```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

Calcul de la proportion de personnes en état de sous-nutrition en 2017

Création du DataFrame population et nombre de personnes en sous-nutrition pour l'année 2017 Comme indiqué par Julien dans ses notes, les valeurs de personnes en état de sous nutrition sont exprimées en million, pour la moyenne des 3 années, tandis que la population par pays est exprimée en millier.

Pour 2017, on peut donc conserver la moyenne des années 2016, 2017 et 2018, soit la tranche 2016-2018.

```
aide_alimentaire = pd.read_csv("aidealimentaire.csv")
dispo_alimentaire = pd.read_csv("dispoalimentaire.csv")
population = pd.read_csv("population.csv")
sous_nutrition = pd.read_csv("sousnutrition.csv")

sousNutrition = pd.read_csv("sousnutrition.csv")

sousNutrition.loc[sousNutrition["Valeur"] == "<0.1", "Valeur"] = 0

sousNutrition["Valeur"] = sousNutrition["Valeur"].fillna(0)

# DataFrame du nombre de personnes en sous-nutrition par pays en 2017
popMondiale = pd.read_csv("population.csv")

popMondialeParPays2017 = popMondiale[popMondiale["Année"] == 2017]

# DataFrame du nombre de personnes en sous-nutrition par pays en 2017

sousNutrition = pd.read_csv("sousnutrition.csv")

sousNutrition2017 = sousNutrition[sousNutrition["Année"] == "2016-
2018"]

# Créer un DataFrame composé des populations et du nombre de personnes
en sous-nutrition par pays en 2017

dfPopEtSousNutri = pd.merge(sousNutrition2017, popMondialeParPays2017,
on="Zone", how="inner")

# Arrondir les valeurs <0.1
dfPopEtSousNutri.loc[dfPopEtSousNutri["Valeur_x"] == "<0.1",
"Valeur_x"] = 0
```

```
# Conserver et renommer les colonnes utiles
dfPopEtSousNutri = dfPopEtSousNutri[["Zone", "Valeur_x", "Valeur_y"]]
dfPopEtSousNutri = dfPopEtSousNutri.rename(columns={"Valeur_x":
"Nombre personnes en sous-nutrition", \
                                                    "Valeur_y":
"Population"})

# Traiter comme des valeur numériques
dfPopEtSousNutri["Nombre personnes en sous-nutrition"] = \
pd.to_numeric(dfPopEtSousNutri["Nombre personnes en sous-
nutrition"]).fillna(0)


# Mettre les deux colonnes dans la même unité (nombre de personnes)
dfPopEtSousNutri["Nombre personnes en sous-nutrition"] *= 1000000
dfPopEtSousNutri["Population"] *= 1000

# Transformer le résultat en integer
dfPopEtSousNutri["Nombre personnes en sous-nutrition"] =
dfPopEtSousNutri["Nombre personnes en sous-nutrition"].astype(int)
dfPopEtSousNutri["Population"] =
dfPopEtSousNutri["Population"].astype(int)

display(dfPopEtSousNutri)
```

```
                                     Zone  \
0                              Afghanistan
1                           Afrique du Sud
2                                  Albanie
3                                  Algérie
4                                Allemagne
..                                     ...
198  Venezuela (République bolivarienne du)
199                               Viet Nam
200                                  Yémen
201                                 Zambie
202                               Zimbabwe

     Nombre personnes en sous-nutrition  Population
0                              10500000    36296113
1                               3100000    57009756
2                                100000     2884169
3                               1300000    41389189
4                                     0    82658409
..                                  ...         ...
198                             8000000    29402484
199                             6500000    94600648
200                                   0    27834819
201                                   0    16853599
```

```
202                                    0    14236595

[203 rows x 3 columns]
```

Calcul de la proportion de personnes en sous-nutrition en 2017

Calcul du nombre de personnes en sous-nutrition en 2017

```
nbPersSousNourrie2017 = dfPopEtSousNutri["Nombre personnes en sous-nutrition"].sum()

print("Nombre de personnes sous nourries dans le monde en 2017 = {:,.0f}".format(nbPersSousNourrie2017))
```

```
Nombre de personnes sous nourries dans le monde en 2017 = 535,700,000
```

Calcul de la population mondiale en 2017

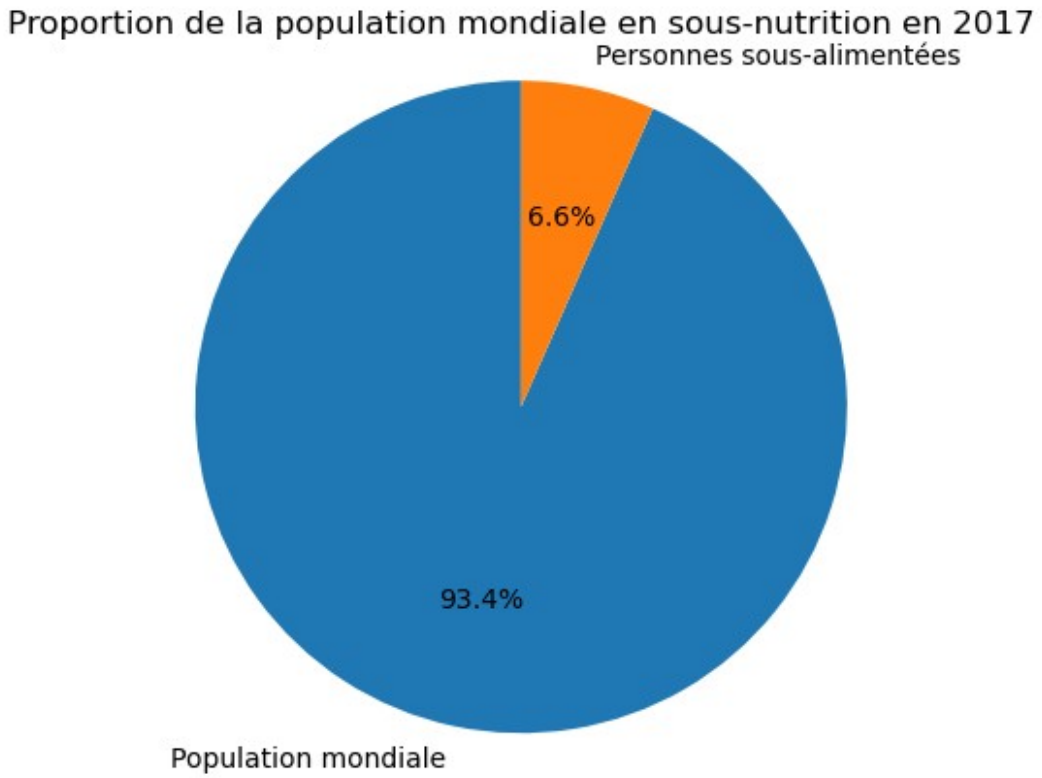
```
popMondiale2017 = dfPopEtSousNutri["Population"].sum()

print("Population mondiale en 2017 = {:,.0f}".format(popMondiale2017))
```

```
Population mondiale en 2017 = 7,543,798,769
```

Calcul de la proportion de personnes en sous-nutrition en 2017

