

Projet 4 du parcours d'Data Analyst
d' OpenClassrooms

Réalisez une étude de
santé publique avec
Python

LIANG Xiuting, 25/01/2021





SOMMAIRE

- 1. Analyse pour l'année 2017 (questions demandées par Marc)**
 - 2. Une étude pour chacun des pays (questions demandées par Mélanie)**
 - 3. Analyse de céréale (une étude des notes de Julien)**
 - 4. Analyse de manioc en Thaïlande (une étude des notes de Julien)**
- **Partie supplémentaire. Graphique interactive avec Voilà et ipywidgets**



Préparation

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

01

Importer les librairies Python

Fichiers utilisés:

- *sous_nutrition.csv*: un fichier d'insécurité alimentaire
- *population.csv*: un fichier de population de chaque pays
- *dispo_alimentaire.csv*: un fichier de disponibilité alimentaire pour l'année 2017
- *aide_alimentaire.csv*: un fichier d'aide alimentaire qui contient le pays bénéficiaire, la quantité qui a été donnée comme aide alimentaire, en tonnes, etc.

Information supplémentaire:

- La liste de *CEREALES ET PRODUITS CEREALIERS*, site FAO: <https://www.fao.org/3/X9892F/x9892f03.htm>

02

Fichiers utilisés et Information supplémentaire

```
df_snutrition = pd.read_csv('sous_nutrition.csv')
df_snutrition
```

	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

01 Importation

```
# Vérifier les zones uniques
print('Les zones uniques: '
```

```
Les zones uniques: 203
```

04 Zone unique

```
# Vérifier les non-null, et les types. et trouver que
df_snutrition.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1218 entries, 0 to 1217
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Zone    1218 non-null   object  
 1   Année   1218 non-null   object  
 2   Valeur  624 non-null   object  
dtypes: object(3)
memory usage: 28.7+ KB
```

02 Info

```
# Vérifier s'il y a des données
doublon_sn = df_snutrition.duplicated()
print('Le nombre de doublon: '
```

```
Le nombre de doublon: 0
```

03 Doublon

```
# Calculer le pourcentage de NaN
```

```
na_num = round(df_snutrition['Valeur'].isna().sum()/len(df_snutrition['Valeur'])*100,2)
print('NaN pourcentage: ' + str(na_num) + '%')
```

```
NaN pourcentage: 48.77%
```

05 NaN

```
# Remplacer les valeur 'NaN' par '0'
df_snutrition['Valeur'] = df_snutrition['Valeur'].fillna('0')
df_snutrition
```

	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
...

01 Remplacer NaN

```
# Remplacer '<0.1' par '0'
df_snutrition.loc[df_snutrition['Valeur']=='<0.1','Valeur']='0'
df_snutrition['Valeur']

0    8.6
1    8.8
2    8.9
3    9.7
4    10.5
...
1213    0
1214    0
1215    0
1216    0
1217    0
Name: Valeur, Length: 1218, dtype: object
```

02 Remplacer '<0.1'

```
# Remplacer Les noms de 'Année'
df_snutrition.loc[df_snutrition['Année'] == '2012-2014', 'Année'] = '2013'
df_snutrition.loc[df_snutrition['Année'] == '2013-2015', 'Année'] = '2014'
df_snutrition.loc[df_snutrition['Année'] == '2014-2016', 'Année'] = '2015'
df_snutrition.loc[df_snutrition['Année'] == '2015-2017', 'Année'] = '2016'
df_snutrition.loc[df_snutrition['Année'] == '2016-2018', 'Année'] = '2017'
df_snutrition.loc[df_snutrition['Année'] == '2017-2019', 'Année'] = '2018'
df_snutrition.head()
```

	Zone	Année	Valeur
0	Afghanistan	2013	8.6
1	Afghanistan	2014	8.8
2	Afghanistan	2015	8.9
3	Afghanistan	2016	9.7
4	Afghanistan	2017	10.5

03 Changer les noms de 'Année'

```
# Changer les types de 'Année' et 'Valeur' à 'int' et 'float'.
df_snutrition['Valeur'] = pd.to_numeric(df_snutrition['Valeur'])
df_snutrition['Année'] = pd.to_numeric(df_snutrition['Année'])
df_snutrition.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1218 entries, 0 to 1217
Data columns (total 3 columns):
 # Column Non-Null Count Dtype 
---  ---  ---  --- 
0   Zone    1218 non-null   object 
1   Année   1218 non-null   int64  
2   Valeur  1218 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 28.7+ KB
```

```
# La jointure des fichiers 'df_snutrition' et 'df_snu_pop' et l
df_snu_pop = pd.merge(df_snutrition, df_pop, on= ['Zone', 'Anné
df_snu_pop = df_snu_pop.rename(columns={'Valeur_x': 'sous_pop_m
df_snu_pop.head()
```

	Zone	Année	sous_pop_million	population_mille
0	Afghanistan	2013	8.6	32269.589
1	Afghanistan	2014	8.8	33370.794
2	Afghanistan	2015	8.9	34413.603
3	Afghanistan	2016	9.7	35383.032
4	Afghanistan	2017	10.5	36296.113

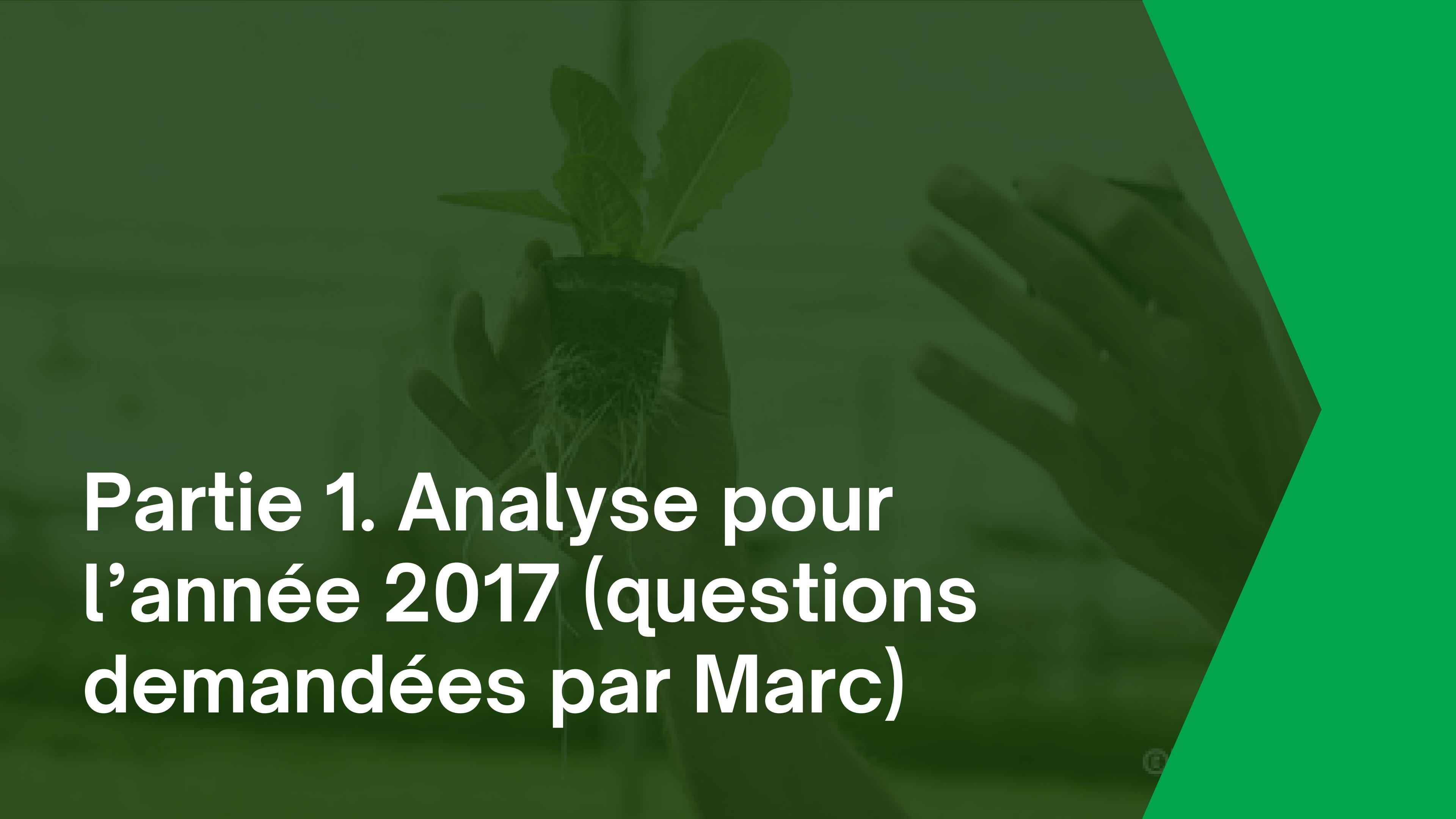
04 Changer les types

05 Faire la jointure

```
# Selectioner la partie de l'année 2017
df_snu17 = df_snu_pop.loc[df_snu_pop['Année'] == 2017, :]
df_snu17.head()
```

	Zone	Année	sous_pop_million	population_mille
4	Afghanistan	2017	10.5	36296.113
10	Afrique du Sud	2017	3.1	57009.756
16	Albanie	2017	0.1	2884.169
22	Algérie	2017	1.3	41389.189
28	Allemagne	2017	0.0	82658.409

06 Selectionner 2017



Partie 1. Analyse pour l'année 2017 (questions demandées par Marc)

Question 1: la proportion de personne en état de sous-nutrition

01

Calculer la population totale et la population sous nutrition par pays

```
# Calculer la population totale et la population sous nutrition par pays
df_snu17['sous_nutrition_pop'] = df_snu17 ['sous_pop_million'] * 1000000
df_snu17['population'] = df_snu17 ['population_mille'] * 1000
df_snu17.head()

C:\Users\xiuti\AppData\Local\Temp\ipykernel_37312\2403251965.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df_snu17['sous_nutrition_pop'] = df_snu17 ['sous_pop_million'] * 1000000
C:\Users\xiuti\AppData\Local\Temp\ipykernel_37312\2403251965.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df_snu17['population'] = df_snu17 ['population_mille'] * 1000
```

	Zone	Année	sous_pop_million	population_mille	sous_nutrition_pop	population
4	Afghanistan	2017	10.5	36296.113	10500000.0	36296113.0
10	Afrique du Sud	2017	3.1	57009.756	3100000.0	57009756.0
16	Albanie	2017	0.1	2884.169	100000.0	2884169.0
22	Algérie	2017	1.3	41389.189	1300000.0	41389189.0
...

