous_nutrition_population_thai['Sous_nutrition'].mean()/1000000, )
population_thai_moyenne = round(sous_nutrition_population_thai['Population'].
 ↪mean()/1000000,)
proportion=round(sous_nutrition_population_thai['Sous_nutrition'].sum() /␣
 ↪sous_nutrition_population_thai['Population'].sum() * 100, 2)
print (f'Entre 2013 et 2018, on estime qu\'en moyenne␣
 ↪{sous_nutrition_thai_moyenne} millions de Thaïlandais étaient en état de␣
 ↪sous nutrition')
print (f'pour une population moyenne de {population_thai_moyenne} millions,␣
 ↪soit {proportion}%. ')
```

```
Entre 2013 et 2018, on estime qu'en moyenne 6 millions de Thaïlandais étaient en
état de sous nutrition
pour une population moyenne de 69 millions, soit 8.91%.
```

[107]:
```
# Création d'une colonne proportion
```

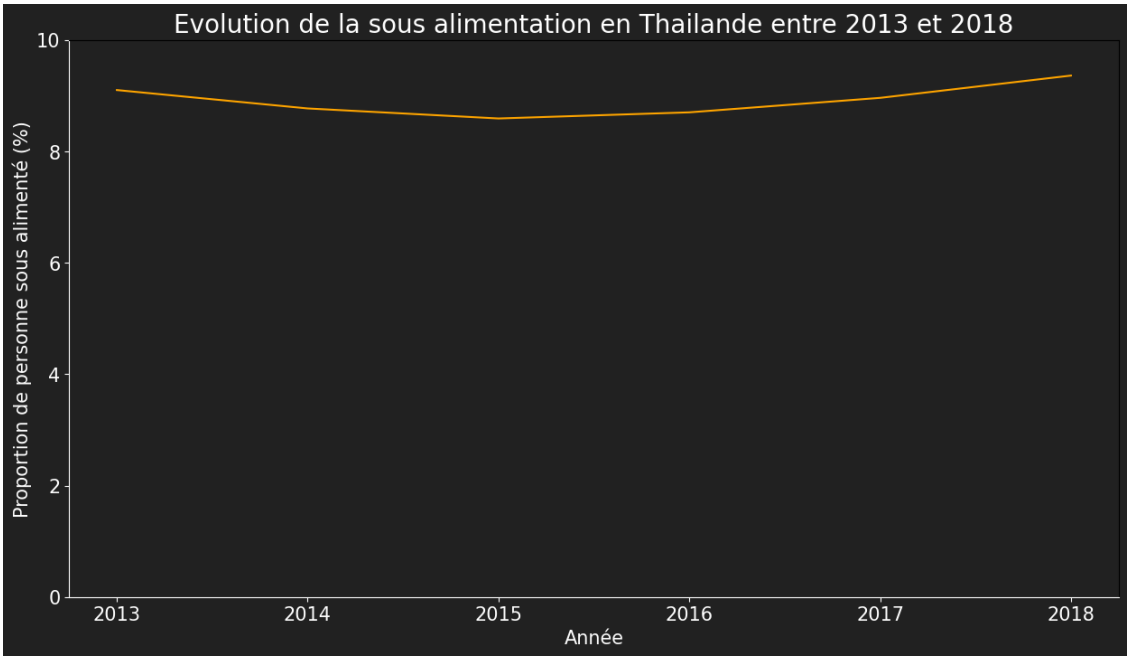
```
sous_nutrition_population_thai['Proportion␣
 ↪%']=(round(sous_nutrition_population_thai['Sous_nutrition']
                                                    /␣
 ↪sous_nutrition_population_thai['Population']*100, 2)
                                                    )
sous_nutrition_population_thai
```

[107]:

	Zone	Année	Sous_nutrition	Population	Proportion %
0	Thaïlande	2013	6200000.0	68144518.0	9.10
1	Thaïlande	2014	6000000.0	68438746.0	8.77
2	Thaïlande	2015	5900000.0	68714511.0	8.59
3	Thaïlande	2016	6000000.0	68971308.0	8.70
4	Thaïlande	2017	6200000.0	69209810.0	8.96
5	Thaïlande	2018	6500000.0	69428453.0	9.36

[108]:
```
# Détermination de l'évolution de la sous nutrition en Thaïlande
fig, ax = plt.subplots(figsize=(15,8))


# Persoonnalisation du graphe
fig.patch.set_facecolor('#212121')
ax.set_facecolor('#212121')
ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe des x
ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
ax.tick_params(axis='y', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des y
ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des x
ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des y
ax.xaxis.label.set_size(15)  # Modifie la taille du label de l'axe des x
ax.yaxis.label.set_size(15)  # Modifie la taille de l'axe des y
ax.set_ylim(0,10)  # Forcer l'axe des Y à commencer à 0


# Afficher le nom des axes, la légende et le titre
plt.xlabel('Année')
plt.ylabel('Proportion de personne sous alimenté (%)')
plt.title('Evolution de la sous alimentation en Thailande entre 2013 et 2018',␣
 ↪color='white', fontsize=20)

plt.plot(sous_nutrition_population_thai['Année'],␣
 ↪sous_nutrition_population_thai['Proportion %'], color='orange')
plt.show
```

[108]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



0.0.3 On calcule la proportion exportée en fonction de la proportion

Je pense qu'il manque des mots à cette phrase....
Je vais calculer la proportion de manioc qu'exporte la Thaïlande par rapport à sa production

[326]:
```
# Création d'une table ne contenant que les lignes correspondantes à la␣
 ↪Thaïlande et au manioc
dispo_alimentaire_thai_manioc=(dispo_alimentaire.
 ↪loc[(dispo_alimentaire['Produit'] == 'Manioc') &
                                (dispo_alimentaire['Zone'] == 'Thaïlande'), :].
 ↪copy()
                               )
dispo_alimentaire_thai_manioc
```

[326]:
```
            Zone Produit   Origine  Aliments pour animaux  \
13809  Thaïlande  Manioc  vegetale                 1800.0

       Autres Utilisations  Disponibilité alimentaire (Kcal/personne/jour)  \
13809               2081.0                                            40.0

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

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

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

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

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

       Semences  Traitement  Variation de stock
13809       0.0         0.0                 0.0
```

[328]:
```
# Calcul est affichage de la proportion de manioc exporté par la Thaïlande
dispo_alimentaire_thai_manioc.loc[:, 'Proportion␣
 ↪%']=(round(dispo_alimentaire_thai_manioc['Exportations - Quantité'] /
                                                                    ␣
 ↪(dispo_alimentaire_thai_manioc['Production'] -
                                                                    ␣
 ↪dispo_alimentaire_thai_manioc['Pertes']) * 100, 2)
                                                                   )
liste=['Zone', 'Produit', 'Exportations - Quantité',  'Disponibilité␣
 ↪intérieure', 'Importations - Quantité', 'Production', 'Pertes', 'Proportion␣
 ↪%']
dispo_alimentaire_thai_manioc[liste]
```

[328]:
```
            Zone Produit  Exportations - Quantité  Disponibilité intérieure  \
13809  Thaïlande  Manioc              2.521400e+10              6.264000e+09

       Importations - Quantité    Production        Pertes  Proportion %
13809              1.250000e+09  3.022800e+10  1.511000e+09          87.8
```

[112]:
```
# Autre façon
# .iloc[0] permet de retourner la première valeur de la colonne. Sans lui la␣
 ↪valeur retournée est le couple index - valeur

production_manioc = (round((dispo_alimentaire_thai_manioc.loc[:, 'Production'].
 ↪iloc[0] -
                            dispo_alimentaire_thai_manioc.loc[:, 'Pertes'].
 ↪iloc[0]) / 1000000, )
                     )
export_manioc = round(dispo_alimentaire_thai_manioc.loc[:, 'Exportations -␣
 ↪Quantité'].iloc[0] / 1000000, )
proportion = round(export_manioc / production_manioc * 100, )
importation_manioc = round(dispo_alimentaire_thai_manioc.loc[:, 'Importations -␣
 ↪Quantité'].iloc[0] / 1000000, )
importation_dispo_interieure = (round(dispo_alimentaire_thai_manioc.loc[:,␣
 ↪'Importations - Quantité'].iloc[0] /
```

```
                                        dispo_alimentaire_thai_manioc.loc[:,␣
 ↪'Disponibilité intérieure'].iloc[0] * 100, )
                                    )

print(f'En 2017, la Thaïlande a exporté {export_manioc} milliers de tonnes de␣
 ↪manioc pour une production totale (sans les pertes)')
print(f'de {production_manioc} milliers de tonnes, soit {proportion}%. Cette␣
 ↪même année, elle en a importé {importation_manioc}')
print(f'milliers de tonnes, soit {importation_dispo_interieure}% de sa␣
 ↪disponibilité intérieure')
```

```
En 2017, la Thaïlande a exporté 25214 milliers de tonnes de manioc pour une
production totale (sans les pertes)
de 28717 milliers de tonnes, soit 88%. Cette même année, elle en a importé 1250
milliers de tonnes, soit 20% de sa disponibilité intérieure
```

Etape 6 - Analyse complémentaires

[114]:
```
#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"
```

Pourquoi Haïti, la République populaire démocratique de Corée et Madagascar sont les pays qui

présentent la plus forte proportion de personnes en état de sous nutrition?

[116]:
```
# Disponibilité alimentaire d'Haïti, de la Corée et de Madagascar
top_3=['Haïti', 'République populaire démocratique de Corée', 'Madagascar']
dispo_alimentaire_kcal.loc[dispo_alimentaire_kcal['Zone'].isin(top_3), :]
```

[116]:

	Zone	Disponibilité alimentaire (Kcal/personne/jour)
65	Haïti	2089.0
91	Madagascar	2056.0
133	République populaire démocratique de Corée	2093.0

Nombre de calories disponibles par habitant pour ces 3 pays

[118]:
```
# Affichage de la table dispo_alimentaire_population
dispo_alimentaire_population.head(3)
```

[118]:
```
          Zone                Produit   Origine  Aliments pour animaux  \
0  Afghanistan       Abats Comestible   animale                    0.0
1  Afghanistan        Agrumes, Autres  vegetale                    0.0
2  Afghanistan  Aliments pour enfants  vegetale                    0.0

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

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

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

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

   Disponibilité intérieure  …  Importations - Quantité  Nourriture  \
0               53000000.0  …                     0.0  53000000.0
1               41000000.0  …                40000000.0  39000000.0
2                2000000.0  …                 2000000.0   2000000.0

      Pertes  Production  Semences  Traitement  Variation de stock   Année  \
0        0.0  53000000.0       0.0         0.0                 0.0  2017.0
1  2000000.0   3000000.0       0.0         0.0                 0.0  2017.0
2        0.0         0.0       0.0         0.0                 0.0  2017.0

   Population    dispo_kcal
0  36296113.0  6.624041e+10
1  36296113.0  1.324808e+10
2  36296113.0  1.324808e+10

[3 rows x 21 columns]
```

[119]:
```
# Calcul de la disponibilité totale par pays
dispo_alimentaire_kcal_pays=dispo_alimentaire_population.groupby('Zone').
  ↪agg({'Disponibilité alimentaire (Kcal/personne/jour)':'sum'}).reset_index()
dispo_alimentaire_kcal_pays.head(3)
```

[119]:
```
              Zone  Disponibilité alimentaire (Kcal/personne/jour)
0      Afghanistan                                          2087.0
1   Afrique du Sud                                          3020.0
2          Albanie                                          3188.0
```

[120]:
```
# Cas d'Haïti, de la Corée et de Madagascar
dispo_alimentaire_kcal_pays.loc[dispo_alimentaire_kcal_pays['Zone'].
 ↪isin(top_3), :]
```

[120]:
```
                                          Zone  \
65                                       Haïti
91                                  Madagascar
133  République populaire démocratique de Corée