```
proportionSousNutrition2017 = nbPersSousNourrie2017 / popMondiale2017 * 100

print("Proportion de la population mondiale en sous nutrition en 2017 = {:,.2f}%".format(proportionSousNutrition2017))

labels = ['Population mondiale', 'Personnes sous-alimentées']
sizes = [popMondiale2017, nbPersSousNourrie2017]
colors = ['#1f77b4', '#ff7f0e']

fig1, ax1 = plt.subplots()
ax1.pie(sizes, colors=colors, labels=labels, autopct='%0.1f%%', startangle=90)
ax1.axis('equal')

plt.title('Proportion de la population mondiale en sous-nutrition en 2017')
plt.show()
```

```
Proportion de la population mondiale en sous nutrition en 2017 = 7.10%
```



Nombre théorique de personnes qui pourraient être nourries selon la disponibilité alimentaire mondiale

Calculer la disponibilité alimentaire en Kcal/jour/pers nous permet de savoir combien de personnes pourraient être théoriquement nourries par jour dans chaque pays.

On partira pour se faire du principe qu'un humain a besoin en moyenne de 2500 Kcal par jour, ce chiffre pouvant en pratique varier selon la taille, l'âge, le poids, le sexe de chaque individu

Calcul de la disponibilité alimentaire pour chaque pays

```
# Calcul de la disponibilité alimentaire pour chaque pays

dispoAlimentaire = pd.read_csv("dispoalimentaire.csv")

dispoParPays = dispoAlimentaire.groupby("Zone").sum(numeric_only=True)

dispoParPays = dispoParPays[["Disponibilité alimentaire
(Kcal/personne/jour)"]]

display(dispoParPays)
```

```
                      Disponibilité alimentaire (Kcal/personne/jour)
Zone
Afghanistan                                                   2087.0
Afrique du Sud                                                3020.0
Albanie                                                       3188.0
Algérie                                                       3293.0
Allemagne                                                     3503.0
...                                                              ...
Émirats arabes unis                                           3275.0
Équateur                                                      2346.0
États-Unis d'Amérique                                         3682.0
Éthiopie                                                      2129.0
Îles Salomon                                                  2383.0

[174 rows x 1 columns]
```

Calcul du nombre de personnes théoriquement nourries par pays

```
habNourrisParPays = pd.DataFrame({'Nombre de personne théoriquement
nourries': \
                       dispoParPays['Disponibilité alimentaire
(Kcal/personne/jour)']/2500},
                      index = dispoParPays.index)

habNourrisParPays = habNourrisParPays.reset_index()

display(habNourrisParPays)
```

```
                       Zone  Nombre de personne théoriquement nourries
0               Afghanistan                                      0.8348
1             Afrique du Sud                                     1.2080
2                    Albanie                                     1.2752
3                    Algérie                                     1.3172
4                  Allemagne                                     1.4012
..                       ...                                        ...
169      Émirats arabes unis                                     1.3100
170                  Équateur                                    0.9384
171    États-Unis d'Amérique                                     1.4728
172                  Éthiopie                                    0.8516
173             Îles Salomon                                     0.9532

[174 rows x 2 columns]
```

Pondération des résultats par la population de chaque pays

```
# Ajouter la colonne population à notre dataframe.

df_ponderation = pd.merge(habNourrisParPays, popMondialeParPays2017,
on='Zone')

# Garder les colonnes utiles
```

```
df_ponderation = df_ponderation[['Zone', 'Nombre de personne
théoriquement nourries', 'Valeur']]

# Renommer la colonne Valeur en 'Population en millier d'habitants'

df_ponderation = df_ponderation.rename(columns={'Valeur' : "Population
en millier d'habitants"})

display(df_ponderation)
```

	Zone	Nombre de personne théoriquement nourries	Population en millier d'habitants
0	Afghanistan	0.8348	36296.113
1	Afrique du Sud	1.2080	57009.756
2	Albanie	1.2752	2884.169
3	Algérie	1.3172	41389.189
4	Allemagne	1.4012	82658.409
..	...	...	...
167	Émirats arabes unis	1.3100	9487.203
168	Équateur	0.9384	16785.361
169	États-Unis d'Amérique	1.4728	325084.756
170	Éthiopie	0.8516	106399.924
171	Îles Salomon	0.9532	636.039

```
[172 rows x 3 columns]
```

```
# Calculer le coeficient de pondération

df_ponderation['Coeficient de Pondération'] =
df_ponderation["Population en millier d'habitants"] /
df_ponderation["Population en millier d'habitants"].sum()

display(df_ponderation)
```

```
                      Zone  Nombre de personne théoriquement nourries
\
0              Afghanistan                                       0.8348

1           Afrique du Sud                                       1.2080

2                  Albanie                                       1.2752

3                  Algérie                                       1.3172

4                Allemagne                                       1.4012

..                     ...                                          ...

167    Émirats arabes unis                                       1.3100

168              Équateur                                       0.9384

169  États-Unis d'Amérique                                       1.4728

170              Éthiopie                                       0.8516

171           Îles Salomon                                       0.9532


     Population en millier d'habitants  Coeficient de Pondération
0                            36296.113                   0.004978
1                            57009.756                   0.007818
2                             2884.169                   0.000396
3                            41389.189                   0.005676
4                            82658.409                   0.011336
..                                 ...                        ...
167                           9487.203                   0.001301
168                          16785.361                   0.002302
169                         325084.756                   0.044582
170                         106399.924                   0.014592
171                            636.039                   0.000087
```

```
[172 rows x 4 columns]
```

Nombre de personnes qui pourraient être théoriquement nourries à l'échelle mondiale

```
# Utiliser la fonction 'average' de numpy pour faire la moyenne
pondérée ('weight')

values = df_ponderation['Nombre de personne théoriquement nourries']
weights = df_ponderation['Coeficient de Pondération']

moyennePonderee = np.average(values, weights = weights)

print("On pourrait nourrir {:,.4f} fois la population mondiale en 2017
(soit {:,.2f}%)"\
      .format(moyennePonderee, moyennePonderee*100))
```

```
On pourrait nourrir 1.1475 fois la population mondiale en 2017 (soit
114.75%)
```

```
nbPersTheoriquementNourriesMonde = float(popMondiale2017) *
moyennePonderee

print("Le nombre de personnes que l'on pourrait théoriquement nourrir
à l'échelle mondiale en 2017 est : {:,.0f}"\
      .format(nbPersTheoriquementNourriesMonde))
```

```
Le nombre de personnes que l'on pourrait théoriquement nourrir à
l'échelle mondiale en 2017 est : 8,656,651,491
```

Nombre de personnes qui pourraient être nourries uniquement avec des aliments d'origine végétale

Calcul de la disponibilité alimentaire des produits d'originie végétale pour chaque pays