02

Calculer: la proportion de personne en état de sous-nutrition mondial

Résultat Q1: la proportion de personne en état de sous-nutrition mondial

```
# Calculer la proportion avec la formule: (la population sous nutrition) / (la population totale mondiale) *100
sn_world = round(sum(df_snu17['sous_nutrition_pop'])/sum(df_snu17['population'])*100,2)
print(' La proportion de personnes en état de sous-nutrition: ' + str(sn_world) + '%')
```

La proportion de personnes en état de sous-nutrition: 7.1%



Question 2: le nombre théorique de personnes qui pourraient être nourries

1. Import file 'dispo_alimentaire.csv'

```
df_dispoa = pd.read_csv('dispo_alimentaire.csv')
df_dispoa.head()
```

Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)
0	Afghanistan	Abats Comestible	animale	NaN	NaN	5.0 1.72
1	Afghanistan	Agrumes, Autres	vegetale	NaN	NaN	1.0 1.29
2	Afghanistan	Aliments pour enfants	vegetale	NaN	NaN	1.0 0.06
3	Afghanistan	Ananas	vegetale	NaN	NaN	0.0 0.00
...

01

Importation

2. Disponibilité alimentaire (Kcal/personne/jour) totale par pays

```
# Calculer la disponibilité alimentaire (Kcal/personne/jour) totale par pays dans le tableau 'df_dispo_pays'
df_dispo_pays = df_dispoa.groupby('Zone').sum()[['Disponibilité alimentaire (Kcal/personne/jour)']]
df_dispo_pays
```

Zone	Disponibilité alimentaire (Kcal/personne/jour)
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

02

Calculer la disponibilité alimentaire (Kcal/personne/jour) totale par pays

df_snu17.head()

	Zone	Année	sous_pop_million	population_mille	sous_nutrition_pop	population
4	Afghanistan	2017	10.5	36296.113	10500000.0	36296113.0
10	Afrique du Sud	2017	3.1	57009.756	3100000.0	57009756.0
16	Albanie	2017	0.1	2884.169	100000.0	2884169.0
22	Algérie	2017	1.3	41389.189	1300000.0	41389189.0
28	Allemagne	2017	0.0	82658.409	0.0	82658409.0

```
# La jointure des tableaux 'df_dispo_pays' et 'df_snu17'
df_dispo_final = pd.merge(df_dispo_pays, df_snu17, on = 'Zone', how='inner')
df_dispo_final.head()
```

	Zone	Disponibilité alimentaire (Kcal/personne/jour)	Année	sous_pop_million	population_mille	sous_nutrition_pop	population
0	Afghanistan	2087.0	2017	10.5	36296.113	10500000.0	36296113.0
1	Afrique du Sud	3020.0	2017	3.1	57009.756	3100000.0	57009756.0
2	Albanie	3188.0	2017	0.1	2884.169	100000.0	2884169.0
3	Algérie	3293.0	2017	1.3	41389.189	1300000.0	41389189.0
4	Allemagne	3503.0	2017	0.0	82658.409	0.0	82658409.0

03

La jointure des 'df_dispo_pays' et 'df_snu17'

Question 2: le nombre théorique de personnes qui pourraient être nourries

04

Calculer la disponibilité alimentaire totale 2017 de chaque pays

```
# Calculer la disponibilité alimentaire totale 2017 de chaque pays
```

```
df_dispo_final['dispo_year'] = df_dispo_final['Disponibilité alimentaire (Kcal/personne/jour)']*df_dispo_final['population']*365
df_dispo_final.head()
```

	Zone	Disponibilité alimentaire (Kcal/personne/jour)	Année	sous_pop_million	population_mille	sous_nutrition_pop	population	dispo_year
0	Afghanistan	2087.0	2017	10.5	36296.113	10500000.0	36296113.0	2.764875e+13
1	Afrique du Sud	3020.0	2017	3.1	57009.756	3100000.0	57009756.0	6.284185e+13
2	Albanie	3188.0	2017	0.1	2884.169	100000.0	2884169.0	3.356077e+12
3	Algérie	3293.0	2017	1.3	41389.189	1300000.0	41389189.0	4.974753e+13
4	Allemagne	3503.0	2017	0.0	82658.409	0.0	82658409.0	1.056866e+14

05

La somme de la disponibilité mondiale

Calculer le nombre de personnes qui pourraient être nourries

Note: Les besoins de calories par personne par jour : 2500 cal; 2500*365 = les besoins chaque années par personne

```
# La somme de la disponibilité mondiale
dispo_all_sum = sum(df_dispo_final['dispo_year'])
```

```
# Calculer le nombre de personnes qui pourraient être nourries
```

```
dispo_all = int(dispo_all_sum/(2500*365))
```

```
print('Le nombre théorique de personnes qui pourraient être nourries: ' + str(dispo_all))
```

Le nombre théorique de personnes qui pourraient être nourries: 8367593850



Question 3: idem pour la disponibilité alimentaire des produits végétaux

```
# Selectioner l'origine de 'vegetale'  
df_vegi = df_dispoa.loc[df_dispoa['Origine']=='vegetale', :]  
df_vegi.head()
```

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)
1	Afghanistan	Agrumes, Autres	vegetale	NaN	NaN	1.0	1.2
2	Afghanistan	Aliments pour enfants	vegetale	NaN	NaN	1.0	0.0
3	Afghanistan	Ananas	vegetale	NaN	NaN	0.0	0.0
4	Afghanistan	Bananes	vegetale	NaN	NaN	4.0	2.7
6	Afghanistan	Bière	vegetale	NaN	NaN	0.0	0.0

```
# La disponibilité totale des produits végétaux par personne par jour de chaque pays  
df_vegi_pays = df_vegi.groupby('Zone').sum()[['Disponibilité alimentaire (Kcal/personne/jour)']]  
df_vegi_pays.head()
```

Disponibilité alimentaire (Kcal/personne/jour)	
Zone	
Afghanistan	1871.0
Afrique du Sud	2533.0
Albanie	2203.0
Algérie	2915.0
Allemagne	2461.0

01

Selectioner l'origine de 'vegetale'

03

La jointure de 'df_vegi_pays'
et 'df_snu17' et les renommer

02

La disponibilité totale des produits végétaux par personne par jour de chaque pays

```
df_vegi_pop = pd.merge(df_vegi_pays, df_snu17, on = 'Zone', how='inner')
df_vegi_pop = df_vegi_pop.rename(columns={'Disponibilité alimentaire (%)': 'Disponibilité alimentaire (%)'})
df_vegi_pop.head()
```

Zone	Disponibilité végétal (Kcal/personne/jour)	Année	sous_pop_million	population_mille	sous_nutrition_pop	population
0 Afghanistan	1871.0	2017	10.5	36296.113	10500000.0	36296113.0
1 Afrique du Sud	2533.0	2017	3.1	57009.756	3100000.0	57009756.0
2 Albanie	2203.0	2017	0.1	2884.169	100000.0	2884169.0
3 Algérie	2915.0	2017	1.3	41389.189	1300000.0	41389189.0
4 Allemagne	2514.0	2017	8.0	32552.162	900000.0	32552162.0

Question 3: idem pour la disponibilité alimentaire des produits végétaux

04

Calculer le nombre de personnes qui pourraient être nourries par les produits végétaux

```
# Calculer le nombre de personnes qui pourraient être nourries par les produits végétaux  
vegi_year_sum = sum(df_vegi_pop['Disponibilité végétal (Kcal/personne/jour)']*df_vegi_pop['population']*365)  
dispo_vegi = int(vegi_year_sum/(2500*365))  
print('Le nombre théorique de personnes qui pourraient être nourries par des produits végétaux: ' + str(dispo_vegi))
```

Le nombre théorique de personnes qui pourraient être nourries par des produits végétaux: 6904305684



05

Analyse supplémentaire: le pourcentage entre des produits végétaux et la disponibilité alimentaire totale

Analyse supplémentaire: le pourcentage entre des produits végétaux et la disponibilité alimentaire totale

```
# Calculer la proportion des produits végétaux  
vegi_proportion = round(vegi_year_sum/dispo_all_sum * 100, 2)  
print('Le pourcentage des produits végétaux: ' + str(vegi_proportion) + '%')
```

Le pourcentage des produits végétaux: 82.51%



Question 4. l'utilisation de la disponibilité intérieure, en particulier la part qui est attribuée à l'alimentation animale, celle qui est perdue et celle qui est concrètement utilisée pour l'alimentation humaine

01

Calculer les proportions des parties d'utilisation

```
part_function = round(df_dispoa[['Aliments pour animaux', 'Autres Utilisations', 'Nourriture',  
'Pertes','Semences','Traitement']].sum()/df_dispoa['Disponibilité intérieure'].sum()*100,2)  
part_function
```

```
# Calculer les proportions des parties d'utilisation  
part_function = round(df_dispoa[['Aliments pour animaux', 'Autres Utilisations', 'Nourriture', 'Pertes','Semences','Traitement']]  
part_function
```

```
Aliments pour animaux      13.24  
Autres Utilisations        8.78  
Nourriture                  49.51  
Pertes                      4.61  
Semences                     1.57  
Traitement                   22.38  
dtype: float64
```