     Disponibilité alimentaire (Kcal/personne/jour)
65                                           2089.0
91                                           2056.0
133                                          2093.0
```

[121]:
```
# Liste des pays ayant moins de 2200 kcal/personne/jour
dispo_alimentaire_kcal_pays.loc[dispo_alimentaire_kcal_pays['Disponibilité␣
 ↪alimentaire (Kcal/personne/jour)'] < 2200, :]['Zone'].tolist()
```

[121]:
```
['Afghanistan',
 'Haïti',
 'Madagascar',
 'Namibie',
 'Ouganda',
 'République centrafricaine',
 'République populaire démocratique de Corée',
 'Tadjikistan',
 'Tchad',
 'Timor-Leste',
 'Zambie',
 'Zimbabwe',
 'Éthiopie']
```

0.0.4 Proportion de l'alimentation d'origine animale

Travail basé sur la production - les pertes

[123]:
```
dispo_alimentaire.head(3)
```

[123]:
```
          Zone                Produit   Origine  Aliments pour animaux  \
0  Afghanistan       Abats Comestible   animale                    0.0
1  Afghanistan        Agrumes, Autres  vegetale                    0.0
2  Afghanistan  Aliments pour enfants  vegetale                    0.0
```

```
   Autres Utilisations  Disponibilité alimentaire (Kcal/personne/jour)  \
0                  0.0                                             5.0
1                  0.0                                             1.0
2                  0.0                                             1.0

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

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

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

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

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

[124]:
```
# Part de l'alimentation d'origine animale

# Copie de la table dispo_alimentaire
dispo_alimentaire_production_reelle = (dispo_alimentaire).copy()
# Création d'une colonne production réelle
dispo_alimentaire_production_reelle['Production réelle'] =␣
 ↪dispo_alimentaire['Production'] - dispo_alimentaire['Pertes']

# Aggrégation en fonction de l'origine
dispo_alimentaire_origine = (dispo_alimentaire_production_reelle.
 ↪groupby('Origine').agg({'Aliments pour animaux': 'sum',
                                                                 ␣
 ↪'Disponibilité alimentaire (Kcal/personne/jour)': 'sum',
                                                                  'Pertes':␣
 ↪'sum',
                                                                 ␣
 ↪'Production': 'sum',
```

```
 ↪'Production réelle':'sum'})
                            ).reset_index()

dispo_alimentaire_origine.head()
```

[124]:
```
    Origine  Aliments pour animaux  \
0   animale               107626.0
1  vegetale              1196619.0

   Disponibilité alimentaire (Kcal/personne/jour)        Pertes    Production  \
0                                         96660.0  2.393000e+10  1.372054e+12
1                                        398782.0  4.297680e+11  8.637626e+12

   Production réelle
0       1.348124e+12
1       8.207858e+12
```

[125]:
```
# Proportion d'alimentation d'origine animale
animale = dispo_alimentaire_origine.loc[dispo_alimentaire_origine['Origine'] ==
 ↪'animale', 'Production réelle'].sum()
vegetale = dispo_alimentaire_origine.loc[dispo_alimentaire_origine['Origine']
 ↪== 'vegetale', 'Production réelle'].sum()
proportion_animale = round(animale / (animale + vegetale) * 100, 2)
pertes_animale = dispo_alimentaire_origine.
 ↪loc[dispo_alimentaire_origine['Origine'] == 'animale', 'Pertes'].sum()
production_animale = dispo_alimentaire_origine.
 ↪loc[dispo_alimentaire_origine['Origine'] == 'animale', 'Production'].sum()
proportion_pertes_animale = round(pertes_animale / production_animale * 100, 2)
kcal_animale = (dispo_alimentaire_origine.
 ↪loc[dispo_alimentaire_origine['Origine'] == 'animale',
                'Disponibilité alimentaire (Kcal/personne/jour)'].sum()
               )
kcal_vegetale = (dispo_alimentaire_origine.
 ↪loc[dispo_alimentaire_origine['Origine'] == 'vegetale',
                 'Disponibilité alimentaire (Kcal/personne/jour)'].sum()
                )
proportion_kcal_animale = round(kcal_animale / (kcal_animale + kcal_vegetale) *
 ↪100, 2)



print(f'''la production d\'alimentation d\'origine animale représente
 ↪{proportion_animale} % de la production mondiale (animale et végétale)
({animale/1000000} milliards de tonnes).
```

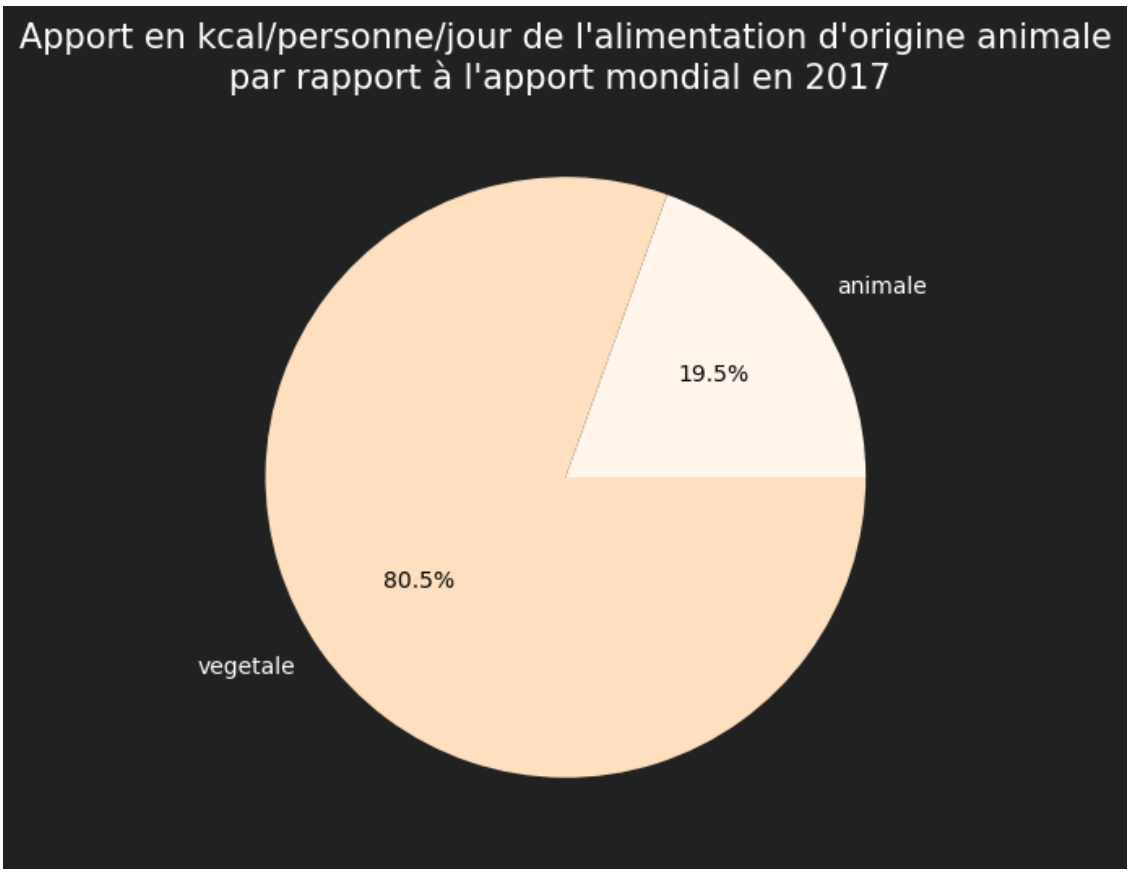
```
Les pertes de la production d'alimentation d'origine animale représentent␣
 ↪{proportion_pertes_animale} % de la production mondiale
({pertes_animale / 1000000} milliards de tonnes).
L'alimentation animale reprétente {proportion_kcal_animale} % de l'apport␣
 ↪mondial en kcal/personne/jour.
''')
```

```
la production d'alimentation d'origine animale représente 14.11 % de la
production mondiale (animale et végétale)
(1348124.0 milliards de tonnes).
Les pertes de la production d'alimentation d'origine animale représentent 1.74 %
de la production mondiale
(23930.0 milliards de tonnes).
L'alimentation animale reprétente 19.51 % de l'apport mondial en
kcal/personne/jour.
```

[126]:
```
# Création d'un graphe

valeurs_animale = dispo_alimentaire_origine['Disponibilité alimentaire (Kcal/
 ↪personne/jour)'].values
labels_animale = dispo_alimentaire_origine['Origine'].values

# Création de la figure et de l'axe
fig, ax = plt.subplots(figsize=(6,6))

# Modification de la couleur de fond
fig.patch.set_facecolor('#212121')

# Personnalisation des couleurs du graphe
cmap = plt.get_cmap('Oranges') # Récupère la palette viridis srockée dans cmap
colors = [cmap(i / len(valeurs_cereales)) for i in range(len(valeurs_cereales))]
# colors = [cmap(i / 3) for i in range(3)] si l'on ne souhaite générer que 3␣
 ↪couleurs
# cmap est un clormap qui génère une couleur en fonction d'un nombre compris␣
 ↪entre 0 et 1
# i / len(valeurs) convertit l'indice i en une valeur normalisée entre 0 et 1␣
 ↪pour une progression uniforme des couleurs
# cmap() retourne la couleur correspondante dans la palette choisie
# for i in range(len(sizes)) boucle sur chaque élément de sizes
# cmap(i / len(sizes)) retourne une couleur pour chaque valeur de i

# Création du graphique
wedges, texts, autotexts = ax.pie(x=valeurs_animale, labels=labels_animale,␣
 ↪autopct='%.1f%%', colors=colors)
# wedges contient les secteurs du piechart
# texts contient les labels des catégories
```

```
# autotexts contient les pourcentages

# Modification de la couleur des légendes
# On utilise une boucle parce que set_color ne peut s'appliquer à une lites, et␣
 ↪texs est une liste.
for text in texts:
    text.set_color('white')

plt.title(f'''Apport en kcal/personne/jour de l\'alimentation d\'origine animale
par rapport à l\'apport mondial en 2017 ''', color='white', fontsize=15)
plt.show
```

[126]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



0.0.5 Les plus gros producteurs d'aliments d'origine animale

[128]:
```
dispo_alimentaire.head(3)
```

[128]:

	Zone	Produit	Origine	Aliments pour animaux	\
0	Afghanistan	Abats Comestible	animale	0.0	

```
1  Afghanistan        Agrumes, Autres  vegetale                    0.0
2  Afghanistan  Aliments pour enfants  vegetale                    0.0

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

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

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

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

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

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

[129]:
```
# Cration de la table production par pays et origine
production_pays_origine = (dispo_alimentaire.groupby(['Zone', 'Origine']).
 ↪agg({'Production': 'sum',
                                                        'Pertes':
 ↪'sum',
                                                                 ␣
 ↪'Disponibilité alimentaire (Kcal/personne/jour)':'sum'})
                 ).reset_index()