```
dispoVege = dispoAlimentaire.loc[dispoAlimentaire["Origine"] ==
"vegetale"]

dispoVegeParPays = dispoVege.groupby("Zone").sum(numeric_only=True)

dispoVegeParPays = dispoVegeParPays[["Disponibilité alimentaire
(Kcal/personne/jour)"]]

display(dispoVegeParPays)
```

```
                      Disponibilité alimentaire (Kcal/personne/jour)
Zone
Afghanistan                                                   1871.0
Afrique du Sud                                                2533.0
```

```
Albanie                                                    2203.0
Algérie                                                    2915.0
Allemagne                                                  2461.0
...                                                           ...
Émirats arabes unis                                        2718.0
Équateur                                                   1732.0
États-Unis d'Amérique                                      2698.0
Éthiopie                                                   2005.0
Îles Salomon                                               2187.0

[174 rows x 1 columns]
```

Calcul du nombre de personnes théoriquement nourries par pays, uniquement avec des produits d'origine végétale

```
habNourrisParPaysVege = pd.DataFrame({"Nombre de personne
théoriquement nourries Vege": \
                      dispoVegeParPays['Disponibilité alimentaire
(Kcal/personne/jour)']/2500},
                     index = dispoVegeParPays.index)

habNourrisParPaysVege = habNourrisParPaysVege.reset_index()

display(habNourrisParPaysVege)
```

```
                     Zone  Nombre de personne théoriquement nourries
Vege
0              Afghanistan
0.7484
1           Afrique du Sud
1.0132
2                  Albanie
0.8812
3                  Algérie
1.1660
4                Allemagne
0.9844
..                     ...
...
169    Émirats arabes unis
1.0872
170               Équateur
0.6928
171  États-Unis d'Amérique
1.0792
172               Éthiopie
0.8020
173           Îles Salomon
0.8748
```

```
[174 rows x 2 columns]
```

Pondération des résultats par la population de chaque pays

```
# Ajouter la colonne population à notre dataframe.

df_ponderationVege = pd.merge(habNourrisParPaysVege,
popMondialeParPays2017, on='Zone')

# Garder les colonnes utiles

df_ponderationVege = df_ponderationVege[['Zone', 'Nombre de personne
théoriquement nourries Vege', 'Valeur']]

# Renommer la colonne Valeur en 'Population en millier d'habitants'

df_ponderationVege = df_ponderationVege.rename(columns={'Valeur' :
"Population en millier d'habitants"})

display(df_ponderationVege)
```

```
                      Zone  Nombre de personne théoriquement nourries
Vege  \
0              Afghanistan
0.7484
1           Afrique du Sud
1.0132
2                  Albanie
0.8812
3                  Algérie
1.1660
4                Allemagne
0.9844
..                     ...
...
167    Émirats arabes unis
1.0872
168               Équateur
0.6928
169  États-Unis d'Amérique
1.0792
170               Éthiopie
0.8020
171           Îles Salomon
0.8748

     Population en millier d'habitants
0                            36296.113
1                            57009.756
```

```
2                            2884.169
3                           41389.189
4                           82658.409
..                                ...
167                          9487.203
168                         16785.361
169                        325084.756
170                        106399.924
171                           636.039

[172 rows x 3 columns]
```

```
# Calculer le coeficient de pondération

df_ponderationVege['Coeficient de Pondération'] =
df_ponderationVege["Population en millier d'habitants"] /
df_ponderationVege["Population en millier d'habitants"].sum()

display(df_ponderationVege)
```

```
                     Zone  Nombre de personne théoriquement nourries
Vege  \
0              Afghanistan
0.7484
1            Afrique du Sud
1.0132
2                  Albanie
0.8812
3                  Algérie
1.1660
4                Allemagne
0.9844
..                     ...
...
167    Émirats arabes unis
1.0872
168               Équateur
0.6928
169  États-Unis d'Amérique
1.0792
170               Éthiopie
0.8020
171           Îles Salomon
0.8748

     Population en millier d'habitants  Coeficient de Pondération
0                            36296.113                   0.004978
1                            57009.756                   0.007818
2                             2884.169                   0.000396
3                            41389.189                   0.005676
```

```
4                                82658.409                    0.011336
..                                     ...                         ...
167                               9487.203                    0.001301
168                              16785.361                    0.002302
169                             325084.756                    0.044582
170                             106399.924                    0.014592
171                                636.039                    0.000087

[172 rows x 4 columns]
```

Nombre de personnes qui pourraient être théoriquement nourries, uniquement à partir de la disponibilité alimentaire d'origine végétale

```
# Utiliser la fonction 'average' de numpy pour faire la moyenne
pondérée ('weight')

values = df_ponderationVege['Nombre de personne théoriquement nourries
Vege']
weights = df_ponderationVege['Coeficient de Pondération']

moyennePondereeVege = np.average(values, weights = weights)

print("On pourrait nourrir {:,.2f}% de la population mondiale en 2017,
uniquement avec des produits\
 d'origine végétale".format(moyennePondereeVege*100))
```

```
On pourrait nourrir 94.68% de la population mondiale en 2017,
uniquement avec des produits d'origine végétale
```

```
nbPersNourriesMondeVege = float(popMondiale2017) * moyennePondereeVege

print("Le nombre de personnes que l'on pourrait théoriquement nourrir
à l'échelle mondiale en 2017, de cette manière\
 est : \n{:,.0f}".format(nbPersNourriesMondeVege))
```

```
Le nombre de personnes que l'on pourrait théoriquement nourrir à
l'échelle mondiale en 2017, de cette manière est :
7,142,814,191
```

Proportion de la disponibilité intérieure en fonction de l'usage

Calcul de la disponibilité intérieure totale

```
dispoAlimentaire = pd.read_csv("dispoalimentaire.csv")

dispoInterieure = dispoAlimentaire["Disponibilité intérieure"].sum()

print("La disponibilité intérieure totale est égale à : {:,.0f}
milliers de tonnes".format(dispoInterieure))
```

```
La disponibilité intérieure totale est égale à : 9,848,994 milliers de
tonnes
```

Proportion de la disponibilité intérieure concrètement utilisée pour l'alimentation humaine

```
nourriture = dispoAlimentaire["Nourriture"].sum()
print ("Total de la disponibilité alimentaire utilisée comme
nourriture : {:,.0f} milliers de tonnes".format(nourriture))
```