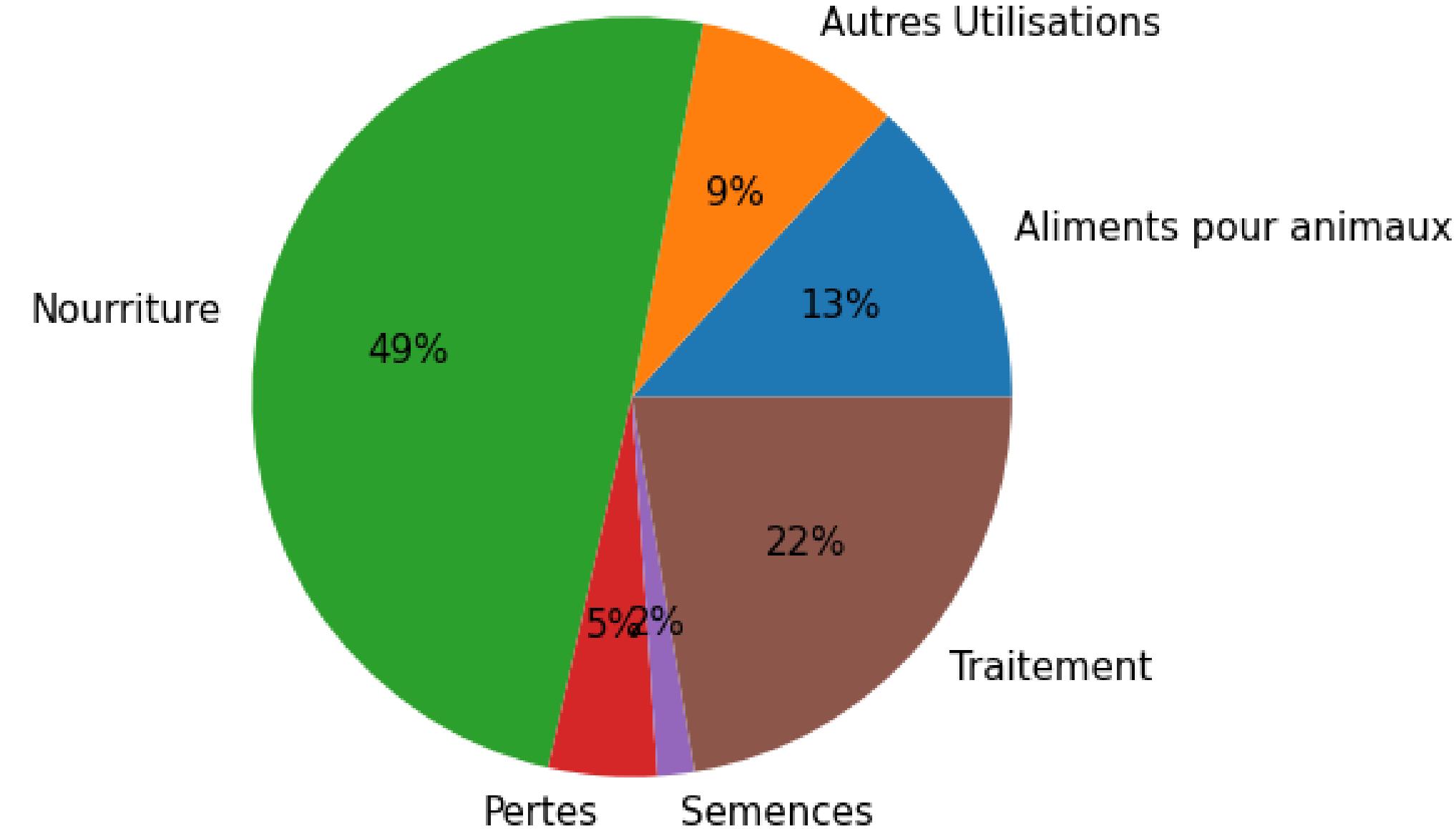
Question 4. l'utilisation de la disponibilité intérieure, en particulier la part qui est attribuée à l'alimentation animale, celle qui est perdue et celle qui est concrètement utilisée pour l'alimentation humaine

02

Présenter les proportions avec le camembert

```
# Présenter les proportions avec le camembert
plt.figure(figsize=(15,7))
part_function.plot.pie(autopct='%.0f%%', fontsize = 15, ylabel = (''))
plt.title("L'utilisation de la disponibilité intérieure", fontsize = 25)|
```

L'utilisation de la disponibilité intérieure



Analyse supplémentaire: la tendance de la population en état de sous-nutrition

01

Analyse supplémentaire: la tendance de la population en état de sous-nutrition

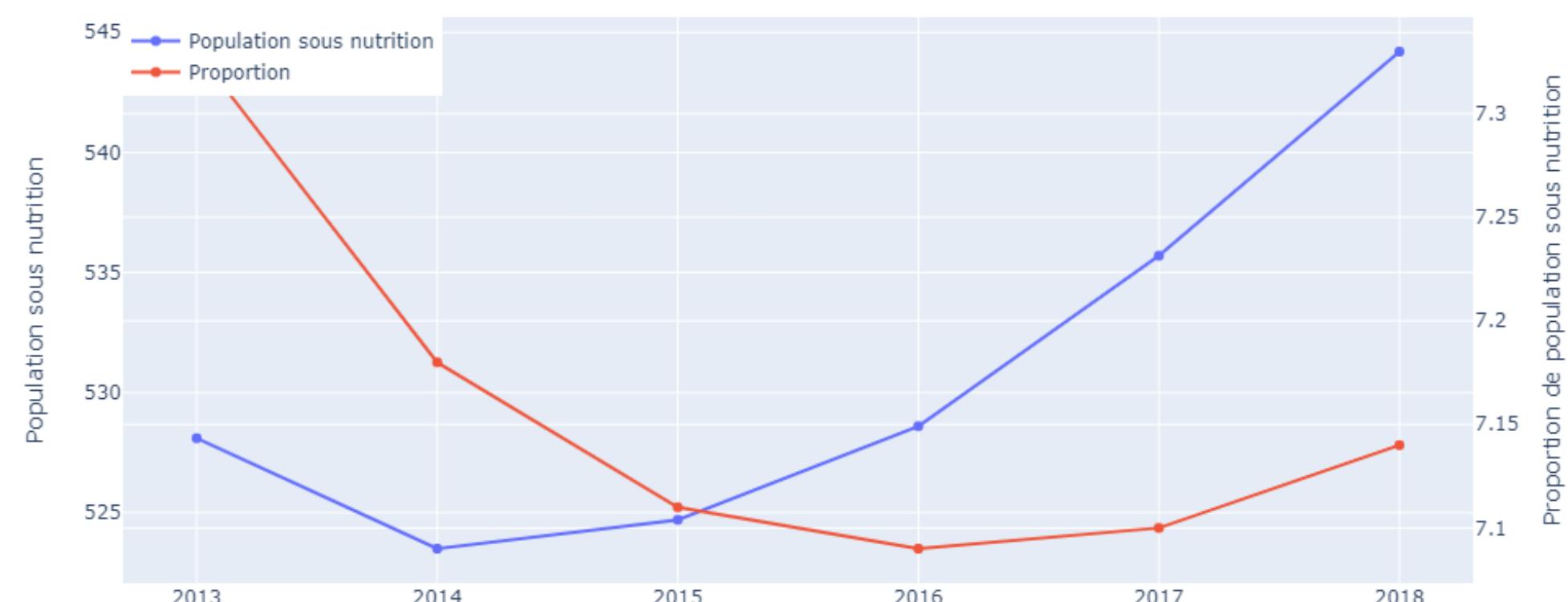
```
# Calculer les sommes de la population en état de sous-nutrition, de la population mondiale et de la proportion sous-nutrition
trend_snu = df_snu_pop.groupby('Année').sum()
trend_snu['trend_proportion'] = round(((trend_snu['sous_pop_million']*1000000)/(trend_snu['population_mille']*1000))*100,2)
trend_snu = trend_snu.reset_index()
trend_snu

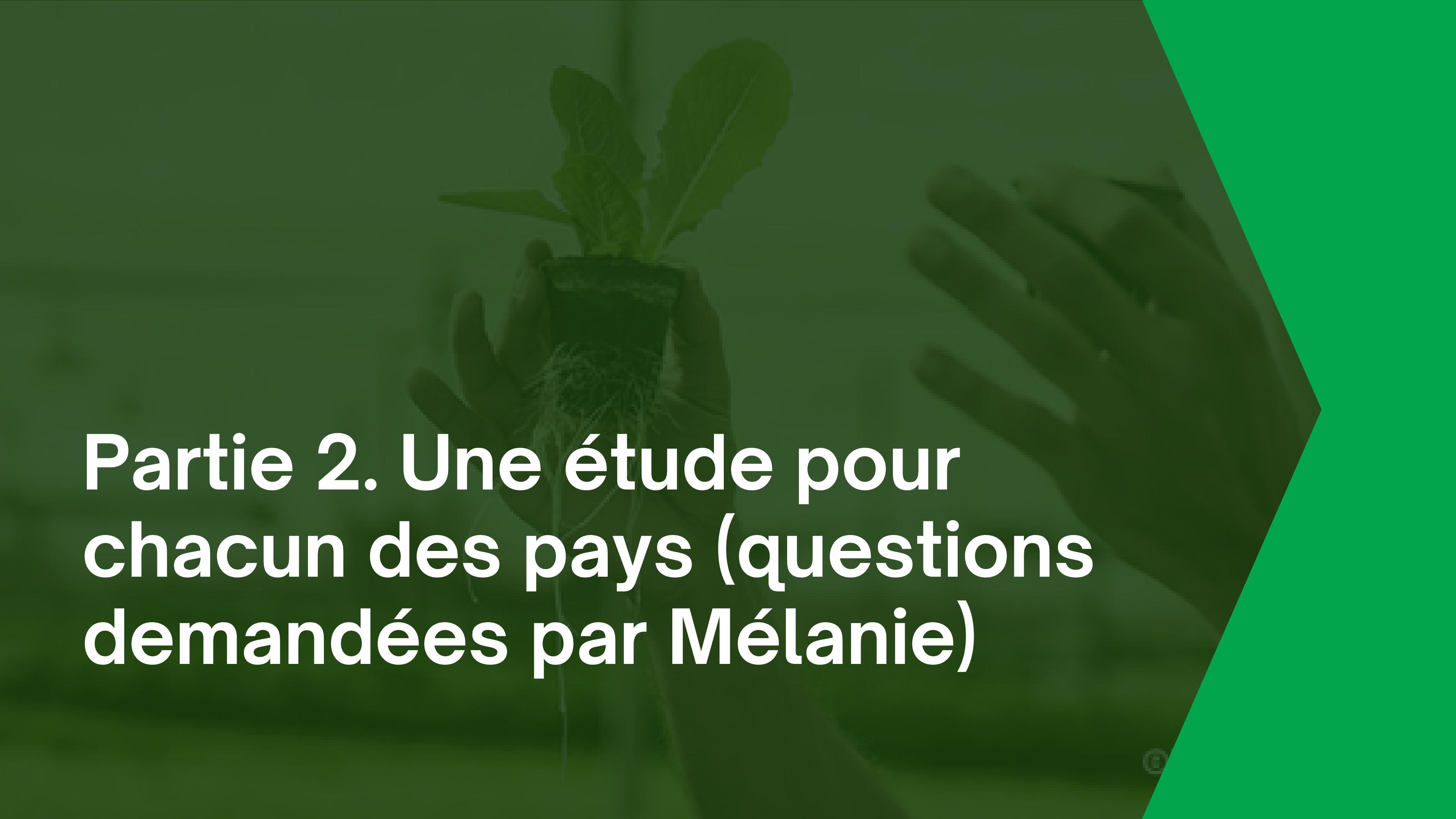
Année sous_pop_million population_mille trend_proportion
0 2013 528.1 7206670.140 7.33
1 2014 523.5 7291346.928 7.18
2 2015 524.7 7375818.210 7.11
3 2016 528.6 7460004.222 7.09
4 2017 535.7 7543798.779 7.10
5 2018 544.2 7626986.351 7.14

# Visualisation de la tendance avec le graphique en courbes
line_trend_snu = go.Scatter(x= trend_snu['Année'], y=trend_snu['sous_pop_million'], name='Population sous nutrition')
line_trend_proportion = go.Scatter(x= trend_snu['Année'], y=trend_snu['trend_proportion'],
                                    name='Proportion', xaxis='x', yaxis= 'y2')
layout = go.Layout(title='La tendance de la population en état de sous_nutrition, 2013-2018',
                    yaxis=dict(title = 'Population sous nutrition'),
                    yaxis2=dict(title = 'Proportion de population sous nutrition', overlaying = 'y', side = 'right'),
                    legend=dict(x=0, y=1))
data1=[line_trend_snu,line_trend_proportion]
fig_trend = go.Figure(data = data1, layout=layout)
fig_trend.show()
```