# Expression des colonnes Production et Pertes en milliers de tonnes (elles␣
 ↪sont exprimées en kg)
production_pays_origine['Production'] = production_pays_origine['Production'] /␣
 ↪1000000
production_pays_origine['Pertes'] = production_pays_origine['Pertes'] / 1000000
```

```
# Création d'une colonne Production réelle
production_pays_origine['Production réelle'] =␣
 ↪(production_pays_origine['Production'].values -
                                    production_pays_origine['Pertes'].values)

# Pour le graphe, renomination de Zone en Pays
production_pays_origine.rename(columns = {'Zone': 'Pays'}, inplace=True )

production_pays_origine.head(3)
```

[129]:
```
             Pays   Origine  Production  Pertes  \
0     Afghanistan   animale      2280.0    63.0
1     Afghanistan  vegetale      8891.0  1072.0
2  Afrique du Sud   animale      7614.0    83.0

   Disponibilité alimentaire (Kcal/personne/jour)  Production réelle
0                                           216.0             2217.0
1                                          1871.0             7819.0
2                                           487.0             7531.0
```

[130]:
```
# Pays les plus producteurs d'alimentation d'origine animale sans compter les␣
 ↪pertes
production_animale = production_pays_origine.
 ↪loc[production_pays_origine['Origine'] == 'animale', :]
production_animale = production_animale.sort_values('Production',␣
 ↪ascending=False)
production_animale.head(10)
```

[130]:
```
                      Pays  Origine  Production  Pertes  \
72     Chine, continentale  animale    230681.0  2848.0
136                   Inde  animale    158226.0  5262.0
342  États-Unis d'Amérique  animale    154474.0   225.0
46                  Brésil  animale     65202.0  1706.0
108   Fédération de Russie  animale     47749.0   217.0
232               Pakistan  animale     44550.0  3924.0
8                Allemagne  animale     43621.0    64.0
106                 France  animale     33525.0    75.0
318                Turquie  animale     23196.0  1226.0
222       Nouvelle-Zélande  animale     22234.0   197.0

     Disponibilité alimentaire (Kcal/personne/jour)  Production réelle
72                                            723.0          227833.0
136                                           235.0          152964.0
342                                           984.0          154249.0
46                                            827.0           63496.0
108                                           843.0           47532.0
```

```
232                                          531.0            40626.0
8                                           1042.0            43557.0
106                                         1183.0            33450.0
318                                          567.0            21970.0
222                                          985.0            22037.0
```

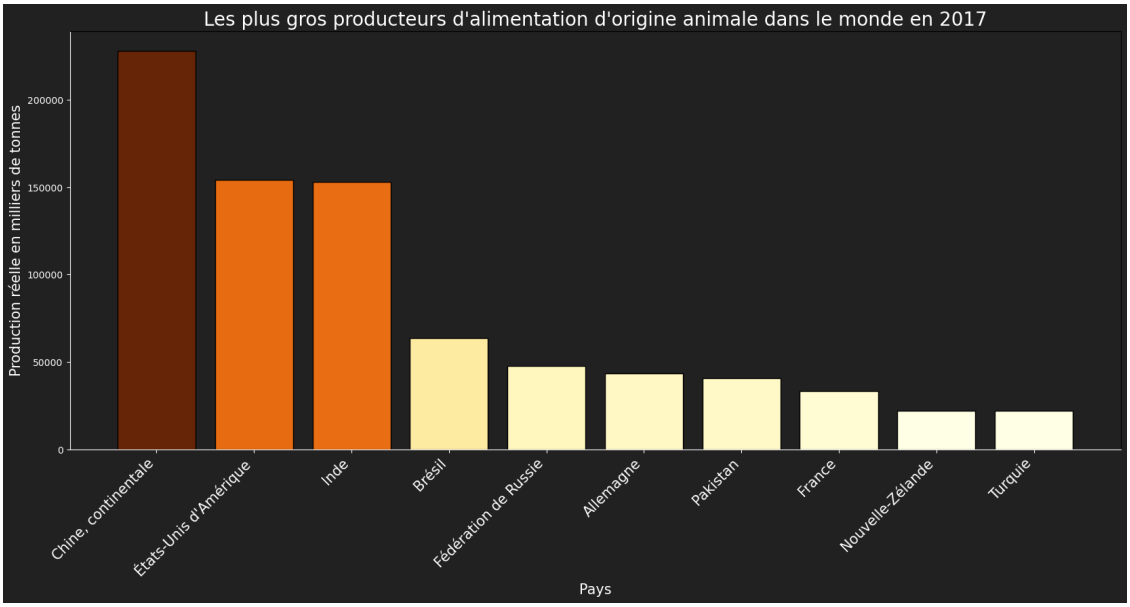
[131]:
```
# Pays les plus producteurs d'alimentation d'origine animale en comptant les pertes
production_animale_corrige = production_animale.sort_values('Production réelle', ascending=False).head(10)

production_animale_corrige.head(10)
```

[131]:
```
                      Pays  Origine  Production  Pertes  \
72     Chine, continentale  animale    230681.0  2848.0
342  États-Unis d'Amérique  animale    154474.0   225.0
136                   Inde  animale    158226.0  5262.0
46                  Brésil  animale     65202.0  1706.0
108   Fédération de Russie  animale     47749.0   217.0
8                 Allemagne  animale     43621.0    64.0
232               Pakistan  animale     44550.0  3924.0
106                 France  animale     33525.0    75.0
222       Nouvelle-Zélande  animale     22234.0   197.0
318                 Turquie  animale     23196.0  1226.0

     Disponibilité alimentaire (Kcal/personne/jour)  Production réelle
72                                            723.0           227833.0
342                                           984.0           154249.0
136                                           235.0           152964.0
46                                            827.0            63496.0
108                                           843.0            47532.0
8                                            1042.0            43557.0
232                                           531.0            40626.0
106                                          1183.0            33450.0
222                                           985.0            22037.0
318                                           567.0            21970.0
```

[132]:
```
# import de matplotlib.cm pour créer un dégradé de couleur
import matplotlib.cm as cm

# Appliquer un dégradé de couleur
norm = plt.Normalize(min(production_animale_corrige['Production réelle']),
                     max(production_animale_corrige['Production réelle']))

# Appliquer un dégradé aux barres
colors = cm.YlOrBr(norm(production_animale_corrige['Production réelle']))
```

```
# Appliquer un dégradé aux valeurs sur les barres
text_colors = cm.Greys_r(norm(production_animale_corrige['Production réelle']))

# fig et ax permettent de personnaliser plus facilement le graphe
fig, ax = plt.subplots(figsize=(20,8))

#  Personnalisation du graphique
fig.patch.set_facecolor('#212121')  # Modifie de la couleur autour du graphe
ax.set_facecolor('#212121')  # Modifie de la couleur du fond du graphe
ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe des x
ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
ax.tick_params(axis='y', colors='white')  # Modifie la couleur des valeurs de␣
 ↪l'axe des y
ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des x
ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des y
ax.set_title('Les plus gros producteurs d\'alimentation d\'origine animale dans␣
 ↪le monde en 2017',fontsize=20, color='white')


# Création d'un diagramme en barre
bars = (ax.bar(height = production_animale_corrige['Production réelle'],
                       x = production_animale_corrige['Pays'],
                      color=colors, edgecolor='black')
       )


plt.xticks(rotation = 45, ha = 'right')  # Rotation des noms des pays
plt.xlabel('Pays', fontsize=15)
plt.ylabel('Production réelle en milliers de tonnes', fontsize=15)

plt.show
```

[132]: <function matplotlib.pyplot.show(close=None, block=None)>



0.0.6 Les plus gros producteurs d'alimentation d'origine végétale

[134]:
```
# Pays les plus producteurs d'alimentation d'origine végétale en  comptant les pertes
production_vegetale_corrige = production_pays_origine.loc[production_pays_origine['Origine'] == 'vegetale', :]
production_vegetale_corrige = production_vegetale_corrige.sort_values('Production réelle', ascending=False).head(10)

production_vegetale_corrige.head(10)
```

[134]:

	Pays	Origine	Production	Pertes	\
73	Chine, continentale	vegetale	1700232.0	86727.0	
47	Brésil	vegetale	1078403.0	74208.0	
137	Inde	vegetale	968044.0	50668.0	
343	États-Unis d'Amérique	vegetale	740194.0	6937.0	
109	Fédération de Russie	vegetale	215547.0	4780.0	
139	Indonésie	vegetale	218177.0	12938.0	
307	Thaïlande	vegetale	193928.0	5635.0	
217	Nigéria	vegetale	175740.0	19792.0	
17	Argentine	vegetale	157381.0	3147.0	
107	France	vegetale	144611.0	3186.0	

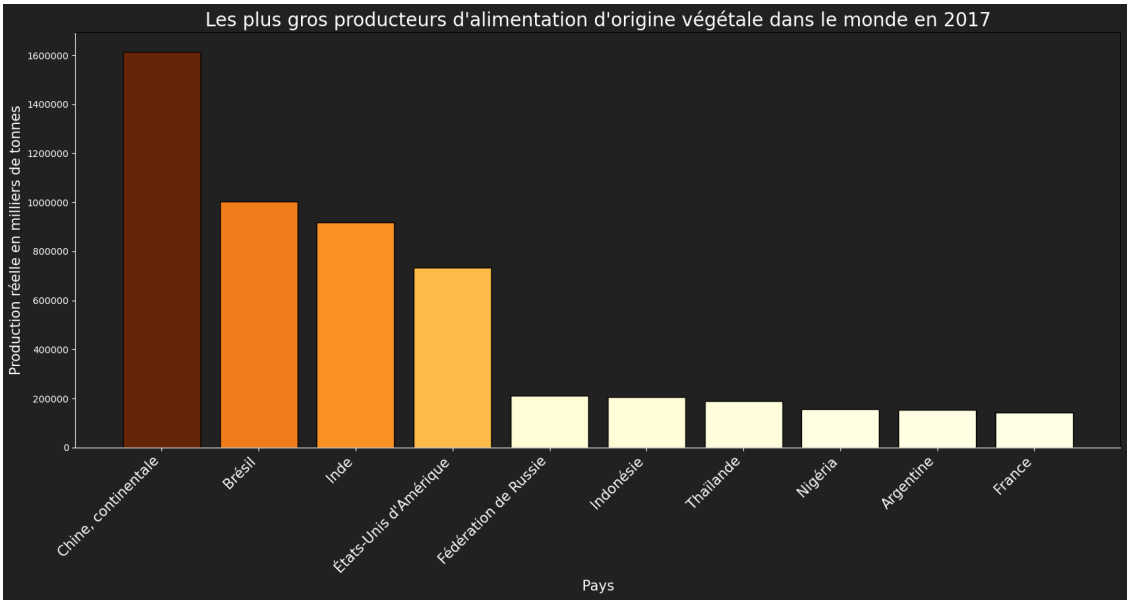
	Disponibilité alimentaire (Kcal/personne/jour)	Production réelle
73	2389.0	1613505.0
47	2435.0	1004195.0
137	2219.0	917376.0

```
343                                    2698.0          733257.0
109                                    2517.0          210767.0
139                                    2594.0          205239.0
307                                    2434.0          188293.0
217                                    2596.0          155948.0
17                                     2239.0          154234.0
107                                    2299.0          141425.0
```

[135]:
```
# import de matplotlib.cm pour créer un dégradé de couleur
import matplotlib.cm as cm

# Appliquer un dégradé de couleur
norm = plt.Normalize(min(production_vegetale_corrige['Production réelle']),
                     max(production_vegetale_corrige['Production réelle']))