```
Total de la disponibilité alimentaire utilisée comme nourriture :
4,876,258 milliers de tonnes
```

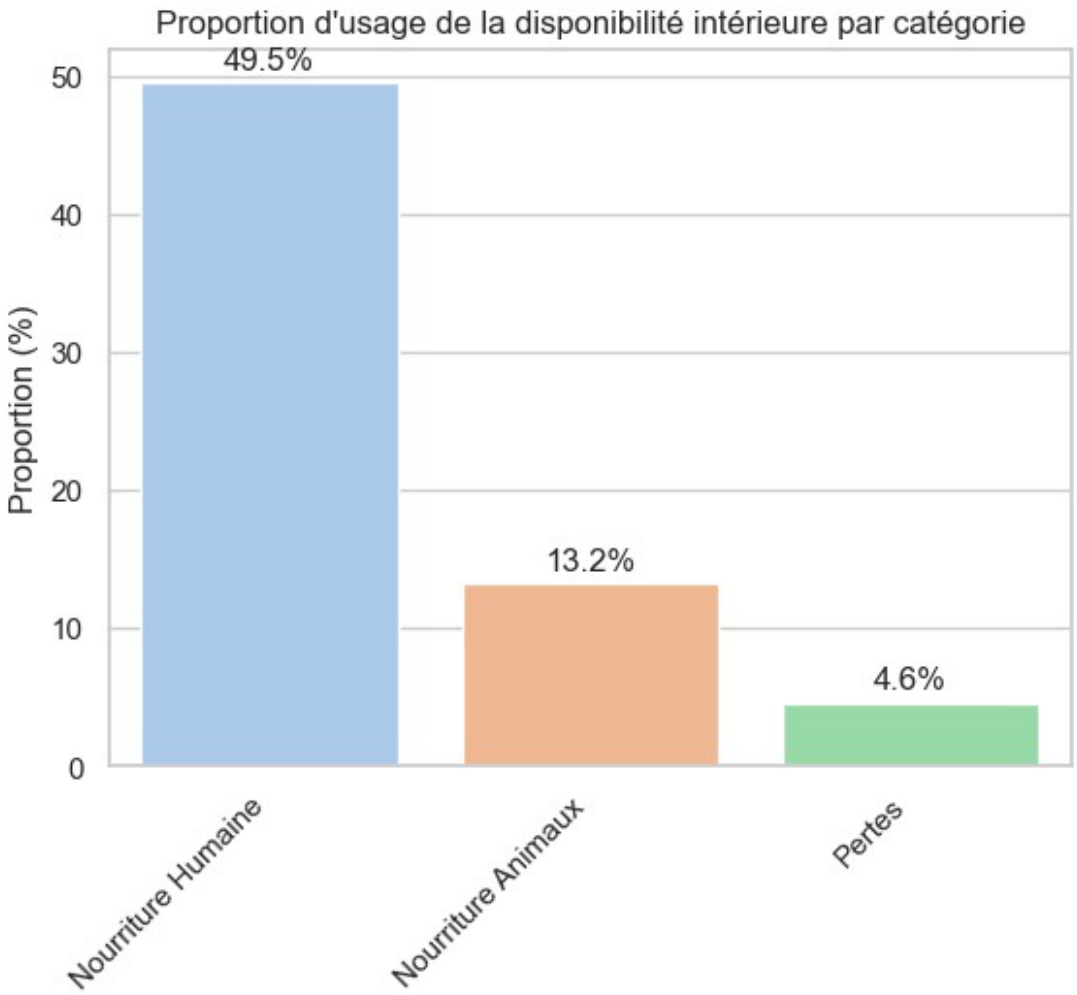
```
 # Calcul de la proportion d'utilisation de la nourriture
proportionNourriture = (nourriture / dispoInterieure) * 100

dfTemp = pd.DataFrame({'value': [proportionNourriture, 100 -
proportionNourriture]},
                  index=['Nourriture', 'Autres usages'])

sns.set_theme(style='whitegrid', palette='pastel')
fig, ax = plt.subplots()
ax.pie(dfTemp['value'], labels=dfTemp.index, autopct='%1.1f%%',
startangle = 90)
ax.axis('equal')
ax.set_title('Proportion de la disponibilité intérieure utilisée comme
nourriture')

plt.show()
```

```
---------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[8], line 2
      1 # Calcul de la proportion d'utilisation de la nourriture
----> 2 proportionNourriture = (nourriture / dispoInterieure) * 100
      4 dfTemp = pd.DataFrame({'value': [proportionNourriture, 100 -
proportionNourriture]},
      5                   index=['Nourriture', 'Autres usages'])
      7 sns.set_theme(style='whitegrid', palette='pastel')

NameError: name 'nourriture' is not defined
```

Détail des proportions

```
# Calcul des proportions

categories = ["Nourriture", "Aliments pour animaux", "Pertes"]

proportions = []
for cat in categories:
```

```
    proportion = (dispoAlimentaire[cat].sum() / dispoInterieure) * 100
    proportions.append(proportion)

dfProportions = pd.DataFrame({'value': proportions},
                  index=['Nourriture Humaine', 'Nourriture Animaux',
'Pertes'])


# Même données mais en version "bar"

sns.set_theme(style='whitegrid', palette='pastel')
fig, ax = plt.subplots()

sns.barplot(x=dfProportions.index, y=dfProportions['value'], ax=ax)

# Ajouter la valeur sur chaque bar
for i, v in enumerate(dfProportions['value']):
    ax.text(i, v + 1, '{:,.1f}%'.format(v), ha='center', fontsize=12)


ax.set_title("Proportion d'usage de la disponibilité intérieure par
catégorie")
ax.set_ylabel("Proportion (%)")
plt.xticks(rotation=45, ha="right")

plt.show()
```



Utilisation des céréales pour l'alimentation animale, par rapport à l'alimentation humaine

Proportion en pourcentage de céréales utilisées pour l'alimentation humaines et l'alimentation animale

```
cereales2 =
['Blé','Riz','Orge','Maïs','Seigle','Avoine','Millet','Sorgho','Céréal
es, Autres']

# Filtrer les lignes contenant une céréale dans la colonne "Produit"
dfCereales2 =
dispoAlimentaire.loc[dispoAlimentaire['Produit'].isin(cereales2),:]

display(dfCereales2)
```

```
                Zone           Produit   Origine  Aliments pour animaux
\
```

```
7       Afghanistan               Blé  vegetale                    NaN
12      Afghanistan  Céréales, Autres  vegetale                    NaN
32      Afghanistan              Maïs  vegetale                  200.0
34      Afghanistan             Millet  vegetale                    NaN
40      Afghanistan              Orge  vegetale                  360.0
...             ...               ...       ...                    ...
15537  Îles Salomon               Blé  vegetale                    NaN
15545  Îles Salomon  Céréales, Autres  vegetale                    NaN
15568  Îles Salomon              Maïs  vegetale                    NaN
15575  Îles Salomon              Orge  vegetale                    NaN
15593  Îles Salomon            Sorgho  vegetale                    0.0

       Autres Utilisations  Disponibilité alimentaire (Kcal/personne/jour)  \
7                      NaN                                           1369.0
12                     NaN                                              0.0
32                     NaN                                             21.0
34                     NaN                                              3.0
40                     NaN                                             26.0
...                    ...                                              ...
15537                  NaN                                            184.0
15545                  NaN                                              0.0
15568                  NaN                                              1.0
15575                  NaN                                              0.0
15593                  NaN                                              NaN

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

7	160.23
12	0.00
32	2.50
34	0.40
40	2.92
...	...
15537	25.37
15545	0.00
15568	0.15
15575	0.07
15593	NaN

	Disponibilité de matière grasse en quantité (g/personne/jour) \
7	4.69
12	0.00
32	0.30
34	0.02
40	0.24
...	...
15537	1.00
15545	0.00
15568	0.01
15575	NaN
15593	NaN