02

La tendance de la population en état de sous_nutrition, 2013-2018





Partie 2. Une étude pour
chacun des pays (questions
demandées par Mélanie)

1.Les 20 pays pour lesquels la proportion de personnes sous-alimentées est la plus forte en 2017

01

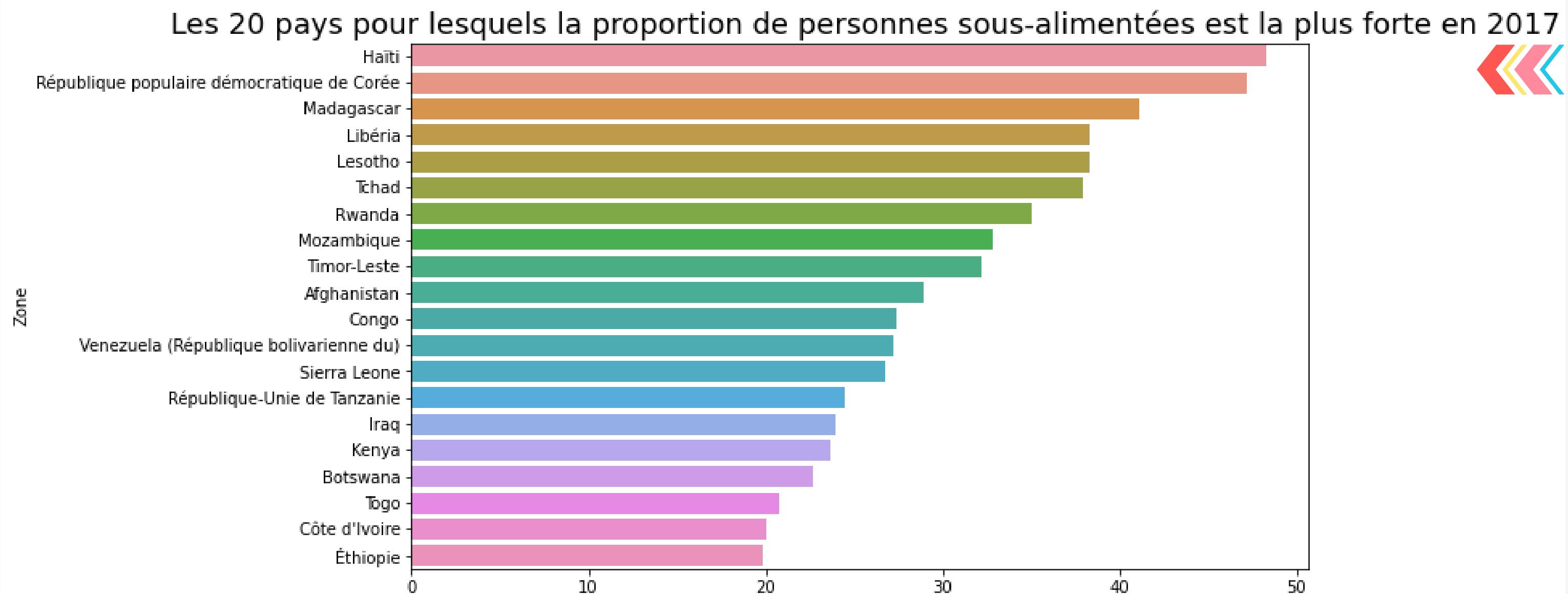
```
# Calculer la proportion de personne en état de sous-nutrition par pays:  
df_snu17['proportion'] = df_snu17['sous_nutrition_pop']/df_snu17['population']*100  
df_snu17.head()
```

02

```
# Faites la mise en ordre en fonction de 'proportion' et sélectionnez les 20 les plus  
df_snu_top = df_snu17.sort_values('proportion', ascending=False)[['Zone','proportion']].iloc[:20]
```

```
# Visualisation avec l'histogramme  
plt.figure(figsize=(10,6))  
sns.barplot(x=df_snu_top['proportion'],y=df_snu_top['Zone'])  
plt.title(' Les 20 pays pour lesquels la proportion de personnes sous-alimentées est la plus forte en 2017', fontsize = 18)
```

03



2. Les 15 pays qui ont le plus bénéficié d'aide depuis 2013

01

```
# La quantité qui a été donnée comme aide alimentaire, en tonnes, par pays
aide_valeur = aide_alimentaire.groupby('Pays bénéficiaire').sum()[['Valeur']]
aide_valeur = aide_valeur.rename(columns = {'Valeur': 'Valeur_tonnes'}) # renommer à 'Valeur_tonnes'

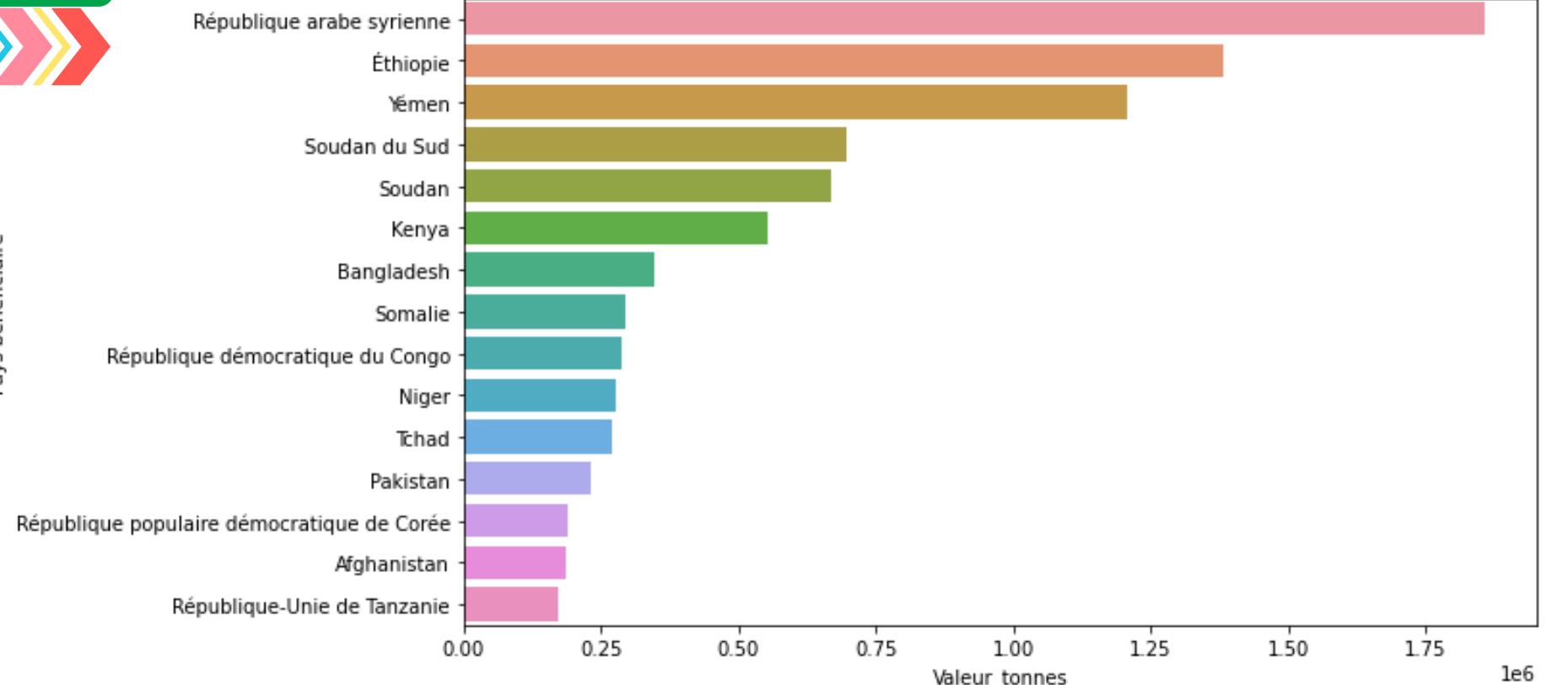
# Faites la mise en ordre en fonction de 'Valeur_tonnes' et sélectionnez les 15 plus
aide15 = aide_valeur.sort_values('Valeur_tonnes', ascending = False).iloc[:15]
aide15 = aide15.reset_index()
aide15
```

	Pays bénéficiaire	Valeur_tonnes
0	République arabe syrienne	1858943
1	Éthiopie	1381294
2	Yémen	1206484
3	Soudan du Sud	695248
4	Soudan	669784
5	Kenya	552836
6	Bangladesh	348188
7	Somalie	292678
8	République démocratique du Congo	288502
9	Niger	276344
10	Tchad	267966
11	Pakistan	231072
12	République populaire démocratique de Corée	187412
13	Afghanistan	185452
14	République-Unie de Tanzanie	172022

```
# Visualisation des 15 pays qui ont le plus bénéficié d'aide depuis 2013
plt.figure(figsize = (10,6))
sns.barplot(x = aide15['Valeur_tonnes'], y = aide15['Pays bénéficiaire'])
plt.title('Les 15 pays qui ont le plus bénéficié d'aide depuis 2013', fontsize = 25)
```