# Appliquer un dégradé aux barres
colors = cm.YlOrBr(norm(production_vegetale_corrige['Production réelle']))

# Appliquer un dégradé aux valeurs sur les barres
text_colors = cm.Greys_r(norm(production_vegetale_corrige['Production réelle']))

# fig et ax permettent de personnaliser plus facilement le graphe
fig, ax = plt.subplots(figsize=(20,8))

#  Personnalisation du graphique
fig.patch.set_facecolor('#212121')  # Modifie de la couleur autour du graphe
ax.set_facecolor('#212121')  # Modifie de la couleur du fond du graphe
ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe des x
ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
ax.tick_params(axis='y', colors='white')  # Modifie la couleur des valeurs de␣
 ↪l'axe des y
ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des x
ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des y
ax.ticklabel_format(style='plain', axis='y')  # Empèche l'affichage␣
 ↪scientifique de l'axe des y
ax.set_title('Les plus gros producteurs d\'alimentation d\'origine végétale␣
 ↪dans le monde en 2017',fontsize=20, color='white')


# Création d'un diagramme en barre
bars = (ax.bar(height = production_vegetale_corrige['Production réelle'],
                        x = production_vegetale_corrige['Pays'],
                       color=colors, edgecolor='black')
       )
```

```
plt.xticks(rotation = 45, ha = 'right')  # Rotation des noms des pays
plt.xlabel('Pays', fontsize=15)
plt.ylabel('Production réelle en milliers de tonnes', fontsize=15)

plt.show
```

[135]: <function matplotlib.pyplot.show(close=None, block=None)>



0.0.7 Les plus gros producteurs d'alimentation

[137]:
```
# Aggrégation par pays
production_pays = production_pays_origine.drop(columns = ['Origine']).
 ↪groupby('Pays').sum().reset_index()
production_pays.head(3)
```

[137]:

	Pays	Production	Pertes	Disponibilité alimentaire (Kcal/personne/jour)	Production réelle
0	Afghanistan	11171.0	1135.0	2087.0	10036.0
1	Afrique du Sud	63263.0	2193.0	3020.0	61070.0
2	Albanie	3964.0	276.0	3188.0	3688.0

[138]:
```
# Les 10 plus gros producteurs
production_pays_trie = production_pays.sort_values('Production réelle',␣
↪ascending=False).head(10)
production_pays_trie
```

[138]:

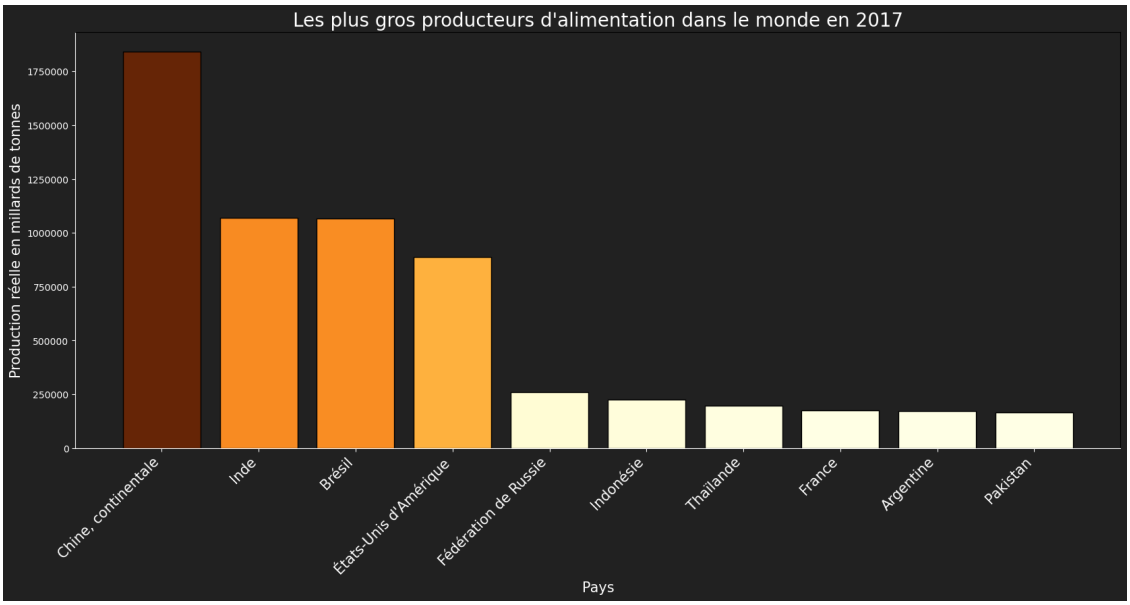
	Pays	Production	Pertes	\
36	Chine, continentale	1930913.0	89575.0	
68	Inde	1126270.0	55930.0	
23	Brésil	1143605.0	75914.0	
171	États-Unis d'Amérique	894668.0	7162.0	
54	Fédération de Russie	263296.0	4997.0	
69	Indonésie	238559.0	13081.0	
153	Thaïlande	201764.0	5749.0	
53	France	178136.0	3261.0	
8	Argentine	176124.0	3522.0	
116	Pakistan	172886.0	5897.0	

	Disponibilité alimentaire (Kcal/personne/jour)	Production réelle
36	3112.0	1841338.0
68	2454.0	1070340.0
23	3262.0	1067691.0
171	3682.0	887506.0
54	3360.0	258299.0
69	2776.0	225478.0
153	2785.0	196015.0
53	3482.0	174875.0
8	3226.0	172602.0
116	2438.0	166989.0

[139]:
```
# import de matplotlib.cm pour créer un dégradé de couleur
import matplotlib.cm as cm

# Appliquer un dégradé de couleur
norm = plt.Normalize(min(production_pays_trie['Production réelle']),
                     max(production_pays_trie['Production réelle']))

# Appliquer un dégradé aux barres
colors = cm.YlOrBr(norm(production_pays_trie['Production réelle']))

# Appliquer un dégradé aux valeurs sur les barres
text_colors = cm.Greys_r(norm(production_pays_trie['Production réelle']))

# fig et ax permettent de personnaliser plus facilement le graphe
fig, ax = plt.subplots(figsize=(20,8))

#  Personnalisation du graphique
fig.patch.set_facecolor('#212121')  # Modifie de la couleur autour du graphe
```

```
ax.set_facecolor('#212121')  # Modifie de la couleur du fond du graphe
ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe des x
ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
ax.tick_params(axis='y', colors='white')  # Modifie la couleur des valeurs de␣
 ↪l'axe des y
ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des x
ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des y
ax.ticklabel_format(style='plain', axis='y')  # Empêche l'affichage␣
 ↪scientifique de l'axe des y
ax.set_title('Les plus gros producteurs d\'alimentation dans le monde en␣
 ↪2017',fontsize=20, color='white')


# Création d'un diagramme en barre
bars = (ax.bar(height = production_pays_trie['Production réelle'],
                       x = production_pays_trie['Pays'],
                      color=colors, edgecolor='black')
       )


plt.xticks(rotation = 45, ha = 'right')  # Rotation des noms des pays
plt.xlabel('Pays', fontsize=15)
plt.ylabel('Production réelle en millards de tonnes', fontsize=15)

plt.show
```

[139]: <function matplotlib.pyplot.show(close=None, block=None)>



0.0.8 Evolution de l'aide et de le sous nutrition

[141]:
```
# Evolution de l'aide alimentaire au cours des années
aide_alimentaire_annee = aide_alimentaire.drop(columns = ['Zone', 'Produit']).
 ↪groupby('Année').sum().reset_index()

# Expression de l'aide_alimentaire en millions de tonnes
aide_alimentaire_annee['Aide_alimentaire'] /= 1000000

aide_alimentaire_annee
```

[141]:

	Année	Aide_alimentaire
0	2013	4165.674
1	2014	3939.152
2	2015	2187.507
3	2016	743.568

[142]:
```
# Evolution de la sous nutition au cours des années
# Remplacement des années par leur valeur médiane après copie de la table pour␣
 ↪conserver l'original
sous_nutrition_annee = (sous_nutrition).copy()
sous_nutrition_annee.loc[sous_nutrition_annee['Année'] == '2012-2014', 'Année']␣
 ↪= 2013
sous_nutrition_annee.loc[sous_nutrition_annee['Année'] == '2013-2015', 'Année']␣
 ↪= 2014
sous_nutrition_annee.loc[sous_nutrition_annee['Année'] == '2014-2016', 'Année']␣
 ↪= 2015
sous_nutrition_annee.loc[sous_nutrition_annee['Année'] == '2015-2017', 'Année']␣
 ↪= 2016
sous_nutrition_annee.loc[sous_nutrition_annee['Année'] == '2016-2018', 'Année']␣
 ↪= 2017
sous_nutrition_annee.loc[sous_nutrition_annee['Année'] == '2017-2019', 'Année']␣
 ↪= 2018

# Expressiosn de la population sous alimentée en millions
sous_nutrition_annee['Sous_nutrition'] /=1000000

# Regroupement par année
sous_nutrition_annee_groupe = sous_nutrition_annee.drop(columns = ['Zone']).
 ↪groupby('Année').sum().reset_index()

sous_nutrition_annee_groupe
```

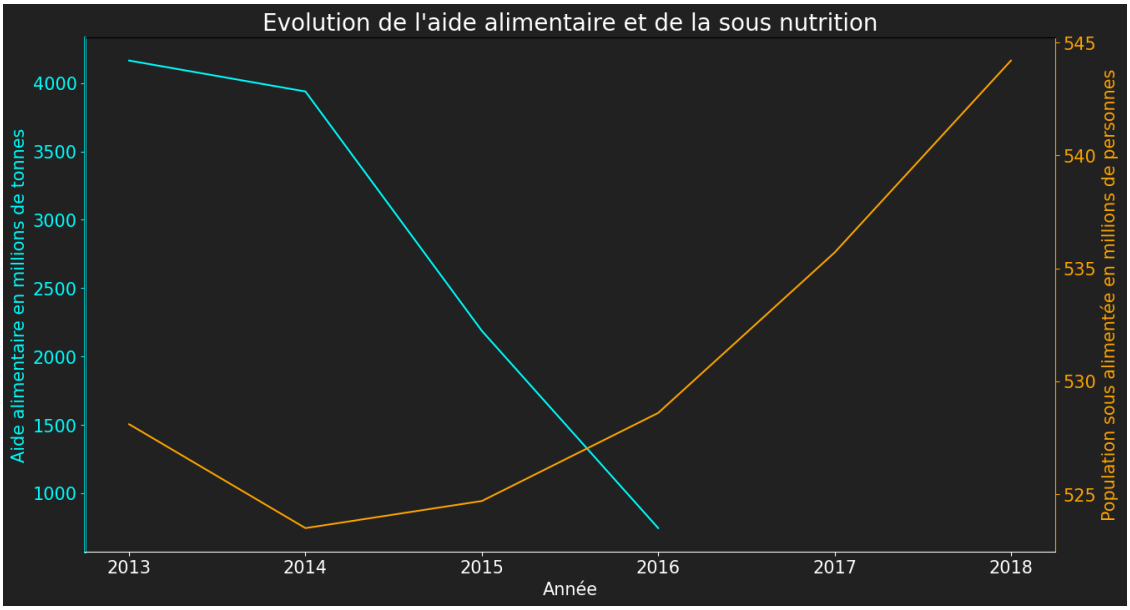
[142]:

	Année	Sous_nutrition
0	2013	528.1
1	2014	523.5
2	2015	524.7
3	2016	528.6
4	2017	535.7
5	2018	544.2

[143]:
```
# Création de courbes pour chaque pays