	Disponibilité de protéines en quantité (g/personne/jour) \
7	36.91
12	0.00
32	0.56
34	0.08
40	0.79
...	...
15537	5.19
15545	0.00
15568	0.03
15575	0.01
15593	NaN

```
       Disponibilité intérieure  Exportations - Quantité  \
7                        5992.0                      NaN
12                          0.0                      NaN
32                        313.0                      0.0
34                         13.0                      NaN
40                        524.0                      NaN
...                         ...                      ...
15537                      14.0                      0.0
15545                       0.0                      NaN
15568                       0.0                      NaN
15575                       1.0                      NaN
15593                       0.0                      NaN

       Importations - Quantité  Nourriture  Pertes  Production
Semences  \
7                         1173.0      4895.0   775.0      5169.0
322.0
12                           0.0         0.0     NaN         NaN
NaN
32                           1.0        76.0    31.0       312.0
5.0
34                           NaN        12.0     1.0        13.0
0.0
40                          10.0        89.0    52.0       514.0
22.0
...                          ...         ...     ...         ...       .
..
15537                       14.0        14.0     NaN         NaN
NaN
15545                        0.0         0.0     0.0         NaN
NaN
15568                        0.0         0.0     NaN         NaN
NaN
15575                        1.0         0.0     NaN         NaN
NaN
15593                        0.0         NaN     NaN         NaN
NaN

       Traitement  Variation de stock
7             NaN              -350.0
12            NaN                 NaN
32            NaN                 NaN
34            NaN                 NaN
40            NaN                 0.0
...           ...                 ...
15537         NaN                 0.0
15545         NaN                 NaN
15568         NaN                 NaN
15575         1.0                 NaN
```

```
15593         NaN                 NaN

[1323 rows x 18 columns]
```

```
proportionAnimaux = dfCereales2["Aliments pour
animaux"].sum()/dfCereales2["Disponibilité intérieure"].sum()*100
print("Proportion d'alimentation animale : {:.2f}
%".format(proportionAnimaux))

proportionHumain =
dfCereales2["Nourriture"].sum()/dfCereales2["Disponibilité
intérieure"].sum()*100
print("Proportion d'alimentation humaine : {:.2f}
%".format(proportionHumain))
```

```
Proportion d'alimentation animale : 43.49%
Proportion d'alimentation humaine : 33.74%
```

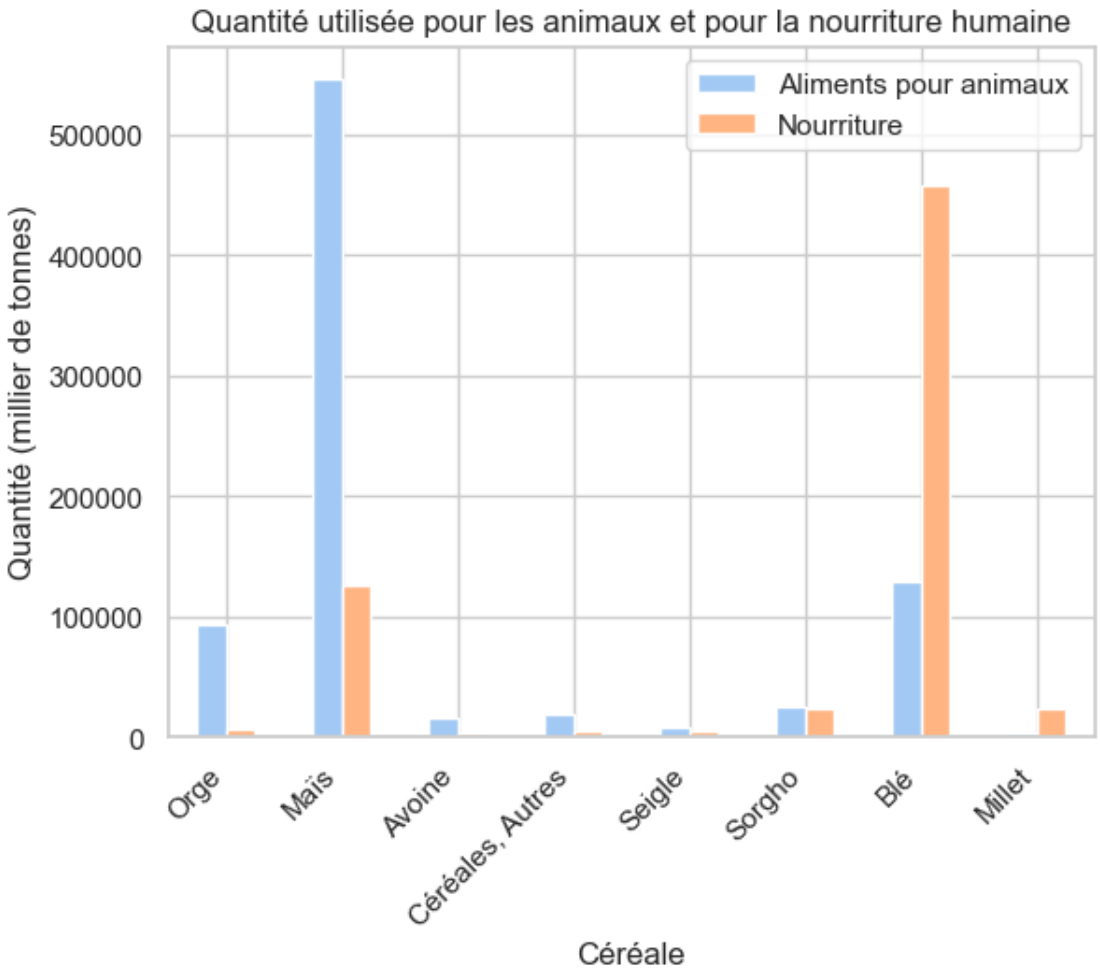
Répartition selon les céréales

```
# Calcul de la quantité, pour chaque céréale, utilisée pour
l'alimentation animale et pour l'alimentation humaine.

dfCereales2 = dfCereales2[["Produit", "Aliments pour animaux",
"Nourriture"]]

dfCereales2 = dfCereales2.groupby(dfCereales2["Produit"]).sum()

# Ajout d'une colonne "Proportion (%)" représentant la part affectée à
l'alimentation animale.

dfCereales2["Proportion d'usage pour les animaux par rapport à la
nourriture (%)"] = \
dfCereales2["Aliments pour animaux"] / (dfCereales2["Aliments pour
animaux"] + dfCereales2["Nourriture"]) * 100

dfCereales2 = dfCereales2.sort_values("Proportion d'usage pour les
animaux par rapport à la nourriture (%)", ascending=False)

dfCereales2 = dfCereales2.reset_index().head(9)

display(dfCereales2)
```