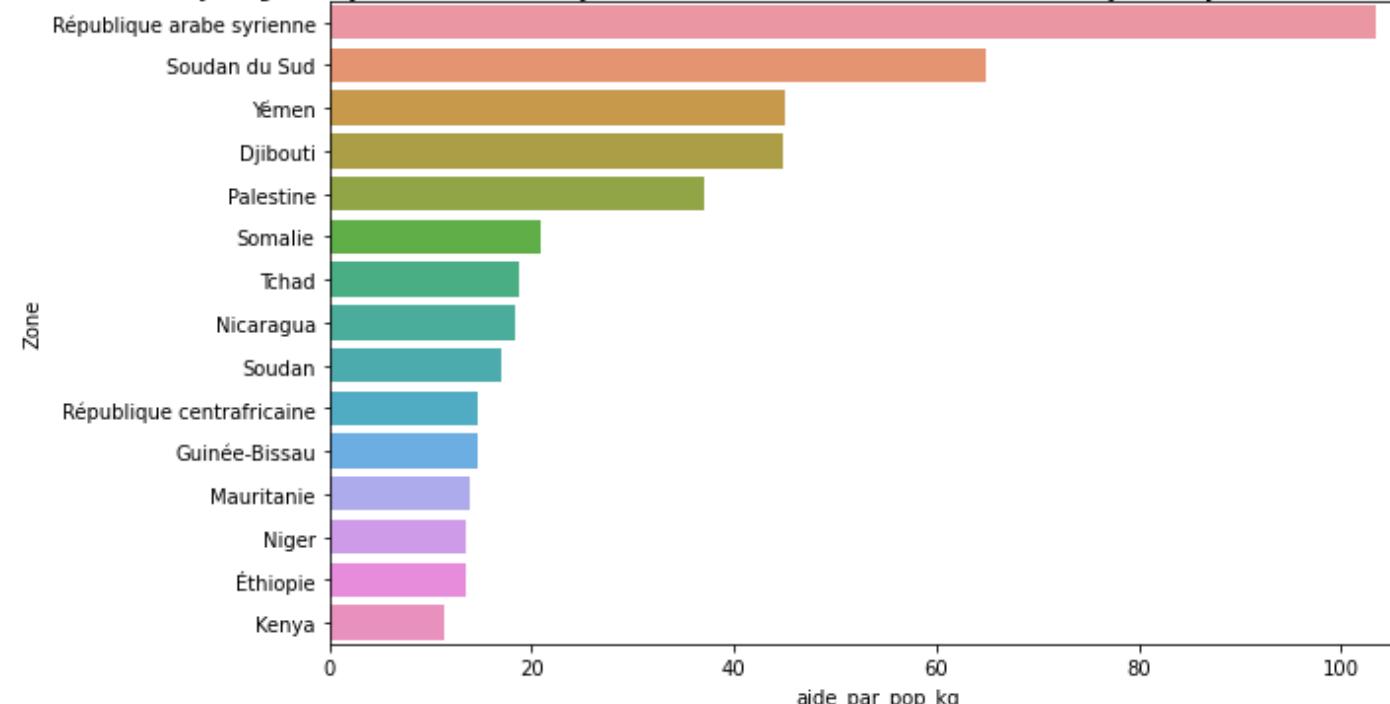
02

Les 15 pays qui ont le plus bénéficié d'aide depuis 2013



Analyse supplémentaire

Les 15 pays qui ont le plus bénéficié d'aide par personne depuis 2013



3 & 4. Les 10 pays avec la moins/la plus de disponibilité / habitant

01

3. Les 10 pays avec la moins de disponibilité/habitant

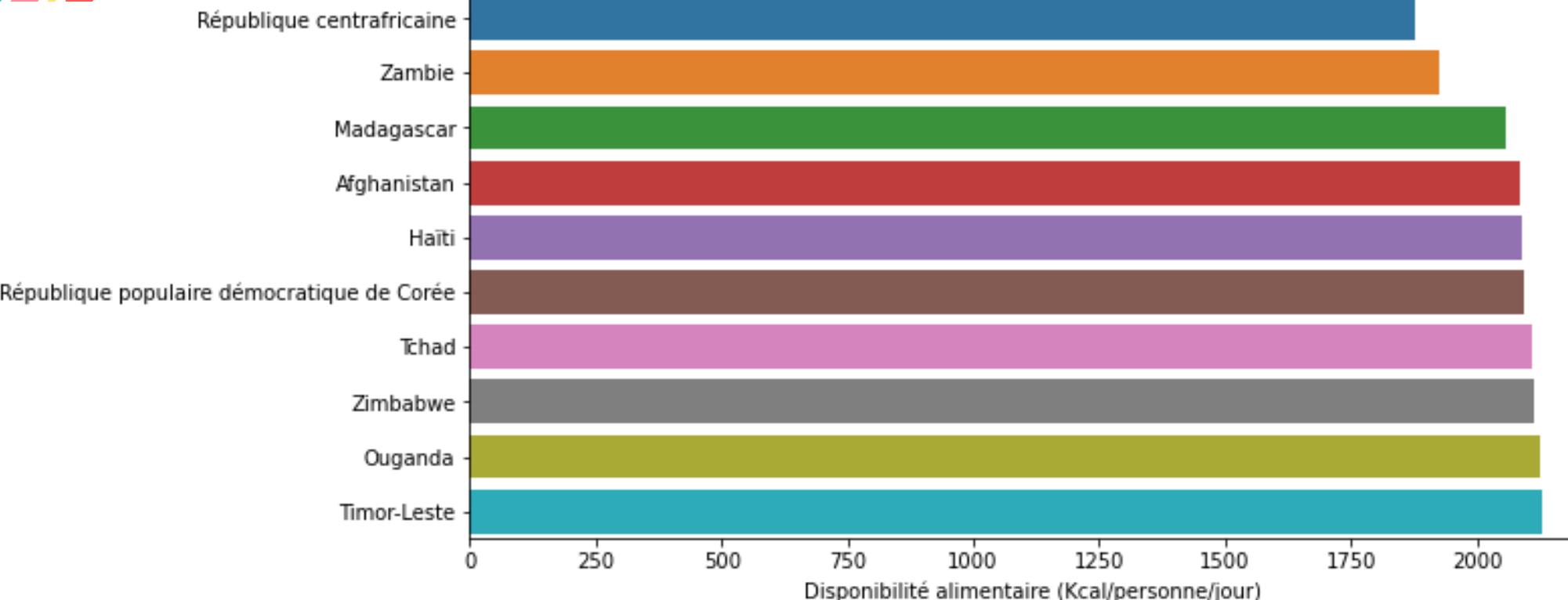
```
# Faire la mise en ordre en fonction de 'Disponibilité alimentaire (Kcal/personne/jour)' et sélectionner les 10 les moins
dispo_moin10 = df_dispo_final.sort_values(by='Disponibilité alimentaire (Kcal/personne/jour)').iloc[:10]
dispo_moin10
```

	Zone	Disponibilité alimentaire (Kcal/personne/jour)	Année	sous_pop_million	population_mille	sous_nutrition_pop	population	dispo_year
127	République centrafricaine	1879.0	2017	0.0	4596.023	0.0	4596023.0	3.152113e+12
164	Zambie	1924.0	2017	0.0	16853.599	0.0	16853599.0	1.183561e+13
91	Madagascar	2056.0	2017	10.5	25570.512	10500000.0	25570512.0	1.918914e+13
0	Afghanistan	2087.0	2017	10.5	36296.113	10500000.0	36296113.0	2.764875e+13
65	Haiti	2089.0	2017	5.3	10982.366	5300000.0	10982366.0	8.373889e+12
132	République populaire démocratique de Corée	2093.0	2017	12.0	25429.825	12000000.0	25429825.0	1.942699e+13
150	Tchad	2109.0	2017	5.7	15016.753	5700000.0	15016753.0	1.155967e+13
165	Zimbabwe	2113.0	2017	0.0	14236.595	0.0	14236595.0	1.097990e+13
114	Ouganda	2126.0	2017	0.0	41166.588	0.0	41166588.0	3.194486e+13
152	Timor-Leste	2129.0	2017	0.4	1243.258	400000.0	1243258.0	9.661171e+11

```
# Visualisation des 10 pays avec le moins de disponibilité/habitant
plt.figure(figsize=(10,5))
sns.barplot(x=dispo_moin10['Disponibilité alimentaire (Kcal/personne/jour)'], y=dispo_moin10['Zone'])
plt.title('Les 10 pays avec le moins de disponibilité/habitant', fontsize=18)
```



Les 10 pays avec le moins de disponibilité/habitant



02

4. Les 10 pays avec la plus de disponibilité/habitant

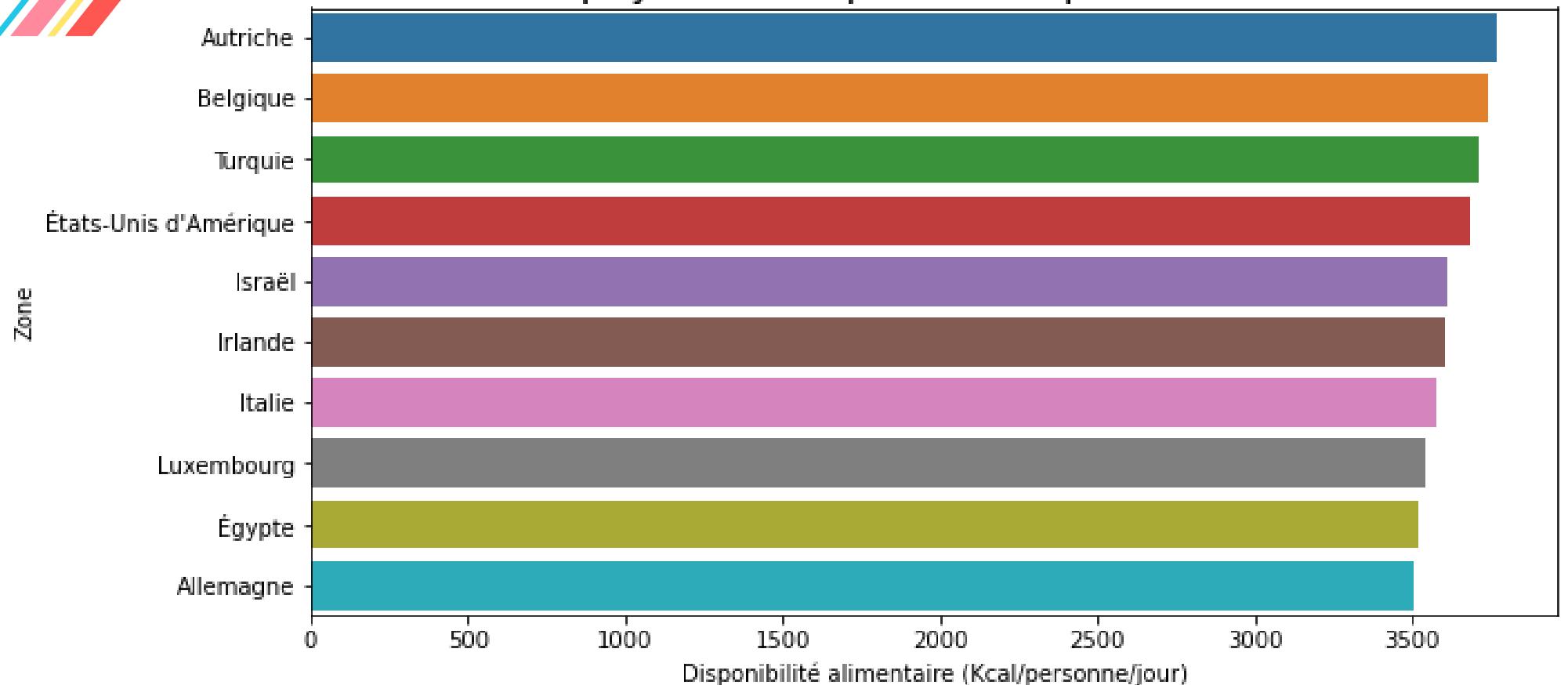
```
# Sélectionner les 10 pays les plus de disponibilité/habitant
dispo_top10 = df_dispo_final.sort_values(by='Disponibilité alimentaire (Kcal/personne/jour)', ascending=False).iloc[:10]
dispo_top10
```