# Import de MaxNLocator pour le forçage des années en entiers
from matplotlib.ticker import MaxNLocator

fig, ax = plt.subplots(figsize=(15,8))

# Persoonnalisation du graphe
fig.patch.set_facecolor('#212121')
ax.set_facecolor('#212121')


# Création du graphique
ax.plot(aide_alimentaire_annee['Année'],␣
 ↪aide_alimentaire_annee['Aide_alimentaire'],
        color='cyan', label='Aide alimentire')
ax.set_ylabel('Aide alimentaire en millions de tonnes', color='cyan')
ax.set_xlabel('Année', size=15, color='white')
ax.spines['left'].set_color('cyan')  # Modifie la couleur de l'axe des y de␣
 ↪gauche
ax.spines['left'].set_linewidth(2)  # Rend plus visible l'axe qui ne␣
 ↪s'affichait pas
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
ax.tick_params(axis='y', labelsize=15, colors='cyan')  # Modifie la couleur des␣
 ↪valeurs de l'axe des y
ax.xaxis.label.set_size(15)  # Modifie la taille du label de l'axe des x
ax.yaxis.label.set_size(15)  # Modifie la taille de label de l'axe des y

# Création d'un deuxième axe des y
ax2 = ax.twinx()

# Personnalisation du deuxième axe
ax2.spines['right'].set_color('orange')
ax2.tick_params(axis='y', labelsize=15, colors='orange')
ax2.yaxis.label.set_color('orange')
ax2.yaxis.label.set_size(15)
ax2.spines['bottom'].set_color('white') # Tracé de l'axe des x. ax s'arrête à␣
 ↪2016, ax2 à 2018
```

```
# Tracé de la seconde courbe
ax2.plot(sous_nutrition_annee_groupe['Année'],␣
 ↪sous_nutrition_annee_groupe['Sous_nutrition'],
       color='orange', label='Sous nutrition')
ax2.set_ylabel('Population sous alimentée en millions de personnes')

plt.title('Evolution de l\'aide alimentaire et de la sous nutrition',␣
 ↪color='white', fontsize=20)

# Forcer l'affichage des années comme des entiers. Le premier afficahge affiche␣
 ↪les années comme des déciamux
#avec des moitiés d'années comme 2013,5
plt.gca().xaxis.set_major_locator(MaxNLocator(integer=True))

plt.show
```

[143]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



0.0.9 Le cas de la France

[145]:
```
# Evolution de la sous nutriion
sous_nutrition_france = sous_nutrition_annee.loc[sous_nutrition_annee['Zone']␣
 ↪== 'France', :]
sous_nutrition_france
```

[145]:
```
       Zone Année  Sous_nutrition
390  France  2013             0.0
391  France  2014             0.0
392  France  2015             0.0
393  France  2016             0.0
394  France  2017             0.0
395  France  2018             0.0
```

[146]:
```
# Pays qui ne présente pas de sous nutrition
sous_nutrition.loc[sous_nutrition['Sous_nutrition'] == 0, :]
```

[146]:
```
            Zone      Année  Sous_nutrition
24     Allemagne  2012-2014             0.0
25     Allemagne  2013-2015             0.0
26     Allemagne  2014-2016             0.0
27     Allemagne  2015-2017             0.0
28     Allemagne  2016-2018             0.0
…          …          …              …
1213    Zimbabwe  2013-2015             0.0
1214    Zimbabwe  2014-2016             0.0
1215    Zimbabwe  2015-2017             0.0
1216    Zimbabwe  2016-2018             0.0
1217    Zimbabwe  2017-2019             0.0

[714 rows x 3 columns]
```

[147]:
```
# Liste des pays qui n'ont jamais eu de personnes sous alimentés entre 2013 et␣
 ↪2018
sous_nutrition_groupe_pays = sous_nutrition_annee.drop(columns = ['Année']).
 ↪groupby('Zone').sum().reset_index()
pays_0 = sous_nutrition_groupe_pays.
 ↪loc[sous_nutrition_groupe_pays['Sous_nutrition'] == 0, :].shape[0]
print(f'{pays_0} pays ont déclaré aucune personne en état de sous nutrition␣
 ↪entre 2013 et 2018, dont la France')
```

```
116 pays ont déclaré aucune personne en état de sous nutrition entre 2013 et
2018, dont la France
```

[148]:
```
# Production et importation de la France en 2017
dispo_alimentaire_france = (dispo_alimentaire.loc[dispo_alimentaire['Zone'] ==␣
 ↪'France', :]
                            .groupby('Zone')
                            .agg({'Disponibilité alimentaire (Kcal/personne/
 ↪jour)':'sum',
                                  'Disponibilité intérieure':'sum',
                                  'Exportations - Quantité':'sum',
                                  'Importations - Quantité':'sum',
```

```
                                    'Nourriture':'sum',
                                   'Pertes':'sum',
                                   'Production':'sum'})
                               ).reset_index()

# Création de la colonne Priduction réelle
dispo_alimentaire_france['Production réelle'] =␣
 ↪dispo_alimentaire_france['Production'] - dispo_alimentaire_france['Pertes']
dispo_alimentaire_france
```

[148]:
```
     Zone  Disponibilité alimentaire (Kcal/personne/jour)  \
0  France                                          3482.0

   Disponibilité intérieure  Exportations - Quantité  Importations - Quantité  \
0              1.469050e+11             6.594500e+10             3.376900e+10

     Nourriture        Pertes    Production  Production réelle
0  6.027200e+10  3.261000e+09  1.781360e+11       1.748750e+11
```

[149]:
```
# Proportion d'importation, d'exportatin et de production par rappot à la␣
 ↪disponiblité intérieure

# Création d'une boucle pour le calcul
colonne_france = {'Importations - Quantité':'Les importations',
                  'Exportations - Quantité':'Les exportations',
                  'Production réelle':'La production'}

# Création de variables pour récupérer les valeurs de la boucle pour le graphe
valeurs_france = []
labels_france = []

for colonne in colonne_france:
    valeur = round((dispo_alimentaire_france[colonne] /␣
 ↪dispo_alimentaire_france['Disponibilité intérieure'] * 100).iloc[0], 2 )
    somme = dispo_alimentaire_france[colonne].sum()
    valeurs_france.append(valeur)
    labels_france.append(colonne)
    nom_affichage = colonne_france.get(colonne, colonne)
    print(f'{nom_affichage} représentent {valeur} % de la disponibilité␣
 ↪intérieure')

proportion_pertes = round((dispo_alimentaire_france['Pertes'] /␣
 ↪dispo_alimentaire_france['Production'] * 100).iloc[0], 2)
print(f'Les pertes représentent {proportion_pertes} % de la production ')
print(somme)
```

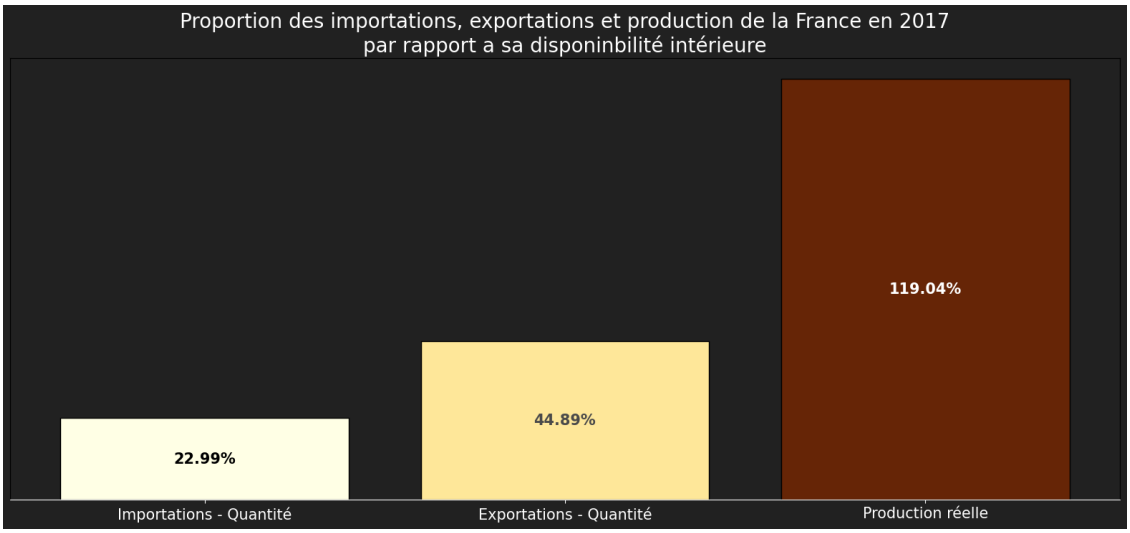
```
Les importations représentent 22.99 % de la disponibilité intérieure
```

```
Les exportations représentent 44.89 % de la disponibilité intérieure
La production représentent 119.04 % de la disponibilité intérieure
Les pertes représentent 1.83 % de la production
174875000000.0
```

[150]:
```python
# import de matplotlib.cm pour créer un dégradé de couleur
import matplotlib.cm as cm

# Appliquer un dégradé de couleur
norm = plt.Normalize(min(valeurs_france),
                     max(valeurs_france))

# Appliquer un dégradé aux barres
colors = cm.YlOrBr(norm(valeurs_france))

# Appliquer un dégradé aux valeurs sur les barres
text_colors = cm.Greys_r(norm(valeurs_france))

# fig et ax permettent de personnaliser plus facilement le graphe
fig, ax = plt.subplots(figsize=(20,8))

#  Personnalisation du graphique
fig.patch.set_facecolor('#212121')  # Modifie de la couleur autour du graphe
ax.set_facecolor('#212121')  # Modifie de la couleur du fond du graphe
ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe des x
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
ax.xaxis.label.set_color('white')# Modifie la couleur du label de l'axe des x
ax.yaxis.set_visible(False)
ax.set_title(f'''Proportion des importations, exportations et production de la␣
 ↪France en 2017
par rapport a sa disponinbilité intérieure''',fontsize=20, color='white')


# Création d'un diagramme en barre
bars = (ax.bar(height = valeurs_france,
                       x = labels_france,
                      color=colors, edgecolor='black')
       )

# inscrire les valeurs sur les barres
for bar, text_color in zip(bars, text_colors):
    yval = bar.get_height()  # Récupère la valeur de la barre
    plt.text(bar.get_x() + bar.get_width() / 2, yval / 2, f'{round(yval, 2)}%',
             ha = 'center', va = 'center', fontsize = 15, color = text_color,␣
 ↪fontweight='bold', rotation = 0)
```