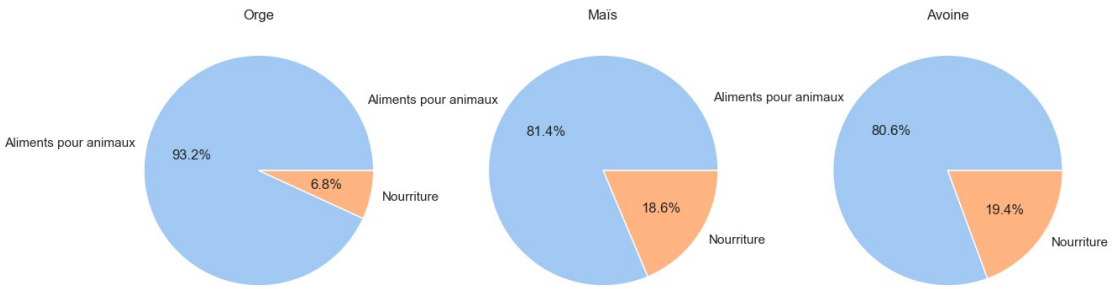
```
            Produit  Aliments pour animaux  Nourriture  \
0              Orge                92658.0      6794.0
1              Maïs               546116.0    125184.0
2            Avoine                16251.0      3903.0
3  Céréales, Autres                19035.0      5324.0
4            Seigle                 8099.0      5502.0
5            Sorgho                24808.0     24153.0
```

```
6               Blé               129668.0    457824.0
7            Millet                 3306.0     23040.0

   Proportion d'usage pour les animaux par rapport à la nourriture (%)

0                                              93.168564

1                                              81.352004

2                                              80.634117

3                                              78.143602

4                                              59.547092

5                                              50.668900

6                                              22.071449

7                                              12.548394
```

```
# Creation d'un graphique en barres
ax = dfCereales2.plot(x='Produit', y=['Aliments pour animaux',
'Nourriture'], kind='bar')

# Titre et noms des axes
sns.set_theme(style='whitegrid', palette='pastel')
ax.set_title('Quantité utilisée pour les animaux et pour la nourriture
humaine')
ax.set_xlabel('Céréale')
ax.set_ylabel('Quantité (millier de tonnes)')
plt.xticks(rotation=45, ha="right")

plt.show()
```




```
cereals = ["Orge", "Maïs", "Avoine"]
df_cereals = dfCereales2[dfCereales2["Produit"].isin(cereals)]

# Extraire les valeurs pour chaque céréale
cereals_data = df_cereals.groupby("Produit")[["Aliments pour animaux",
"Nourriture"]].sum()

# Créer un graphique à 3 colonnes
fig, axs = plt.subplots(1, 3, figsize=(15, 5))

# Créer un graphique pour chacune des 3 céréales les plus utilisées
pour l'alimentation animale
for i, cereal in enumerate(cereals):
    axs[i].pie(cereals_data.loc[cereal], labels=["Aliments pour
animaux", "Nourriture"], autopct="%1.1f%%")
    axs[i].set_title(cereal)

# Display the plot
plt.show()
```



Les 10 Pays où la proportion personnes sous-alimentées est la plus élevée en 2017

```
dfPopEtSousNutri['Proportion'] = dfPopEtSousNutri['Nombre personnes en sous-nutrition'] / dfPopEtSousNutri['Population']

dfPopEtSousNutri_sorted = dfPopEtSousNutri.sort_values('Proportion', ascending=False)

dfPopEtSousNutri_sorted = dfPopEtSousNutri_sorted[["Zone", "Proportion"]].head(10)

display(dfPopEtSousNutri_sorted)
```

	Zone	Proportion
78	Haïti	0.482592
157	République populaire démocratique de Corée	0.471887
108	Madagascar	0.410629
103	Libéria	0.382797
100	Lesotho	0.382494
183	Tchad	0.379576
161	Rwanda	0.350556
121	Mozambique	0.328109
186	Timor-Leste	0.321735
0	Afghanistan	0.289287

Les 10 Pays ayant le plus bénéficié de l'aide depuis 2013

```
aideAlimentaire = pd.read_csv("aidealimentaire.csv")

aideAlimentaireParPays = aideAlimentaire.groupby("Pays bénéficiaire").sum(numeric_only=True)

aideAlimentaireParPays_sorted = aideAlimentaireParPays.sort_values('Valeur', ascending=False)

aideAlimentaireParPays_sorted = aideAlimentaireParPays_sorted[["Valeur"]].head(10)
```

```
aideAlimentaireParPays_sorted =
aideAlimentaireParPays_sorted.rename(columns={"Valeur" : "Quantité
(tonnes)"})

display(aideAlimentaireParPays_sorted)
```

Pays bénéficiaire	Quantité (tonnes)
République arabe syrienne	1858943
Éthiopie	1381294
Yémen	1206484
Soudan du Sud	695248
Soudan	669784
Kenya	552836
Bangladesh	348188
Somalie	292678
République démocratique du Congo	288502
Niger	276344

Disponibilité par habitant

Pays ayant le plus de disponibilité par habitant

```
topDispoParPays = dispoParPays.sort_values("Disponibilité alimentaire
(Kcal/personne/jour)", ascending=False).head().reset_index()
display(topDispoParPays)
```

	Zone	Disponibilité alimentaire (Kcal/personne/jour)
0	Autriche	3770.0
1	Belgique	3737.0
2	Turquie	3708.0
3	États-Unis d'Amérique	3682.0
4	Israël	3610.0

Pays ayant le moins de disponibilité par habitant

```
lowDispoParPays = dispoParPays.sort_values("Disponibilité alimentaire
(Kcal/personne/jour)", ascending=True).head(10).reset_index()
display(lowDispoParPays)
```

	Zone	\
0	République centrafricaine	
1	Zambie	
2	Madagascar	

```
3                                  Afghanistan
4                                        Haïti
5  République populaire démocratique de Corée
6                                        Tchad
7                                     Zimbabwe
8                                      Ouganda
9                                   Timor-Leste

   Disponibilité alimentaire (Kcal/personne/jour)
0                                          1879.0
1                                          1924.0
2                                          2056.0
3                                          2087.0
4                                          2089.0
5                                          2093.0
6                                          2109.0
7                                          2113.0
8                                          2126.0
9                                          2129.0
```

Export du manioc par la Thaïlande

```
sousNutritionThailande = dfPopEtSousNutri[dfPopEtSousNutri["Zone"] ==
"Thaïlande"]
sousNutritionThailande = sousNutritionThailande.copy()
sousNutritionThailande['Proportion (%)'] =
sousNutritionThailande['Nombre personnes en sous-nutrition'] /
sousNutritionThailande['Population'] * 100
display(sousNutritionThailande)
```

```
          Zone  Nombre personnes en sous-nutrition  Population
Proportion  \
185  Thaïlande                                6200000    69209810
0.089583

     Proportion (%)
185        8.958268
```