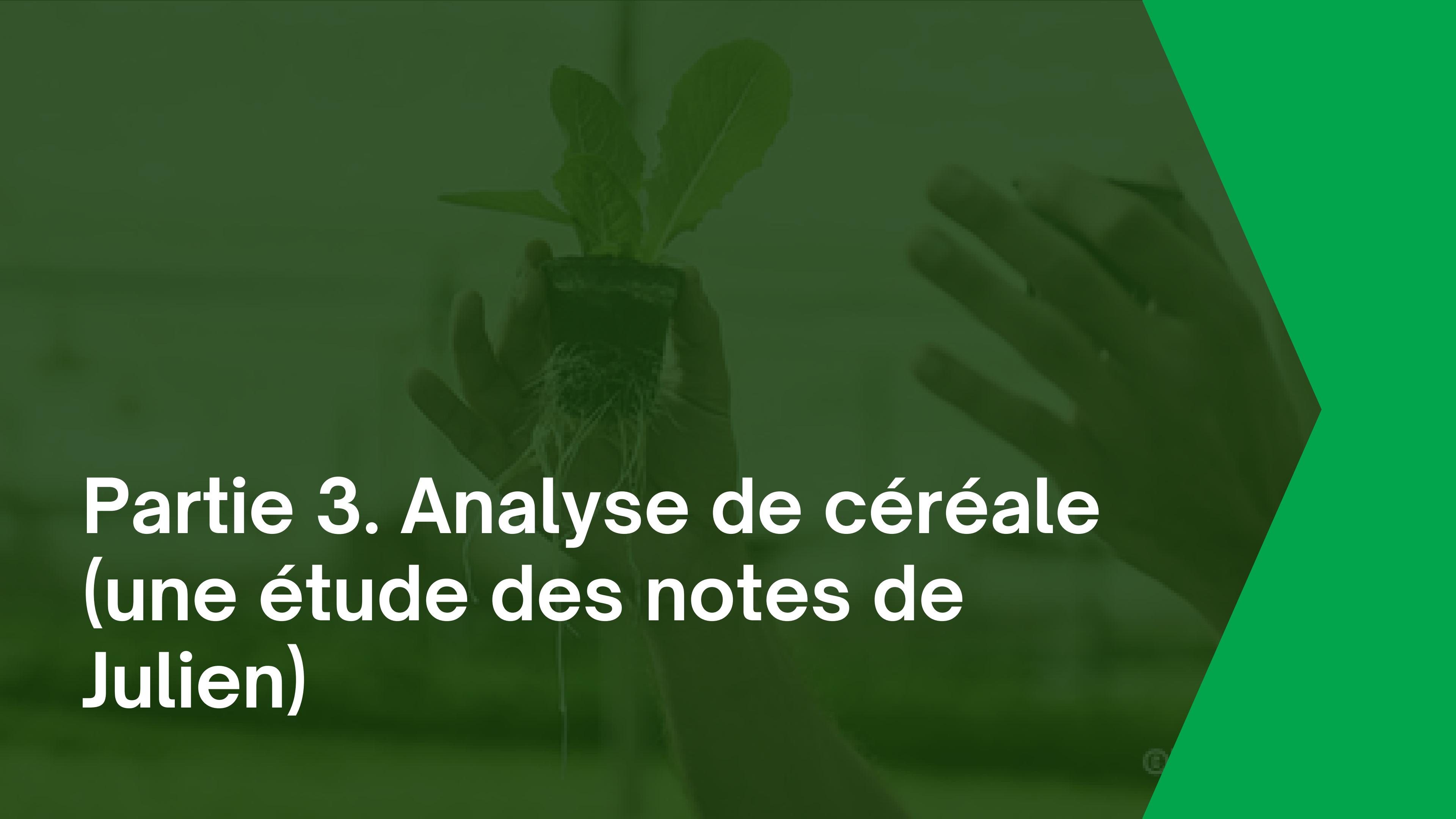
	Zone	Disponibilité alimentaire (Kcal/personne/jour)	Année	sous_pop_million	population_mille	sous_nutrition_pop	population	dispo_year
11	Autriche	3770.0	2017	0.0	8819.901	0.0	8819901.0	1.213662e+13
16	Belgique	3737.0	2017	0.0	11419.748	0.0	11419748.0	1.557659e+13
157	Turquie	3708.0	2017	0.0	81116.450	0.0	81116450.0	1.097846e+14
169	États-Unis d'Amérique	3682.0	2017	0.0	325084.756	0.0	325084756.0	4.368912e+14
74	Israël	3610.0	2017	0.0	8243.848	0.0	8243848.0	1.086251e+13
72	Irlande	3602.0	2017	0.0	4753.279	0.0	4753279.0	6.249278e+12
75	Italie	3578.0	2017	0.0	60673.701	0.0	60673701.0	7.923803e+13
89	Luxembourg	3540.0	2017	0.0	591.910	0.0	591910.0	7.648069e+11
166	Égypte	3518.0	2017	4.6	96442.591	4600000.0	96442591.0	1.238390e+14
4	Allemagne	3503.0	2017	0.0	82658.409	0.0	82658409.0	1.056866e+14

```
# Visualisation des 10 pays avec le plus de disponibilité/habitant
plt.figure(figsize=(10,5))
sns.barplot(x=dispo_top10['Disponibilité alimentaire (Kcal/personne/jour)'], y=dispo_top10['Zone'])
plt.title('Les 10 pays avec le plus de disponibilité/habitant', fontsize=18)
```



Les 10 pays avec le plus de disponibilité/habitant





Partie 3. Analyse de céréale (une étude des notes de Julien)

```
# Lister les 9 types de céréales
list_cereales = ['Blé', 'Seigle', 'Orge', 'Avoine', 'Autres']
list_cereales

['Blé',
 'Seigle',
 'Orge',
 'Avoine',
 'Mais',
 'Riz (Eq Blanchi)',
 'Sorgho',
 'Millet',
 'Céréales, Autres']
```

01

9 types de céréales: Blé, Seigle, Orge, Avoine, Maïs, Riz (Eq Blanchi), Sorgho, Millet, Autres Céréales

```
# Compter les sommes de chaque type d'utilisation
part_cereales = cereales_all[['Aliments pour animaux', 'Autres Utilisations', 'Nourriture', 'Pertes', 'Semences', 'Traitement']]
part_cereales
part_cereales.sum()
```

Aliments pour animaux	873535.0
Autres Utilisations	234787.0
Nourriture	1029010.0
Pertes	107120.0
Semences	68538.0
Traitement	94589.0
dtype:	float64

03

Compter les sommes de chaque type d'utilisation

# Selectionner les produits des céréales						
	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)
7	Afghanistan	Blé	vegetale	NaN	NaN	1369.0
12	Afghanistan	Céréales, Autres	vegetale	NaN	NaN	0.0
32	Afghanistan	Maïs	vegetale	200.0	NaN	21.0
34	Afghanistan	Millet	vegetale	NaN	NaN	3.0
40	Afghanistan	Orge	vegetale	360.0	NaN	26.0

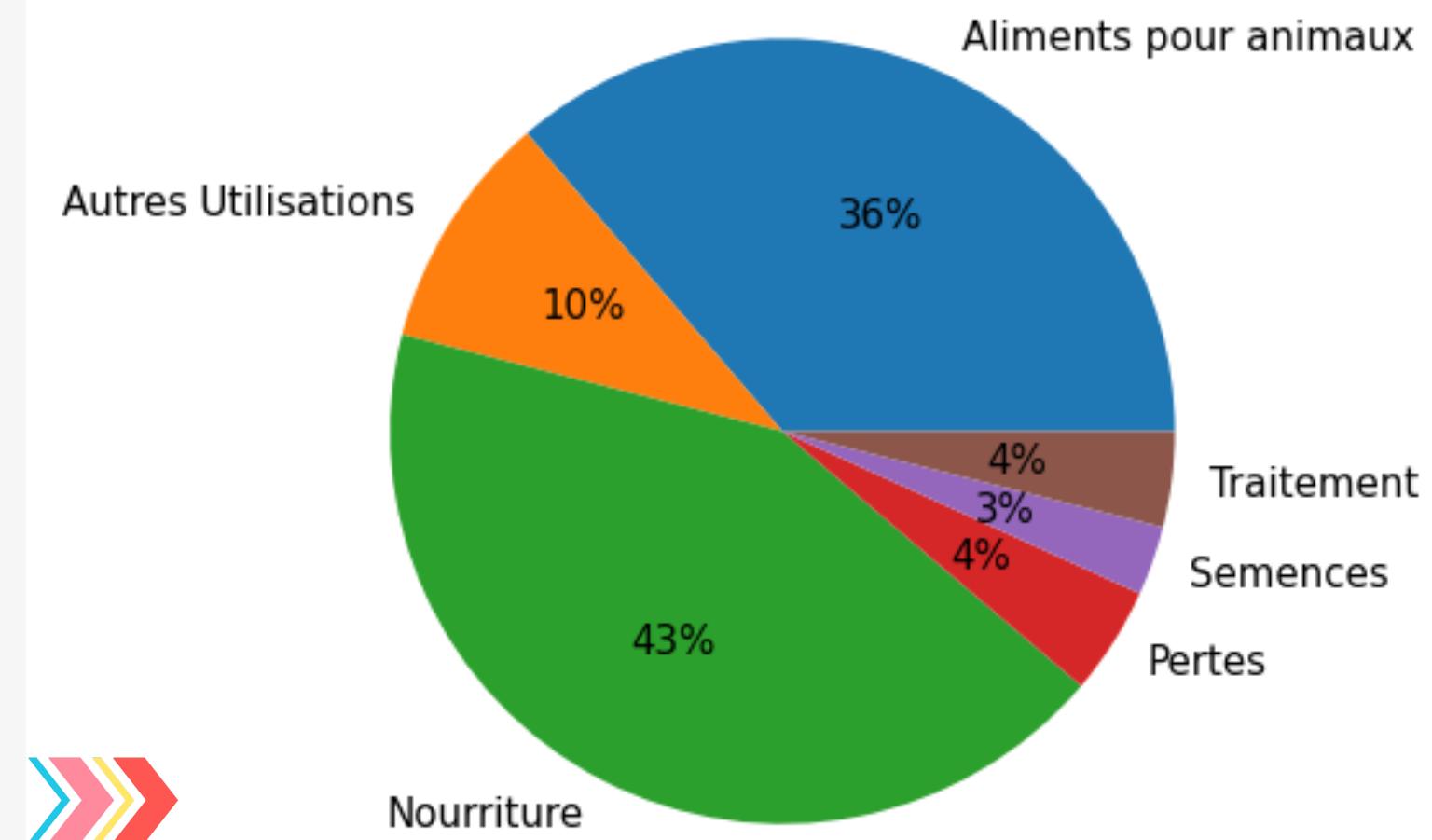
02

Selectionner les produits des céréales

```
# Visualisation avec le camembert
plt.figure(figsize=(15,7))
part_cereales.plot.pie(autopct='%.0f%%', fontsize = 15, ylabel = (''))
plt.title("La répartition de céréales", fontsize = 25)
```