```
plt.show
```

[150]: <function matplotlib.pyplot.show(close=None, block=None)>

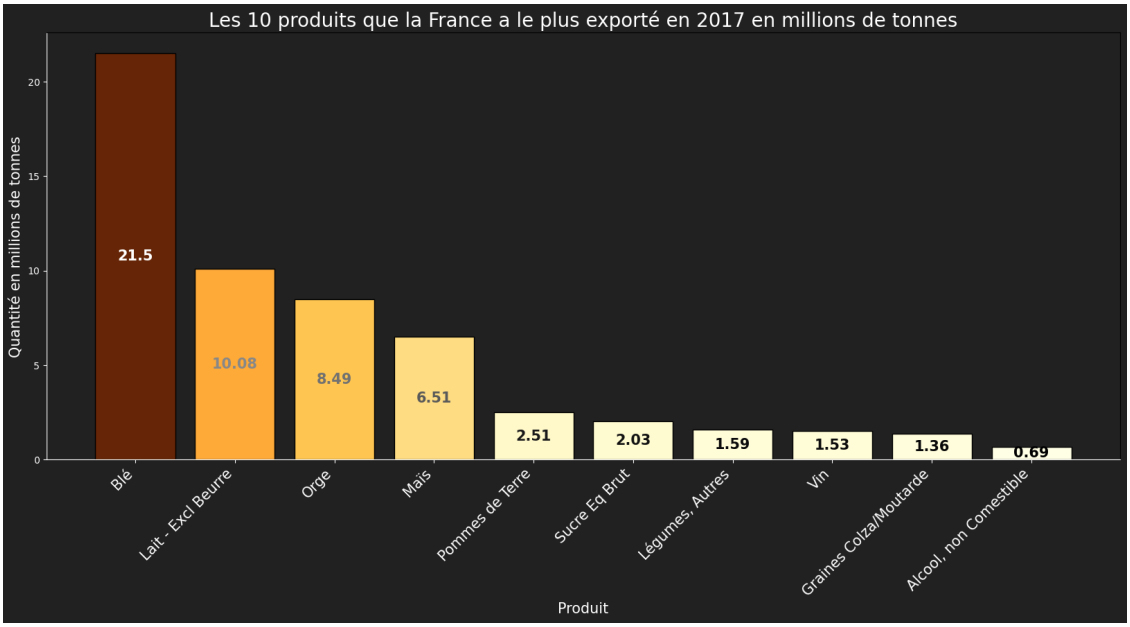


[151]:
```
# Produit principalement exporté par la france

dispo_alimentaire_france_detail = dispo_alimentaire.
 ↪loc[dispo_alimentaire['Zone'] == 'France',:].copy()

# Conversion des colonnes Importation et exportation en millions de tonnes␣
 ↪(elles sont en kg)
dispo_alimentaire_france_detail['Exportations - Quantité'] /= 1000000000
dispo_alimentaire_france_detail['Importations - Quantité'] /= 1000000000

# Tri par ordre décroissant
exportation_produit = (dispo_alimentaire_france_detail.
 ↪sort_values('Exportations - Quantité', ascending=False)
                       .head(10)['Produit'].values.tolist()
                      )
exportation_valeur = (dispo_alimentaire_france_detail.sort_values('Exportations␣
 ↪- Quantité', ascending=False)
                       .head(10)['Exportations - Quantité'].values.tolist()
                      )

# import de matplotlib.cm pour créer un dégradé de couleur
import matplotlib.cm as cm

# Appliquer un dégradé de couleur
norm = plt.Normalize(min(exportation_valeur),
```

```
                    max(exportation_valeur))  # Table et nom de la colonne pour␣
 ↪faire les barres.

# Appliquer un dégradé aux barres
colors = cm.YlOrBr(norm(exportation_valeur))

# Appliquer un dégradé aux valeurs sur les barres. _r pour inverser le dégradé.
text_colors = cm.Greys_r(norm(exportation_valeur))

# fig et ax permettent de personnaliser facilement le graphe
fig, ax = plt.subplots(figsize=(20,8))

# Personnalisation du graphe
#plt.gca().set_facecolor('black')  # Modifie la couleur du fond du graphique␣
 ↪quand on souhaite modifier ce fond uniquement
fig.patch.set_facecolor('#212121')  # Modifie de la couleur autour du graphe
ax.set_facecolor('#212121')  # Modifie de la couleur du fond du graphe
ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe des x
ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
ax.tick_params(axis='y', colors='white')  # Modifie la couleur des valeurs de␣
 ↪l'axe des y
ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des x
ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des y
#ax.title.set_color('White')  # Modifie la couleur du titre
ax.set_title('Les 10 produits que la France a le plus exporté en 2017 en␣
 ↪millions de tonnes',fontsize=20, color='white')


# Création d'un diagramme en barre
bars = ax.bar(height = exportation_valeur, x = exportation_produit,
              color=colors, edgecolor='black')

# inscrire les valeurs sur les barres
for bar, text_color in zip(bars, text_colors):
    yval = bar.get_height()  # Récupère la valeur de la barre
    plt.text(bar.get_x() + bar.get_width() / 2, yval / 2, round(yval,2),
             ha = 'center', va = 'center', fontsize = 15, color = text_color,␣
 ↪fontweight='bold', rotation = 0)

# bar.get_x() + bar.get_width() / 2  centre le texte sur la barre
# yval / 2 place le texte au milieur de la barre
# ha = 'center'  centre le texte horizontalement
# va = 'center' place le texte au centre de la barre
```

```
plt.xticks(rotation = 45, ha = 'right')  # Rotation des noms des pays
plt.xlabel('Produit', fontsize=15)
plt.ylabel('Quantité en millions de tonnes', fontsize=15)

plt.show
```

[151]: <function matplotlib.pyplot.show(close=None, block=None)>

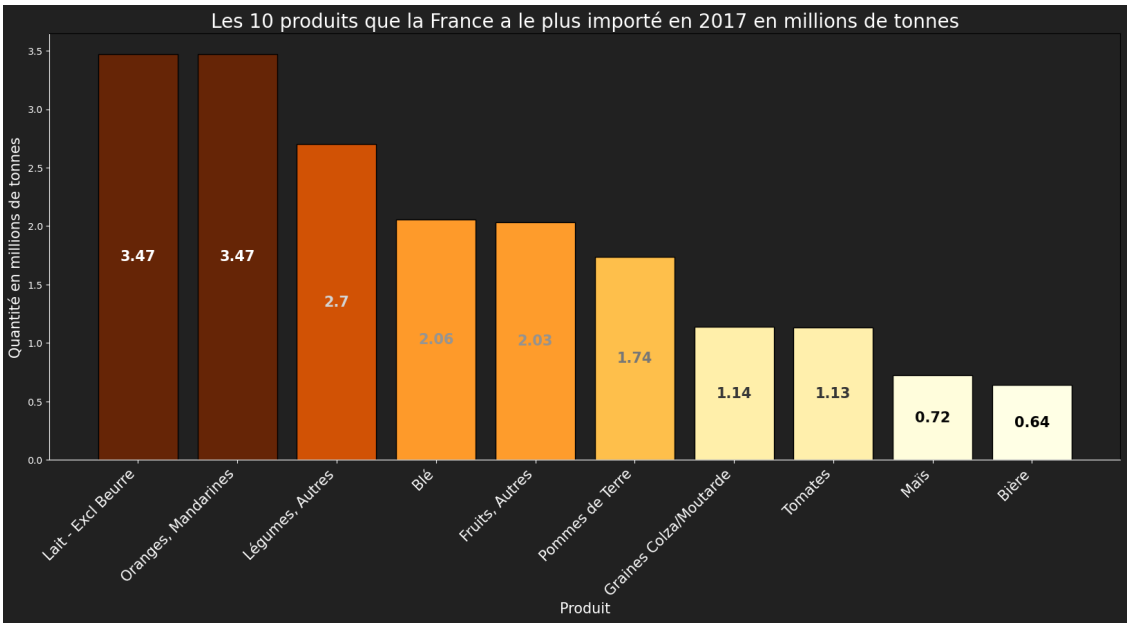


[152]:
```
# Produit principalement exporté par la france

importation_valeur = (dispo_alimentaire_france_detail.sort_values('Importations␣
 ↪- Quantité', ascending=False)
                .head(10)['Importations - Quantité' ].values.tolist()
              )
importation_produit = (dispo_alimentaire_france_detail.
 ↪sort_values('Importations - Quantité', ascending = False)
                       .head(10)['Produit'].values.tolist()
                      )

# import de matplotlib.cm pour créer un dégradé de couleur
import matplotlib.cm as cm

# Appliquer un dégradé de couleur
norm = plt.Normalize(min(importation_valeur),
                     max(importation_valeur))  # Table et nom de la colonne pour␣
 ↪faire les barres.
```

```
# Appliquer un dégradé aux barres
colors = cm.YlOrBr(norm(importation_valeur))

# Appliquer un dégradé aux valeurs sur les barres. _r pour inverser le dégradé.
text_colors = cm.Greys_r(norm(importation_valeur))

# fig et ax permettent de personnaliser facilement le graphe
fig, ax = plt.subplots(figsize=(20,8))

# Personnalisation du graphe
#plt.gca().set_facecolor('black')  # Modifie la couleur du fond du graphique␣
 ↪quand on souhaite modifier ce fond uniquement
fig.patch.set_facecolor('#212121')  # Modifie de la couleur autour du graphe
ax.set_facecolor('#212121')  # Modifie de la couleur du fond du graphe
ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe des x
ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
ax.tick_params(axis='y', colors='white')  # Modifie la couleur des valeurs de␣
 ↪l'axe des y
ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des x
ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des y
#ax.title.set_color('White')  # Modifie la couleur du titre
ax.set_title('Les 10 produits que la France a le plus importé en 2017 en␣
 ↪millions de tonnes',fontsize=20, color='white')


# Création d'un diagramme en barre
bars = ax.bar(height = importation_valeur, x = importation_produit,
              color=colors, edgecolor='black')

# inscrire les valeurs sur les barres
for bar, text_color in zip(bars, text_colors):
    yval = bar.get_height()  # Récupère la valeur de la barre
    plt.text(bar.get_x() + bar.get_width() / 2, yval / 2, round(yval,2),
             ha = 'center', va = 'center', fontsize = 15, color = text_color,␣
 ↪fontweight='bold', rotation = 0)

# bar.get_x() + bar.get_width() / 2  centre le texte sur la barre
# yval / 2 place le texte au milieur de la barre
# ha = 'center'  centre le texte horizontalement
# va = 'center' place le texte au centre de la barre

plt.xticks(rotation = 45, ha = 'right')  # Rotation des noms des pays
plt.xlabel('Produit', fontsize=15)
plt.ylabel('Quantité en millions de tonnes', fontsize=15)
```

```
plt.show
```

[152]: <function matplotlib.pyplot.show(close=None, block=None)>



[153]:
```
dispo_alimentaire_france_detail.head()
```

[153]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)
4839	France	Abats Comestible	animale	0.0	0.0	18.0	5.86
4840	France	Agrumes, Autres	vegetale	0.0	0.0	0.0	0.11
4841	France	Alcool, non Comestible	vegetale	0.0	56.0	0.0	0.00
4842	France	Aliments pour enfants	vegetale	0.0	0.0	0.0	0.00
4843	France	Ananas	vegetale	0.0	0.0	3.0	2.60

```
      Disponibilité de matière grasse en quantité (g/personne/jour)  \
4839                                                       0.51
4840                                                       0.00
4841                                                       0.00
4842                                                       0.00
4843                                                       0.01

      Disponibilité de protéines en quantité (g/personne/jour)  \
4839                                                       2.93
4840                                                       0.00
4841                                                       0.00
4842                                                       0.00
4843                                                       0.02

      Disponibilité intérieure  Exportations - Quantité  \
4839              377000000.0                     0.166
4840                8000000.0                     0.001
4841               56000000.0                     0.693
4842              -90000000.0                     0.133
4843              183000000.0                     0.030

      Importations - Quantité   Nourriture      Pertes   Production  Semences  \
4839                    0.090  377000000.0         0.0  453000000.0       0.0
4840                    0.009    7000000.0         0.0          0.0       0.0
4841                    0.174          0.0         0.0  575000000.0       0.0
4842                    0.044          0.0         0.0          0.0       0.0
4843                    0.213  167000000.0  16000000.0          0.0       0.0