Proportion de manioc exporté par la Thaïlande par rapport à sa production

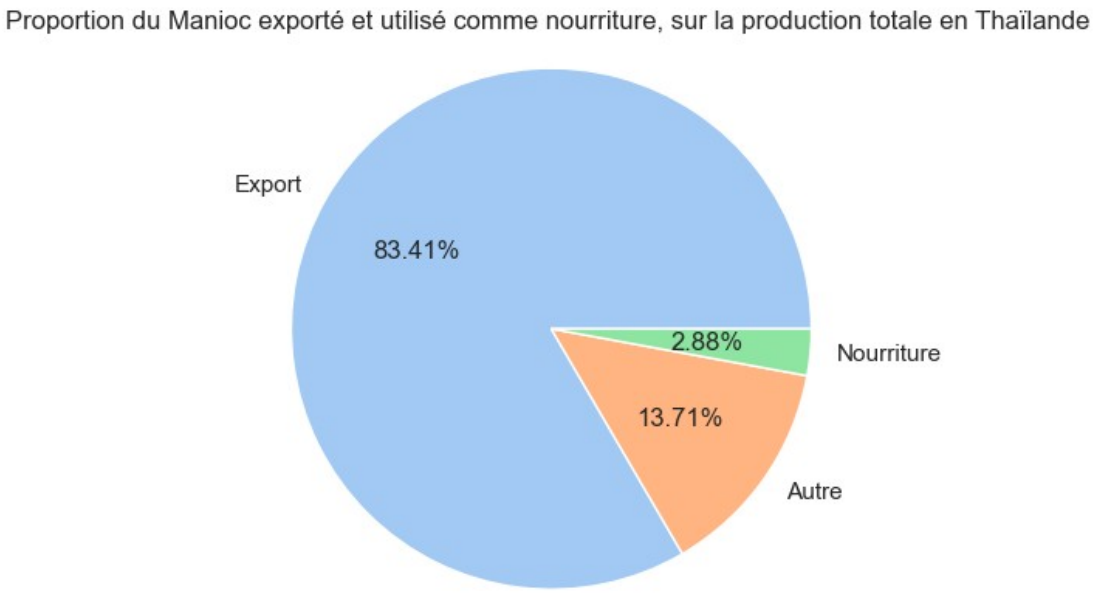
```
maniocGate = dispoAlimentaire[(dispoAlimentaire["Zone"]=="Thaïlande")
& (dispoAlimentaire["Produit"].str.contains("Manioc"))]

maniocGate = maniocGate[["Zone", "Produit", "Disponibilité
intérieure", "Exportations - Quantité", "Importations - Quantité",
"Nourriture", "Pertes", "Production"]]

display(maniocGate)
```

```
           Zone Produit  Disponibilité intérieure  Exportations -
Quantité  \
13809  Thaïlande  Manioc                     6264.0
25214.0

       Importations - Quantité  Nourriture  Pertes  Production
13809                   1250.0       871.0  1511.0     30228.0
```

```
export = maniocGate["Exportations - Quantité"].sum()
production = maniocGate["Production"].sum()
nourriture = maniocGate["Nourriture"].sum()

proportionExport = (export / production) * 100
proportionNourriture  = (nourriture / production) * 100

print("Proportion de manioc exporté : {:,.2f}
%".format(proportionExport))
print("Proportion de manioc utilisée pour l'alimentation : {:,.2f}
%".format(proportionNourriture))
```

```
Proportion de manioc exporté : 83.41%
Proportion de manioc utilisée pour l'alimentation : 2.88%
```

```
# Création d'un graphique pour visualiser les résultats

dfManioc = pd.DataFrame({'value': [proportionExport, 100 -
proportionExport - proportionNourriture, proportionNourriture]},
                  index=['Export', 'Autre', 'Nourriture'])

sns.set_theme(style='whitegrid', palette='pastel')
fig, ax = plt.subplots()
ax.pie(dfManioc['value'], labels=dfManioc.index, autopct='%1.2f%%')
ax.axis('equal')
ax.set_title('Proportion du Manioc exporté et utilisé comme
nourriture, sur la production totale en Thaïlande ')

plt.show()
```



Apport supplémentaire

Création d'un DataFrame avec la proportion de personnes en sous-nutrition pour chaque année

```
# Créer un mapping pour faire correspondre les années et les périodes
entre population.csv et sous_nutrition.csv
year_mapping = {
    '2012-2014': 2013,
    '2013-2015': 2014,
    '2014-2016': 2015,
    '2015-2017': 2016,
    '2016-2018': 2017,
    '2017-2019': 2018
}

# Appliquer le mapping sur sousNutritionTemp sans affecter
sousNutrition
sousNutritionTemp = sousNutrition.copy()
sousNutritionTemp['Année'] =
sousNutritionTemp['Année'].map(year_mapping)

# Merge les Dataframe
evolutionSousNutrition = pd.merge(sousNutritionTemp, popMondiale,
on=['Zone', 'Année'])

# Renommer les colonnes, unifier les unités et traiter comme des
integer
```

```
evolutionSousNutrition =
evolutionSousNutrition.rename(columns={"Valeur_x": "Nombre personnes
en sous-nutrition", \
                                       "Valeur_y":
"Population"})

evolutionSousNutrition["Nombre personnes en sous-nutrition"] = \
pd.to_numeric(evolutionSousNutrition["Nombre personnes en sous-
nutrition"]).fillna(0)

evolutionSousNutrition["Nombre personnes en sous-nutrition"] *=
1000000
evolutionSousNutrition["Nombre personnes en sous-nutrition"] =
evolutionSousNutrition["Nombre personnes en sous-
nutrition"].astype(int)
evolutionSousNutrition["Population"] *= 1000
evolutionSousNutrition["Population"] =
evolutionSousNutrition["Population"].astype(int)

evolutionSousNutrition["Proportion (%)"] =
evolutionSousNutrition["Nombre personnes en sous-
nutrition"]/evolutionSousNutrition["Population"]*100

display(evolutionSousNutrition)
```

```
             Zone  Année  Nombre personnes en sous-nutrition
Population  \
0     Afghanistan   2013                            8600000
32269589
1     Afghanistan   2014                            8800000
33370794
2     Afghanistan   2015                            8900000
34413603
3     Afghanistan   2016                            9700000
35383032
4     Afghanistan   2017                           10500000
36296113
...           ...    ...                                ...         .
..
1213     Zimbabwe   2014                                  0
13586707
1214     Zimbabwe   2015                                  0
13814629
1215     Zimbabwe   2016                                  0
14030331
1216     Zimbabwe   2017                                  0
14236595
1217     Zimbabwe   2018                                  0
14438802
```

```
      Proportion (%)
0          26.650479
1          26.370364
2          25.861866
3          27.414270
4          28.928718
...              ...
1213        0.000000
1214        0.000000
1215        0.000000
1216        0.000000
1217        0.000000

[1218 rows x 5 columns]
```

Sélectionner uniquement les 5 pays les plus en difficulté en 2018

```
evolutionSousNutrition_2018 =
evolutionSousNutrition.loc[evolutionSousNutrition['Année'] == 2018]

evolutionSousNutrition_sorted =
evolutionSousNutrition_2018.sort_values("Proportion
(%)",ascending=False)

top_5 = list(evolutionSousNutrition_sorted["Zone"].head(5))

print("Les 5 pays les plus en difficulté sont : ", top_5)
```