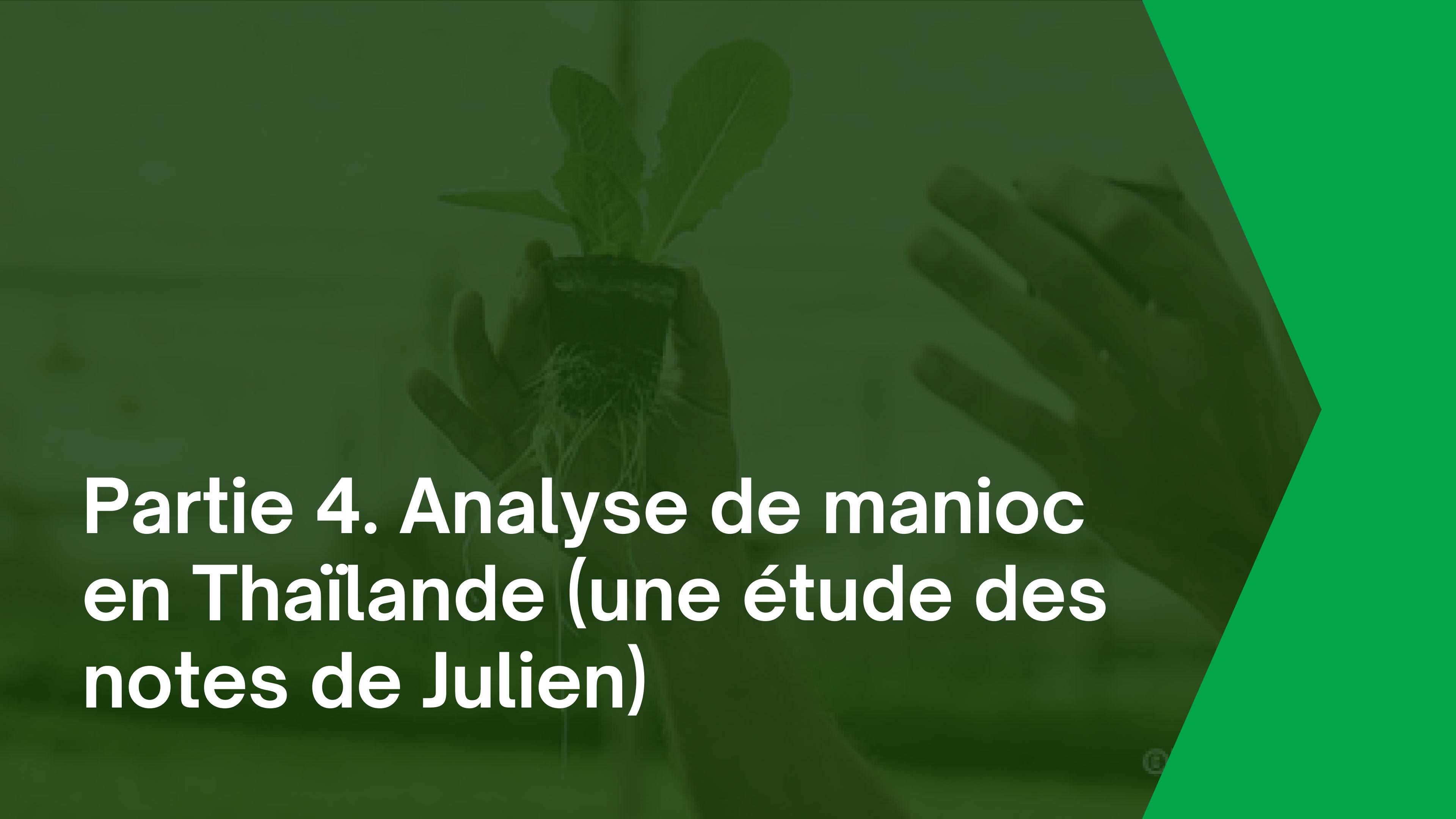
Text(0.5, 1.0, 'La répartition de céréales')

La répartition de céréales



04

Visualisation avec le camembert



Partie 4. Analyse de manioc en Thaïlande (une étude des notes de Julien)

01

Compter la proportion de population en état sous-nutrition en Thaïlande.

Compter la proportion de population en état sous-nutrition en Thaïlande

```
# Selectionner 'Thaïlande' dans 'df_snu17' et analyser les informations basiques
snu_thai = df_snu17.loc[df_snu17['Zone']=='Thaïlande', :]
```

Zone	Année	sous_pop_million	population_mille	sous_nutrition_pop	population	proportion
1114	Thaïlande	2017	6.2	69209.81	6200000.0	69209810.0

```
# La proportion de population en état sous-nutrition
proportion_thai_snu = round(snu_thai['proportion'].iloc[0],2)
print('La proportion de population en état sous nutrition en Thaïlande: ' + str(proportion_thai_snu) + '%')

La proportion de population en état sous nutrition en Thaïlande: 8.96%
```

02

La quantité totale vs. la quantité d'exportation de manioc

La quantité totale vs. la quantité d'exportation de manioc

```
# Selectionner l'info pour Manioc en Thaïlande
manioc = df_dispoa.loc[(df_dispoa['Zone']=='Thaïlande') & (df_dispoa['Produit'] == 'Manioc'), :]
```

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
13809	Thaïlande	Manioc	vegetale	1800.0	2081.0	40.0	13.0	0.05	0.14	6264.0

```
# La taux d'exportation pour manioc en Thaïlande
expo_manioc = round(manioc['Exportations - Quantité']/manioc['Production']*100,2)
print("La taux d'exportation pour manioc: " + str(expo_manioc.iloc[0]) + '%')
```

La taux d'exportation pour manioc: 83.41%

03

La population théorique nourries par manioc vs. la population Thai vs. la population sous-nutrition en Thaïlande.

Les besoins de quantité alimentaire théorique par personne par jour: 2.5kg

```
# Calculer la population théorique nourrie par manioc
manioc_pop = manioc['Exportations - Quantité']*1000000/(2.5*365)
manioc_pop = int(manioc_pop.iloc[0])
```

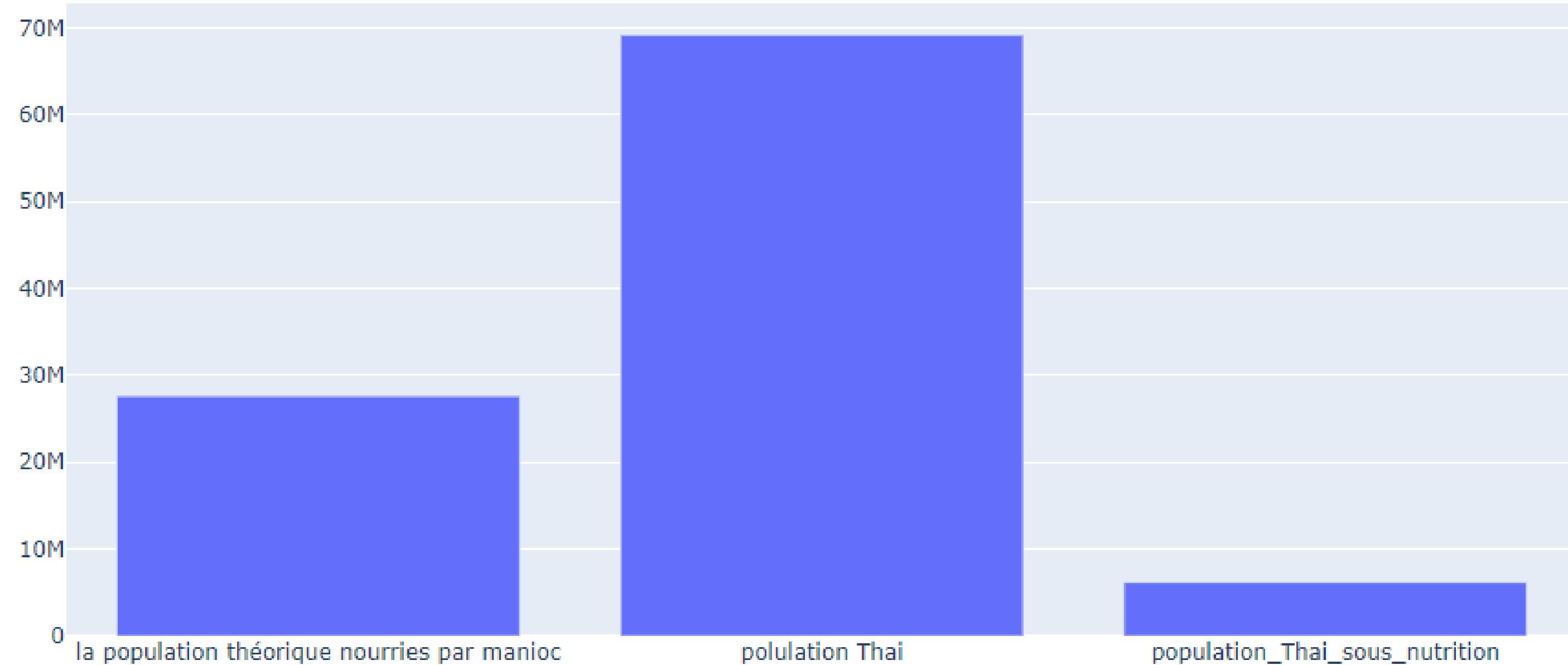
```
thai_pop = int(snu_thai.iloc[0,-2]) # la population Thai
thai_snu_pop = int(snu_thai.iloc[0,-3]) # la population sous nutrition
```

```
# Créer 'df_thai' pour les 3 populations calculées ci-dessus
thai_dict = {'Type of population':['la population théorique nourries par manioc', 'Population': [manioc_pop, thai_pop, thai_snu_pop]}
df_thai = pd.DataFrame(thai_dict)
df_thai
```

	Type of population	Population
0	la population théorique nourries par manioc	27631780
1	population Thai	69209810
2	population_Thai_sous_nutrition	6200000

```
# Visualization par l'histogramme
bar_thai = go.Bar(x= df_thai['Type of population'], y = df_thai['Population'])
layout_thai = go.Layout(title='La population théorique nourries par manioc, la population Thai et la population sous nutrition Thai')
fig_thai = go.Figure(data = bar_thai, layout = layout_thai)
fig_thai.show()
```