      Traitement  Variation de stock
4839         0.0                 0.0
4840         0.0                 0.0
4841         0.0                 0.0
4842         0.0                 0.0
4843         0.0                 0.0
```

0.0.10 Relation entre sous nutrition et production

[155]:
```
sous_nutrition.head()
```

[155]:
```
          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
```

[156]:
```
production_pays.head()
```

[156]:
```
             Pays  Production  Pertes  \
0     Afghanistan     11171.0  1135.0
1  Afrique du Sud     63263.0  2193.0
2         Albanie      3964.0   276.0
3         Algérie     26359.0  3753.0
4       Allemagne    154547.0  3781.0

   Disponibilité alimentaire (Kcal/personne/jour)  Production réelle
0                                          2087.0            10036.0
1                                          3020.0            61070.0
2                                          3188.0             3688.0
3                                          3293.0            22606.0
4                                          3503.0           150766.0
```

[157]:
```
# Jointure des tables sous_nutrition et production_pays

sous_nutrition_production = (pd.merge(production_pays, sous_nutrition.
 ↪loc[sous_nutrition['Année'] == '2016-2018',
                                        ['Zone',␣
 ↪'Sous_nutrition']],left_on='Pays', right_on='Zone', how='left')
                             )
# Retrait des valeurs extrèmes constatées sur un nuage de points sous nutrition␣
 ↪- production réelle
sous_nutrition_production_corrige = (sous_nutrition_production.
 ↪loc[(sous_nutrition_production['Production réelle'] < 500000) &
                                     ␣
 ↪(sous_nutrition_production['Sous_nutrition'] < 50000000), :]
                                     ).copy()

# Expression de la colonne Sous nutrition en millions de personnes
sous_nutrition_production_corrige['Sous_nutrition']=sous_nutrition_production_corrige['Sous_nu
 ↪/ 1e6

sous_nutrition_production_corrige.head(3)
```

[157]:
```
             Pays  Production  Pertes  \
0     Afghanistan     11171.0  1135.0
1  Afrique du Sud     63263.0  2193.0
2         Albanie      3964.0   276.0

   Disponibilité alimentaire (Kcal/personne/jour)  Production réelle  \
0                                          2087.0            10036.0
1                                          3020.0            61070.0
2                                          3188.0             3688.0
```

```
             Zone  Sous_nutrition
0     Afghanistan            10.5
1  Afrique du Sud             3.1
2         Albanie             0.1
```

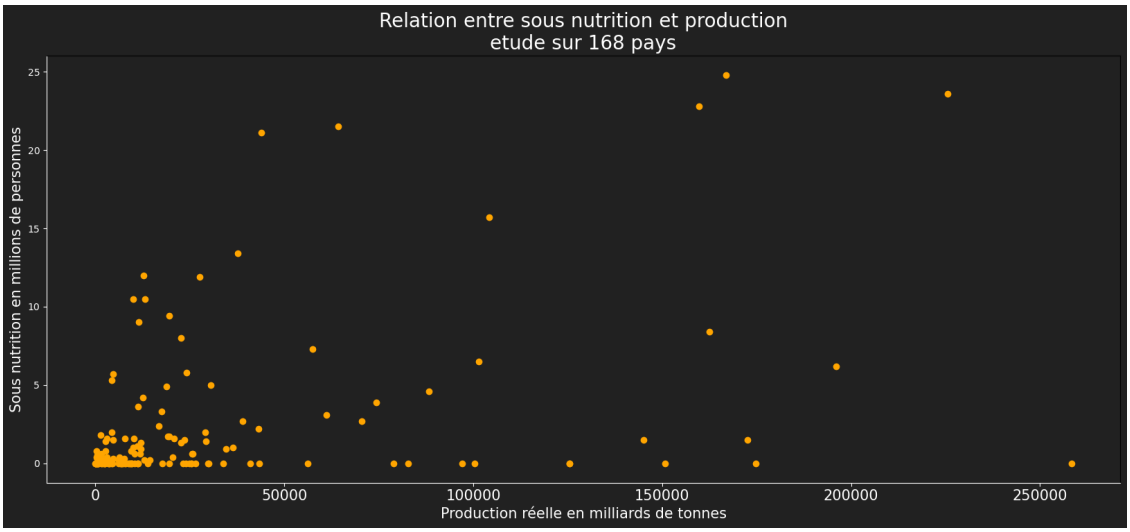
[158]:
```
sous_nutrition_production_corrige.describe()
```

[158]:
```
          Production        Pertes  \
count     168.000000    168.000000
mean    28725.642857   1331.559524
std     48689.890313   2498.910919
min         6.000000      0.000000
25%      1804.750000     84.000000
50%      9987.500000    435.000000
75%     27002.250000   1465.250000
max    263296.000000  19854.000000

       Disponibilité alimentaire (Kcal/personne/jour)  Production réelle  \
count                                      168.000000         168.000000
mean                                      2834.833333       27394.083333
std                                        435.356756       46816.583392
min                                       1879.000000           6.000000
25%                                       2523.250000        1655.500000
50%                                       2821.500000        9525.500000
75%                                       3172.250000       25545.250000
max                                       3770.000000      258299.000000

       Sous_nutrition
count      168.000000
mean         2.057143
std          4.622641
min          0.000000
25%          0.000000
50%          0.000000
75%          1.525000
max         24.800000
```

[159]:
```
# fig et ax permettent de personnaliser plus facilement le graphe
fig, ax = plt.subplots(figsize=(20,8))

#  Personnalisation du graphique
fig.patch.set_facecolor('#212121')  # Modifie de la couleur autour du graphe
ax.set_facecolor('#212121')  # Modifie de la couleur du fond du graphe

ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe des x
ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
```

```
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
ax.tick_params(axis='y', colors='white')  # Modifie la couleur des valeurs de␣
 ↪l'axe des y
ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des x
ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des y
ax.ticklabel_format(style='plain', axis='y')  # Empêche l'affichage␣
 ↪scientifique de l'axe des y
ax.set_title(f'''Relation entre sous nutrition et production
etude sur 168 pays''',fontsize=20, color='white')

ax.scatter(sous_nutrition_production_corrige['Production réelle'],␣
 ↪sous_nutrition_production_corrige['Sous_nutrition'], color='orange')

plt.xlabel('Production réelle en milliards de tonnes', fontsize=15)
plt.ylabel('Sous nutrition en millions de personnes', fontsize=15)

plt.show
```

[159]: <function matplotlib.pyplot.show(close=None, block=None)>



[160]:
```
# Relation entre sous nutrition et proportion de personnes sous alimentées

# Jointure de la table sous nutrition production corrige et population pour␣
 ↪l'année 2017
sous_nutrition_proportion=(pd.merge(sous_nutrition_production_corrige,␣
 ↪population.loc[population['Année'] == 2017, ['Zone', 'Population']],
                                    left_on='Pays', right_on='Zone', how='left')
                                   )
```

```
# Epression de la colonne Population en millions de personnes (exprimée en␣
 ↪nombre d'habitants)
sous_nutrition_proportion['Population']=sous_nutrition_proportion['Population']␣
 ↪/ 1000000


# Création d'une colonne Proportion de sous nutrition par rapport à la␣
 ↪population
sous_nutrition_proportion['Proportion %'] =␣
 ↪(round(sous_nutrition_proportion['Sous_nutrition']
                                                    /␣
 ↪sous_nutrition_proportion['Population'] * 100, 2)
                                                   )

sous_nutrition_proportion.head(3)
```

[160]:
```
             Pays  Production  Pertes  \
0     Afghanistan     11171.0  1135.0
1  Afrique du Sud     63263.0  2193.0
2         Albanie      3964.0   276.0

   Disponibilité alimentaire (Kcal/personne/jour)  Production réelle  \
0                                          2087.0            10036.0
1                                          3020.0            61070.0
2                                          3188.0             3688.0

           Zone_x  Sous_nutrition          Zone_y  Population  Proportion %
0     Afghanistan            10.5     Afghanistan   36.296113         28.93
1  Afrique du Sud             3.1  Afrique du Sud   57.009756          5.44
2         Albanie             0.1         Albanie    2.884169          3.47
```

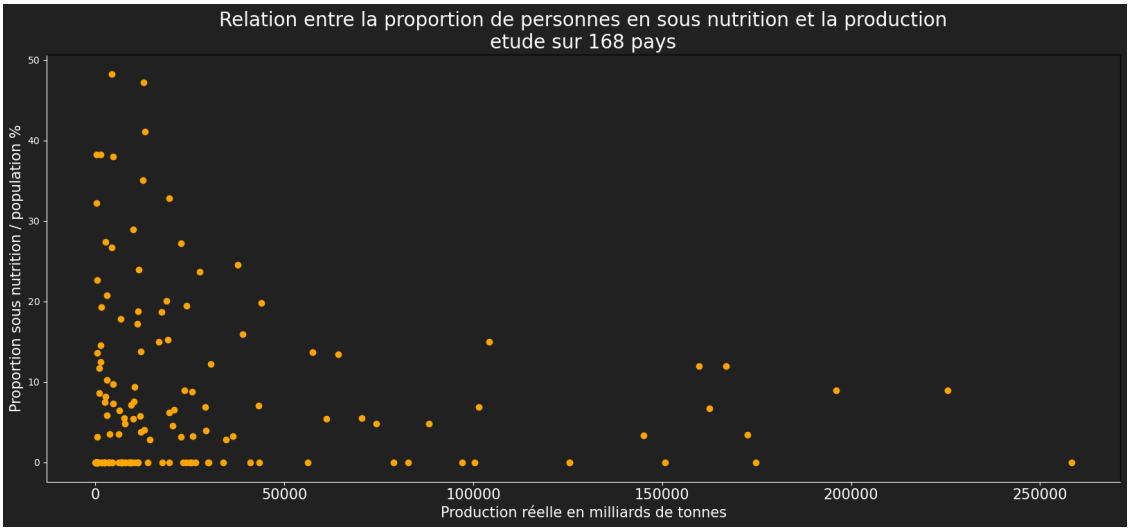
[161]:
```
sous_nutrition_proportion.describe()
```

[161]:
```
          Production        Pertes  \
count     168.000000    168.000000
mean    28725.642857   1331.559524
std     48689.890313   2498.910919
min         6.000000      0.000000
25%      1804.750000     84.000000
50%      9987.500000    435.000000
75%     27002.250000   1465.250000
max    263296.000000  19854.000000

       Disponibilité alimentaire (Kcal/personne/jour)  Production réelle  \
count                                      168.000000         168.000000
mean                                      2834.833333       27394.083333
```

```
std                                          435.356756       46816.583392
min                                         1879.000000           6.000000
25%                                         2523.250000        1655.500000
50%                                         2821.500000        9525.500000
75%                                         3172.250000       25545.250000
max                                         3770.000000      258299.000000

       Sous_nutrition  Population  Proportion %
count      168.000000  168.000000    168.000000
mean         2.057143   23.805260      6.978869
std          4.622641   39.605994     10.572215
min          0.000000    0.052045      0.000000
25%          0.000000    2.211687      0.000000
50%          0.000000    9.154640      0.000000
75%          1.525000   28.038369      9.475000
max         24.800000  264.650963     48.260000
```

[162]:
```
# fig et ax permettent de personnaliser plus facilement le graphe
fig, ax = plt.subplots(figsize=(20,8))