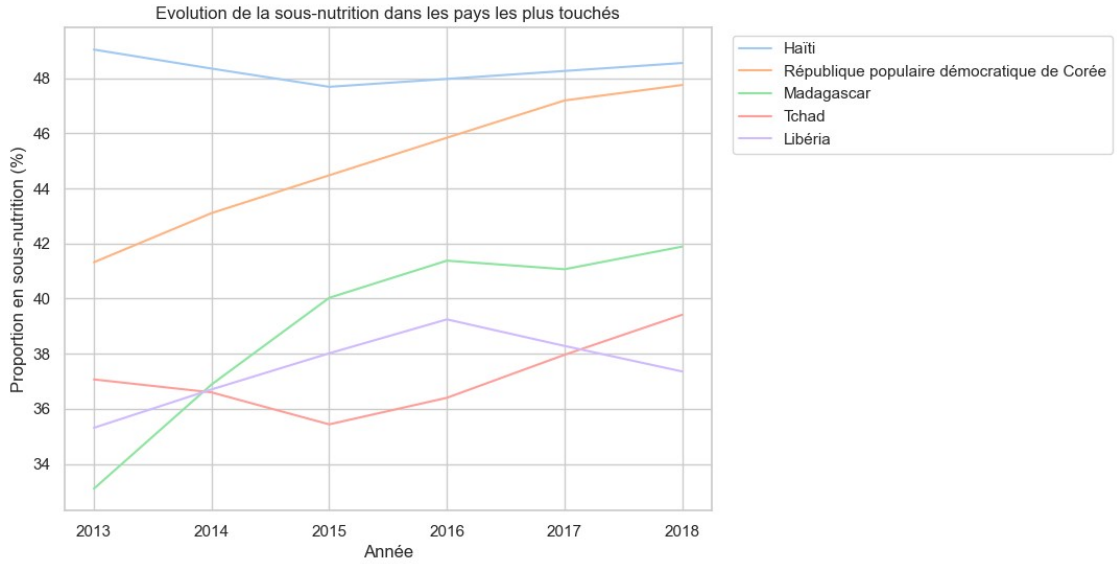
```
Les 5 pays les plus en difficulté sont :  ['Haïti', 'République
populaire démocratique de Corée', 'Madagascar', 'Tchad', 'Libéria']
```

```
# Création d'un dataframe avec toutes les valeurs pour chaque année et
pour chaque pays en difficulté
evolutionTop5 =
evolutionSousNutrition[evolutionSousNutrition["Zone"].isin(top_5)]
```

Visualisation de l'évolution de la proportion de chaque pays en sous-nutrition

```
# Create a figure and axis object
fig, ax = plt.subplots(figsize=(8,6))

# Iterate over the top 5 countries and plot their data
for pays in top_5:
    # filter data for the current country
    pays_data = evolutionTop5[evolutionTop5['Zone'] == pays]

    # plot the data for the current country
    ax.plot(pays_data['Année'], pays_data['Proportion (%)'],
label=pays)

# Add labels and legend
```

```
ax.set_xlabel('Année')
ax.set_ylabel('Proportion en sous-nutrition (%)')
ax.set_title('Evolution de la sous-nutrition dans les pays les plus
touchés')
legend = ax.legend(bbox_to_anchor=(1.02, 1), loc='upper left')

# Show the plot
plt.show()
```

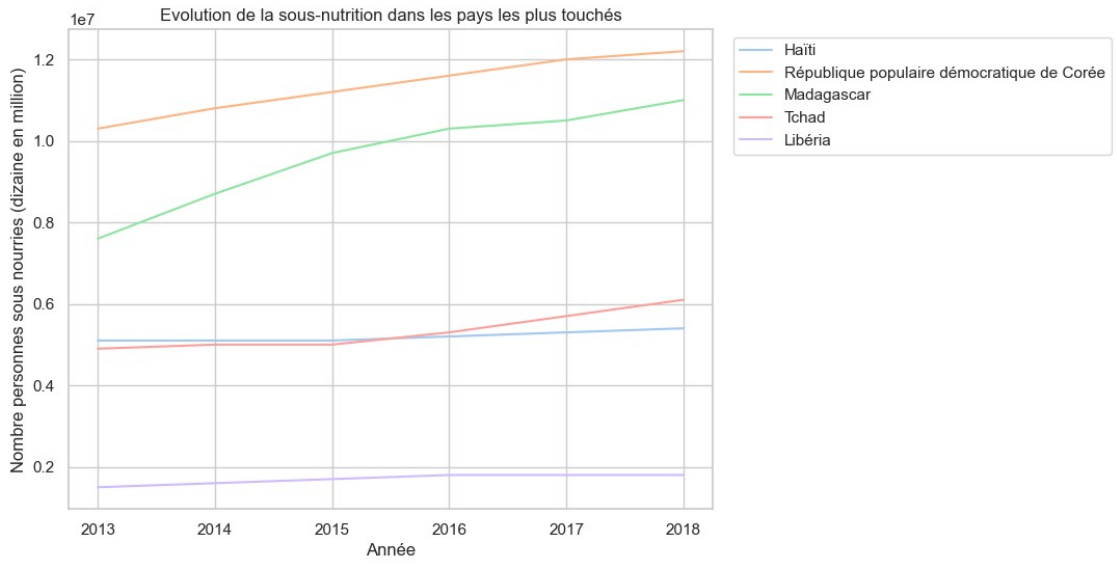


Ce graphique en courbe pose plusieurs courbes questions qui peuvent êtes discutées par l'organisation et dépendent des priorité que l'on souhaite donner, en effet on peut constater que :

La situation en Haïti reste la plus critique en termes de proportion de la population touchée par la sous-nutrition La situation en République populaire démocratique de Corée et à Madagascar se sont aggravées ces dernières années

```
Visualisation de l'évolution du nombre de personne en sous-nutrition

  Cell In[168], line 1
    Visualisation de l'évolution du nombre de personne en sous-
nutrition
                  ^
SyntaxError: invalid syntax


# Create a figure and axis object
fig, ax = plt.subplots(figsize=(8,6))

# Iterate over the top 5 countries and plot their data
for pays in top_5:
    # filter data for the current country
    pays_data = evolutionTop5[evolutionTop5['Zone'] == pays]
```

```
    # plot the data for the current country
    ax.plot(pays_data['Année'], pays_data['Nombre personnes en sous-nutrition'], label=pays)

# Add labels and legend
ax.set_xlabel('Année')
ax.set_ylabel('Nombre personnes sous nourries (dizaine en million)')
ax.set_title('Evolution de la sous-nutrition dans les pays les plus touchés')
legend = ax.legend(bbox_to_anchor=(1.02, 1), loc='upper left')

# Show the plot
plt.show()
```



Ce second graphique met en évidence qu'en termes de nombre de personnes touchées par la sous-nutrition les deux pays à la fois les plus touchés mais aussi où la situation s'est le plus aggravées sont :

La République populaire démocratique de Corée Madagascar