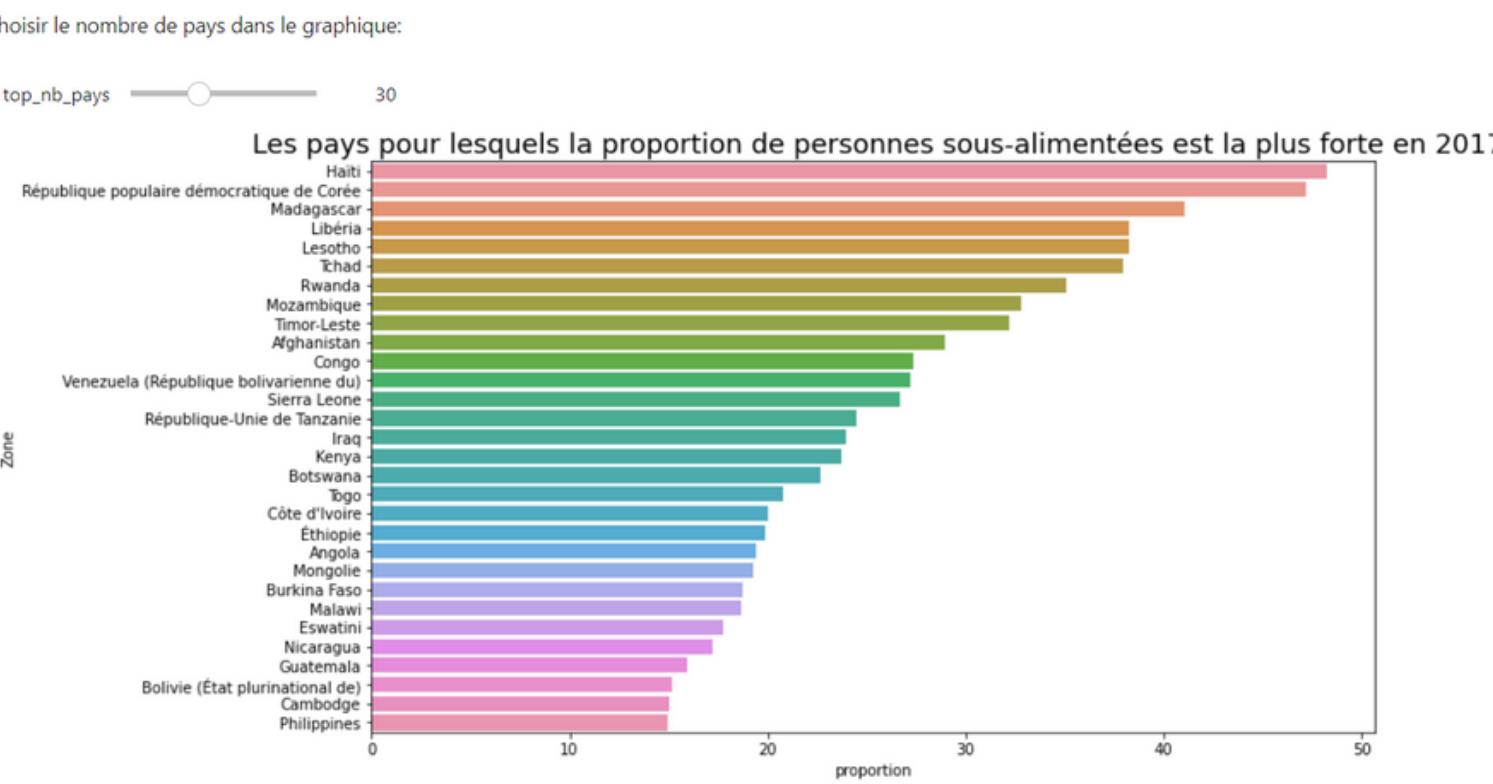
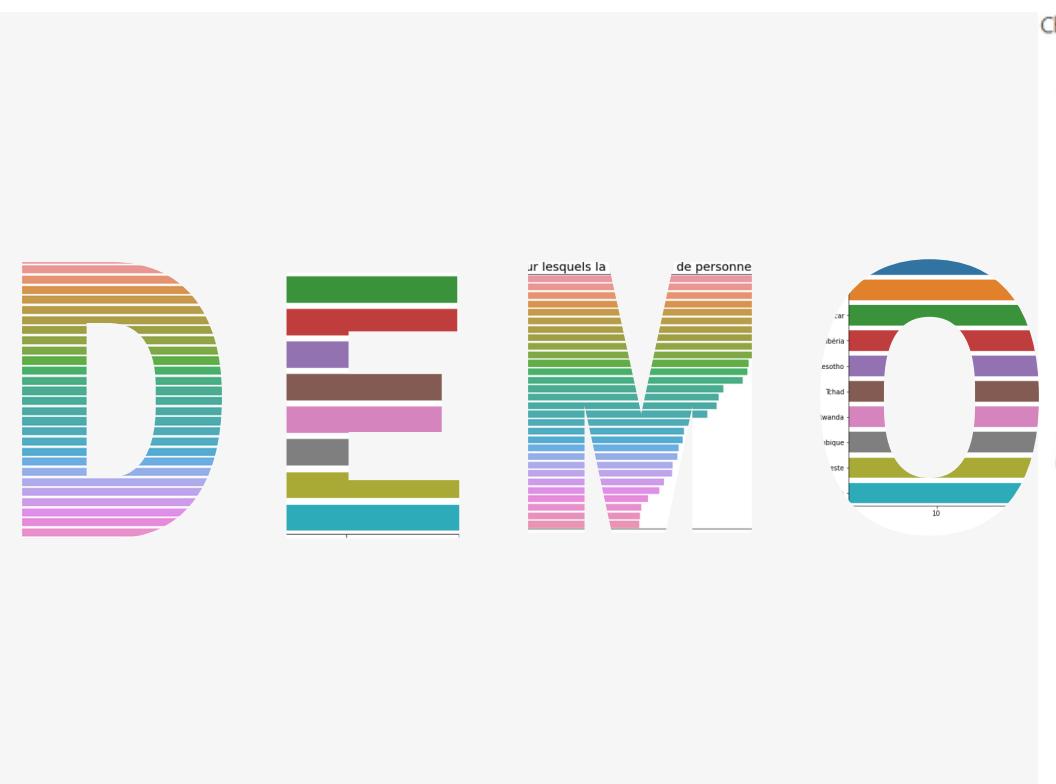
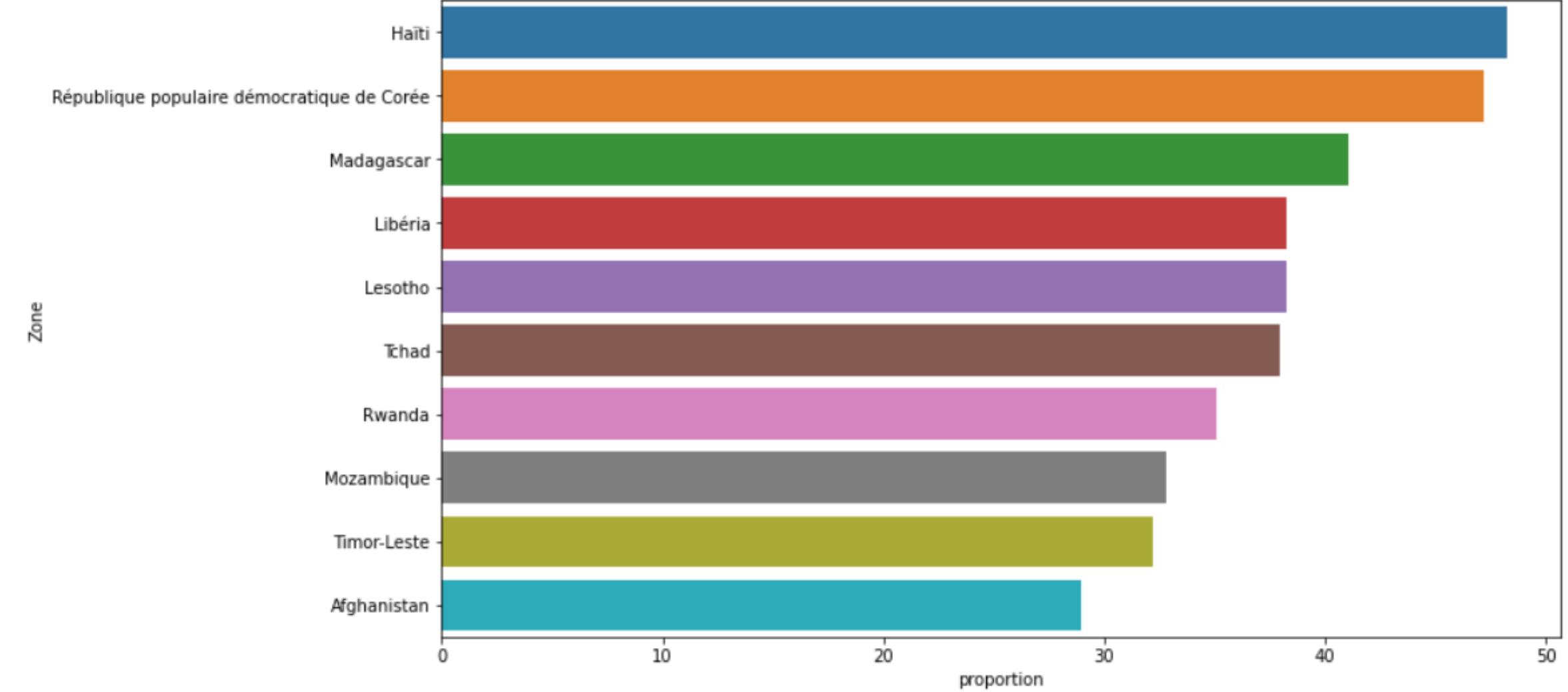
La population théorique nourries par manioc, la population Thai et la population sous nutrition Thai



Choisir les pays:

top_nb_pays

Les pays pour lesquels la proportion de personnes sous-alimentées est la plus forte en 2017



Partie supplémentaire. Graphique interactif avec Voilà et ipywidgets

Les pays pour lesquels la proportion de personnes sous-alimentées est la plus forte en 2017

```
import ipywidgets as widgets
from IPython.display import display
```

```
def snu_widget(top_nb_pays = 10):
    show_snu = df_snu_dsc.iloc[0:top_nb_pays]
    plt.figure(figsize=(12,7))
    sns.barplot(x=show_snu['proportion'],y=show_snu['Zone'])
    plt.title(' Les pays pour lesquels la proportion de personnes sous-alimentées est la plus forte en 2017')
```

Choisir le nombre de pays dans le graphique:

```
widgets.interact(snu_widget, top_nb_pays = (0, len(df_snu_dsc)))
```

MERCI.