#  Personnalisation du graphique
fig.patch.set_facecolor('#212121')  # Modifie de la couleur autour du graphe
ax.set_facecolor('#212121')  # Modifie de la couleur du fond du graphe

ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe des x
ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la couleur␣
 ↪des valeurs de l'axe des x
ax.tick_params(axis='y', colors='white')  # Modifie la couleur des valeurs de␣
 ↪l'axe des y
ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des x
ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe des y
ax.ticklabel_format(style='plain', axis='y')  # Empêche l'affichage␣
 ↪scientifique de l'axe des y
ax.set_title(f'''Relation entre la proportion de personnes en sous nutrition et␣
 ↪la production
etude sur 168 pays''',fontsize=20, color='white')

ax.scatter(sous_nutrition_proportion['Production réelle'],␣
 ↪sous_nutrition_proportion['Proportion %'], color='orange')

plt.xlabel('Production réelle en milliards de tonnes', fontsize=15)
plt.ylabel('Proportion sous nutrition / population %', fontsize=15)

plt.show
```

[162]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



0.0.11 Pays ayant reçu le moins d'aide alimentaire

[164]:
```
# Pays ayant reçu le moins d'aide alimentaire entre 2013 et 2016
aide_alimentaire_10pays_moins = (aide_alimentaire_pays.
 ↪sort_values('Aide_alimentaire', ascending=True).head(10)['Zone'].tolist())

# Sélection des années entre 2013 et 2016 de la table sous nutrition
sous_nutrition_2016=sous_nutrition_annee.loc[sous_nutrition_annee['Année'].
 ↪between (2013, 2016), :]

# Aggrégation par Pays et Année
sous_nutrition_pays=sous_nutrition_2016.groupby(['Zone', 'Année']).sum().
 ↪reset_index()

# Création d'un colonne Proportion après jointure des tables␣
 ↪sous_nutrition_pays et population
# Réinitialisation des index parce que la jointure ne fonctionne pas.
sous_nutrition_pays=sous_nutrition_pays.reset_index(drop=True)
population=population.reset_index(drop=True)
sous_nutrition_proportion_pays=pd.merge(sous_nutrition_pays, population,␣
 ↪on=['Zone', 'Année'], how='left')
sous_nutrition_proportion_pays['Sous_nutrition']=sous_nutrition_proportion_pays['Sous_nutritio
 ↪* 1000000 # Expression en habitants
sous_nutrition_proportion_pays['Proportion']=(round(sous_nutrition_proportion_pays['Sous_nutri
                                                    /␣
 ↪sous_nutrition_proportion_pays['Population'] * 100, 2)
                                                   )
```

```
# Pays affichant la plus forte proportion de personne sous nutries entre 2013␣
 ↪et 2016
sous_nutrition_pays_plus=sous_nutrition_proportion_pays.
 ↪sort_values('Proportion', ascending=False).head(10)['Zone'].unique().tolist()

# Détermination des éléments en commun
communs = set(aide_alimentaire_10pays_moins) & set(sous_nutrition_pays_plus)  #␣
 ↪Retourne l'intersection des ensembles
if communs:
    print(f'''Ces pays sont les pays qui ont le moins reçu d\'aide alimentaire␣
 ↪et qui présentent la plus forte proportion de
    personnes en état de sous nutrtion {communs} ''')
else:
    print('Les pays qui pésentent le plus d\'habitants en sous nutrition ne␣
 ↪sont pas ceux qui ont reçu le plus d\'aide alimentaire ')
print(f'Les pays suivants présentent la plus forte proportion de personnes en␣
 ↪état de sous nutrition 10 fois consécutives entre 2013 et 2à16')
print(sous_nutrition_pays_plus)
```

```
Les pays qui pésentent le plus d'habitants en sous nutrition ne sont pas ceux
qui ont reçu le plus d'aide alimentaire
Les pays suivants présentent la plus forte proportion de personnes en état de
sous nutrition 10 fois consécutives entre 2013 et 2à16
['Haïti', 'République populaire démocratique de Corée', 'Madagascar']
```

0.0.12 Évolution de l'aide alimentaire pour les 10 pays présentant la plus forte proportion de personnes

0.0.13 en état de sous nutrition

[166]:
```
# Aggrégation de sous_nutrition_pays_plus par pays pour obtenir les 10 pays␣
 ↪présentant la mlus forte proportion
# de personnes sous nutries entre 2013 et 2016
sous_nutrition_groupe=sous_nutrition_proportion_pays.drop(columns=['Année']).
 ↪groupby('Zone').sum().reset_index()

# Calcul de la colonne Proportion
sous_nutrition_groupe['Proportion']=round(sous_nutrition_groupe['Sous_nutrition']␣
 ↪/ sous_nutrition_groupe['Population'] * 100, 2)

# Tri de la table et conservation des 10 premières lignes
sous_nutrition_groupe=sous_nutrition_groupe.sort_values('Proportion',␣
 ↪ascending=False).head(10)

# Création d'une liste des 10 pays
liste_10pays=sous_nutrition_groupe['Zone'].tolist()
```

```
liste_10pays
```

[166]:
```
['Haïti',
 'République populaire démocratique de Corée',
 'Madagascar',
 'Libéria',
 'Tchad',
 'Timor-Leste',
 'Rwanda',
 'Lesotho',
 'Sierra Leone',
 'Mozambique']
```

[167]:
```
# Création d'une table aide alimentaire pour ces 10 pays
aide_alimentaire_10pays=aide_alimentaire.loc[aide_alimentaire['Zone'].
 ↪isin(liste_10pays), :]

# Aggrégation par pays
aide_alimentaire_10pays=aide_alimentaire_10pays.drop(columns=['Produit']).
 ↪groupby(['Zone', 'Année']).sum().reset_index()

# Expression de la colonne Aide_alimentaire en milliers de tonnes (elle est en␣
 ↪kg)
aide_alimentaire_10pays['Aide_alimentaire']=aide_alimentaire_10pays['Aide_alimentaire']␣
 ↪/ 1e6

aide_alimentaire_10pays.head(3)
```

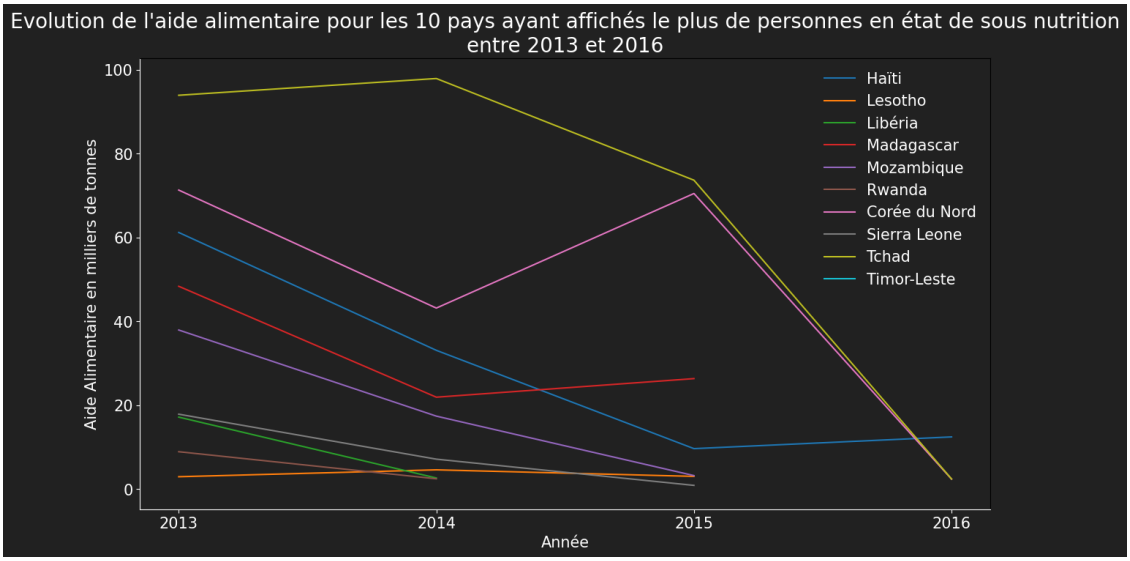
[167]:

	Zone	Année	Aide_alimentaire
0	Haïti	2013	61.214
1	Haïti	2014	33.108
2	Haïti	2015	9.666

[168]:
```
# Création de courbes pour chaque pays

# Import de MaxNLocator pour le forçage des années en entiers
from matplotlib.ticker import MaxNLocator

# Création d'une liste des pays
pays_unique_10 = aide_alimentaire_10pays['Zone'].unique()

fig, ax = plt.subplots(figsize=(15,8))

# Boucle pour générer les courbes
for pays in pays_unique_10:
    pays_data = aide_alimentaire_10pays.loc[aide_alimentaire_10pays['Zone'] ==␣
 ↪pays]
```

```
    line, = plt.plot(pays_data['Année'], pays_data['Aide_alimentaire'],␣
 ↪label=pays)  # Récupère les données pour chaque courbe
    # La virgule permet de décomposer la liste retournée par plt.plot

    # Persoonnalisation du graphe
    fig.patch.set_facecolor('#212121')
    ax.set_facecolor('#212121')
    ax.spines['bottom'].set_color('white')  # Modifie de la couleur de l'axe␣
 ↪des x
    ax.spines['left'].set_color('white')  # Modifie de la couleur de l'axe des y
    ax.tick_params(axis='x', labelsize=15, colors='white')  # Modifie la␣
 ↪couleur des valeurs de l'axe des x
    ax.tick_params(axis='y', labelsize=15, colors='white')  # Modifie la␣
 ↪couleur des valeurs de l'axe des y
    ax.xaxis.label.set_color('white')  # Modifie la couleur du label de l'axe␣
 ↪des x
    ax.yaxis.label.set_color('white')  # Modifie la couleur du label de l'axe␣
 ↪des y
    ax.xaxis.label.set_size(15)  # Modifie la taille du label de l'axe des x
    ax.yaxis.label.set_size(15)  # Modifie la taille de l'axe des y

# Afficher le nom des axes, la légende et le titre
plt.xlabel('Année')
plt.ylabel('Aide Alimentaire en milliers de tonnes')
plt.title(f'''Evolution de l\'aide alimentaire pour les 10 pays ayant affichés␣
 ↪le plus de personnes en état de sous nutrition
entre 2013 et 2016''', color='white', fontsize=20)

# Forcer l'affichage des années comme des entiers. Le premier afficahge affiche␣
 ↪les années comme des déciamux
#avec des moitiés d'années comme 2013,5
plt.gca().xaxis.set_major_locator(MaxNLocator(integer=True))

# Création d'une légende et modification de la couleur du texte
leg = plt.legend(loc = 'upper right', fontsize=15, frameon=False)
plt.setp(leg.get_texts(), color='white')

# Modification de 'République ppopulaire démocratique de Corée' qui est trop␣
 ↪long
for text in leg.get_texts():
    if text.get_text() == 'République populaire démocratique de Corée':
        text.set_text('Corée du Nord')

plt.show
```

[168]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```


Evolution de l'aide alimentaire pour les 10 pays ayant affichés le plus de personnes en état de sous nutrition entre 2013 et 2016
Aide Alimentaire en milliers de tonnes
Année
0
20
40
60
80
100
2013
2014
2015
2016
Haïti
Lesotho
Libéria
Madagascar
Mozambique
Rwanda
Corée du Nord
Sierra Leone
Tchad
Timor-Leste

[ ]:

[ ]: