

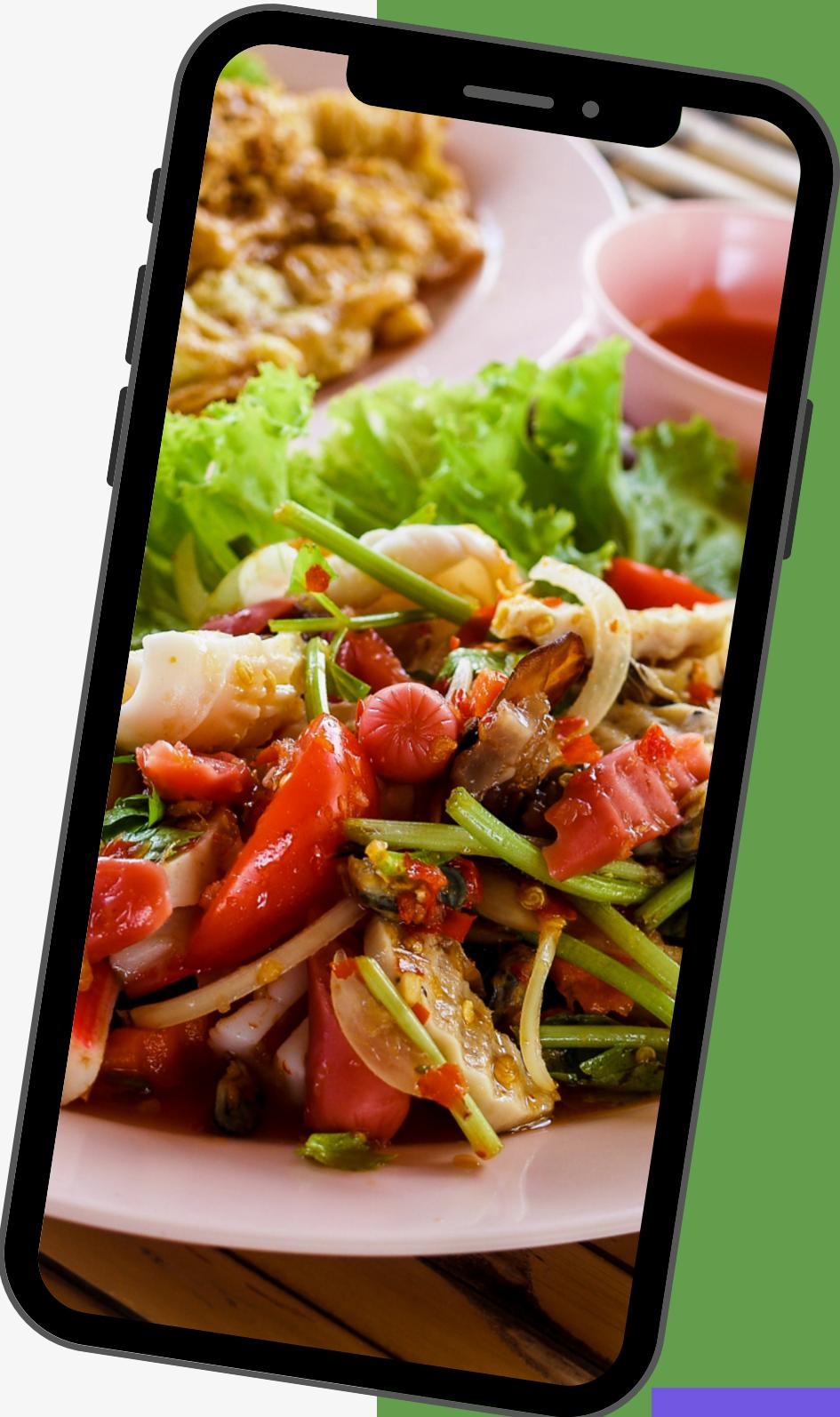


Projet 3

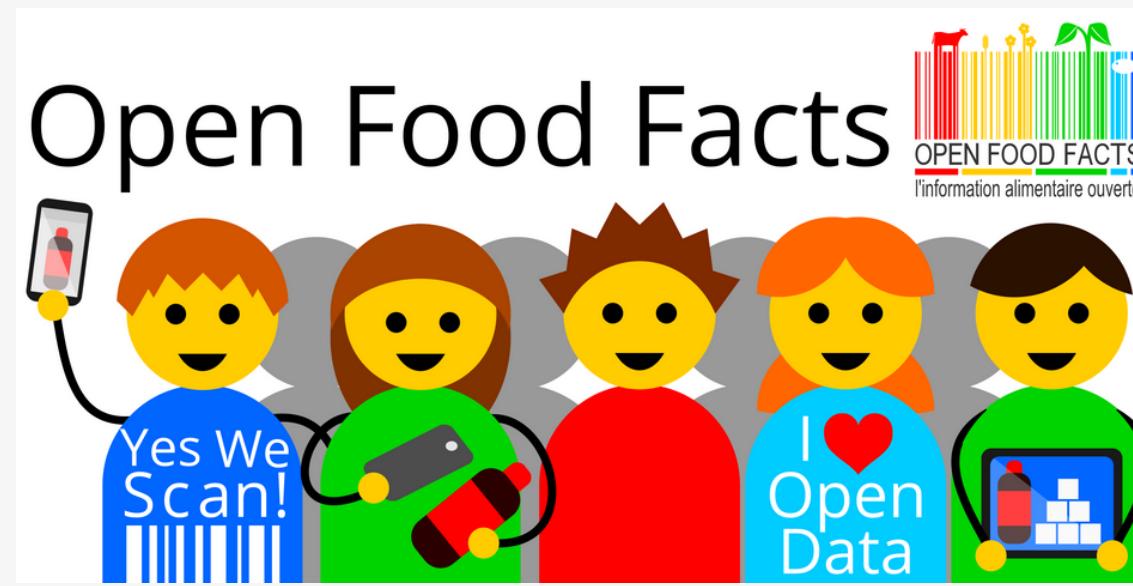
Concevez une application au service de la santé publique



YOUSRA AZZABI
03/10/2022



PLAN DE LA PRESENTATION



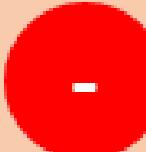
- 1 Idee d'application**
- 2 Description de la base de données**
- 3 Nettoyages des données**
- 4 Analyse exploratoire des variables importantes**
- 5 Analyse descriptive: ACP**
- 6 Conclusion**

1.Idée de l'application

Objectifs de l'application :

- permettre aux femmes enceintes de sélectionner des aliments adaptées(moins du sucre et moins de sel et avec un bon nutri-score)
- fournir pour chaque produit une information nutritionnelle

5 logos adaptés à la qualité nutritionnelle de chaque produit :



2. Description de la base de données

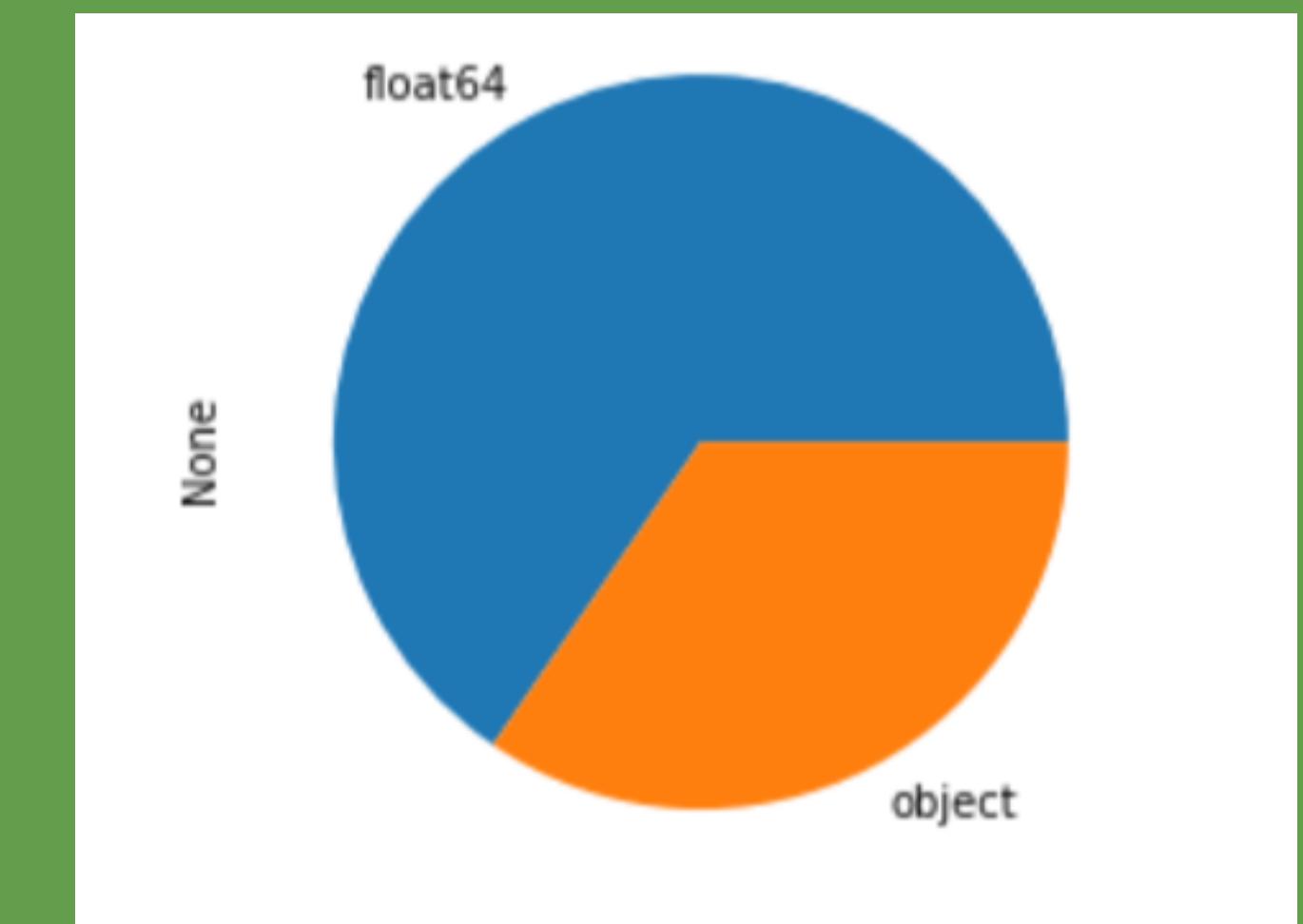
Lignes = 320772

Colonnes = 162

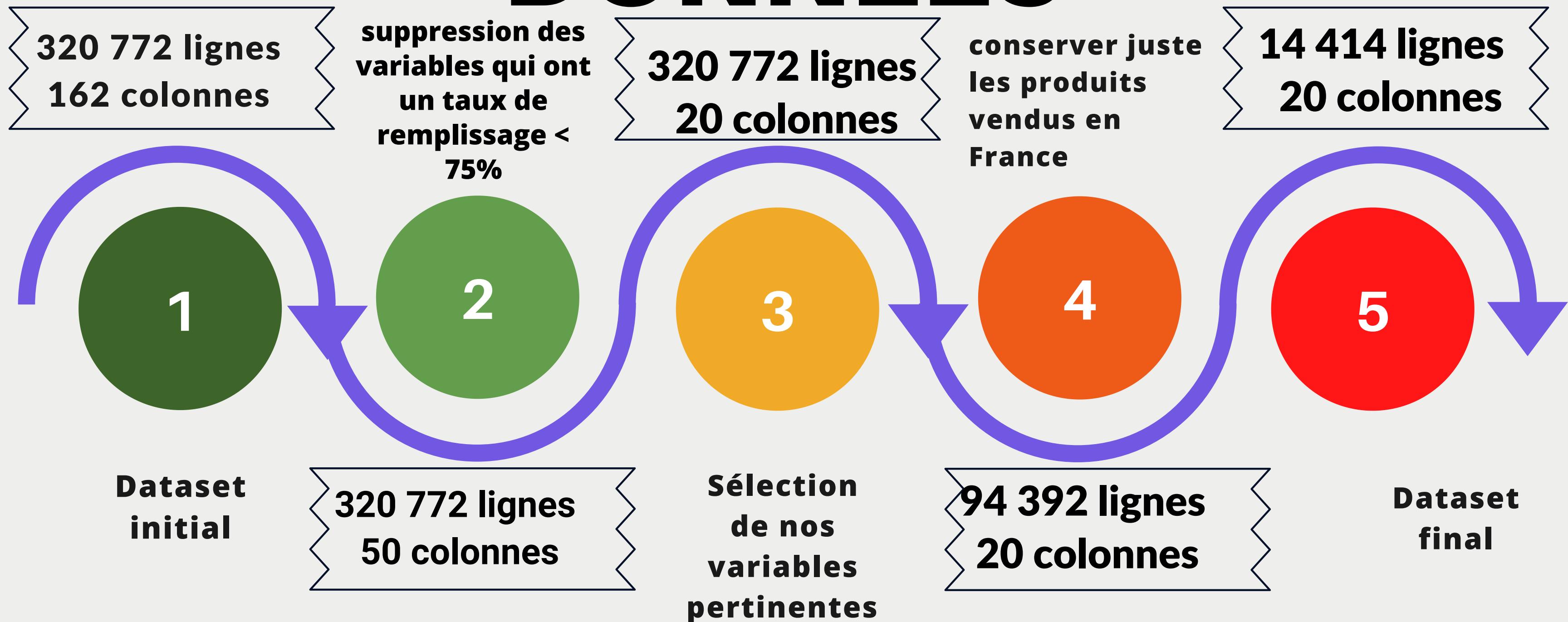
	code	url	creator	created_t	created_datetime	last_modified_t	last_modified_datetime
0	0000000003087	http://world-fr.openfoodfacts.org/produit/0000...	openfoodfacts-contributors	1474103866	2016-09-17T09:17:46Z	1474103893	2016-09-17T09:17:46Z
1	0000000004530	http://world-fr.openfoodfacts.org/produit/0000...	usda-ndb-import	1489069957	2017-03-09T14:32:37Z	1489069957	2017-03-09T14:32:37Z
2	0000000004559	http://world-fr.openfoodfacts.org/produit/0000...	usda-ndb-import	1489069957	2017-03-09T14:32:37Z	1489069957	2017-03-09T14:32:37Z
3	0000000016087	http://world-fr.openfoodfacts.org/produit/0000...	usda-ndb-import	1489055731	2017-03-09T10:35:31Z	1489055731	2017-03-09T10:35:31Z
4	0000000016094	http://world-fr.openfoodfacts.org/produit/0000...	usda-ndb-import	1489055653	2017-03-09T10:34:13Z	1489055653	2017-03-09T10:34:13Z

5 rows × 162 columns

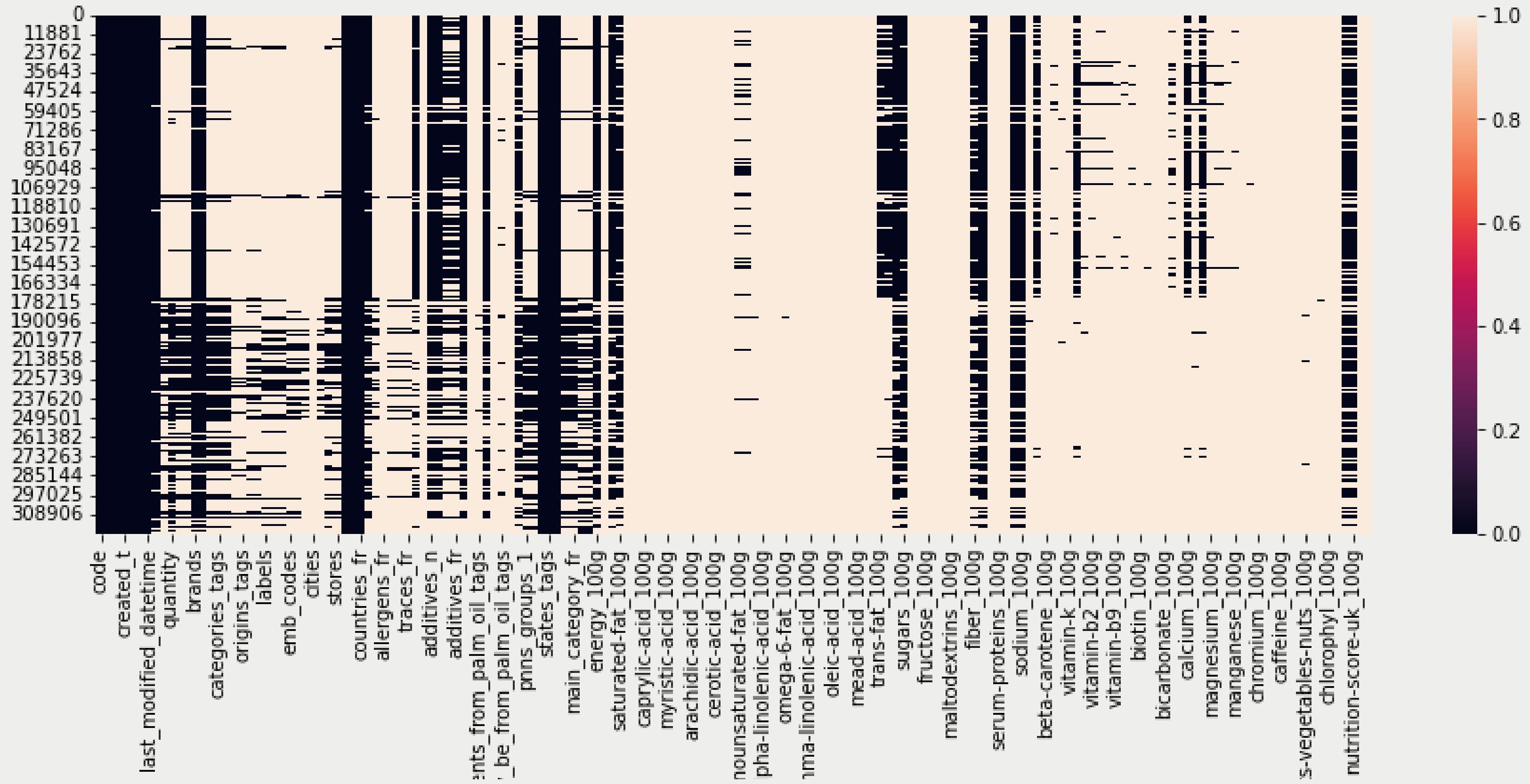
float64 = 106
object = 56



3. NETTOYAGE DES DONNEES



visualisation des valeurs manquantes



On remarque qu'il y a beaucoup des variables qui ne sont pas assez remplis donc on va supprimer les variables qui ont un taux de remplissage < 75%

```
def supprim_nan(df,pourcentage):  
    return df[df.columns[df.isna().sum()/df.shape[0] < pourcentage]]
```

```
data_foodfacts = supprim_nan(data_foodfacts, 0.75)  
data_foodfacts
```

```
data_foodfacts.shape
```

```
(320772, 50)
```

Sélection de nos variables pertinentes

```
# Choix de 20 variables
data_food_global = data_foodfacts.loc[:, ['code', 'last_modified_datetime', 'pnns_groups_1', 'saturated-fatty-acids_100g', 'product_name', 'brands', 'countries_fr', 'ingredients_text', 'serving_size', 'calcium_100g', 'nutrition_grade_fr', 'energy_100g', 'fat_100g', 'carbohydrates_100g', 'sugars_100g', 'fiber_100g', 'proteins_100g', 'salt_100g', 'sodium_100g', 'nutrition-score-fr_100g']]
```

```
data_food_global.shape
```

```
(320772, 20)
```

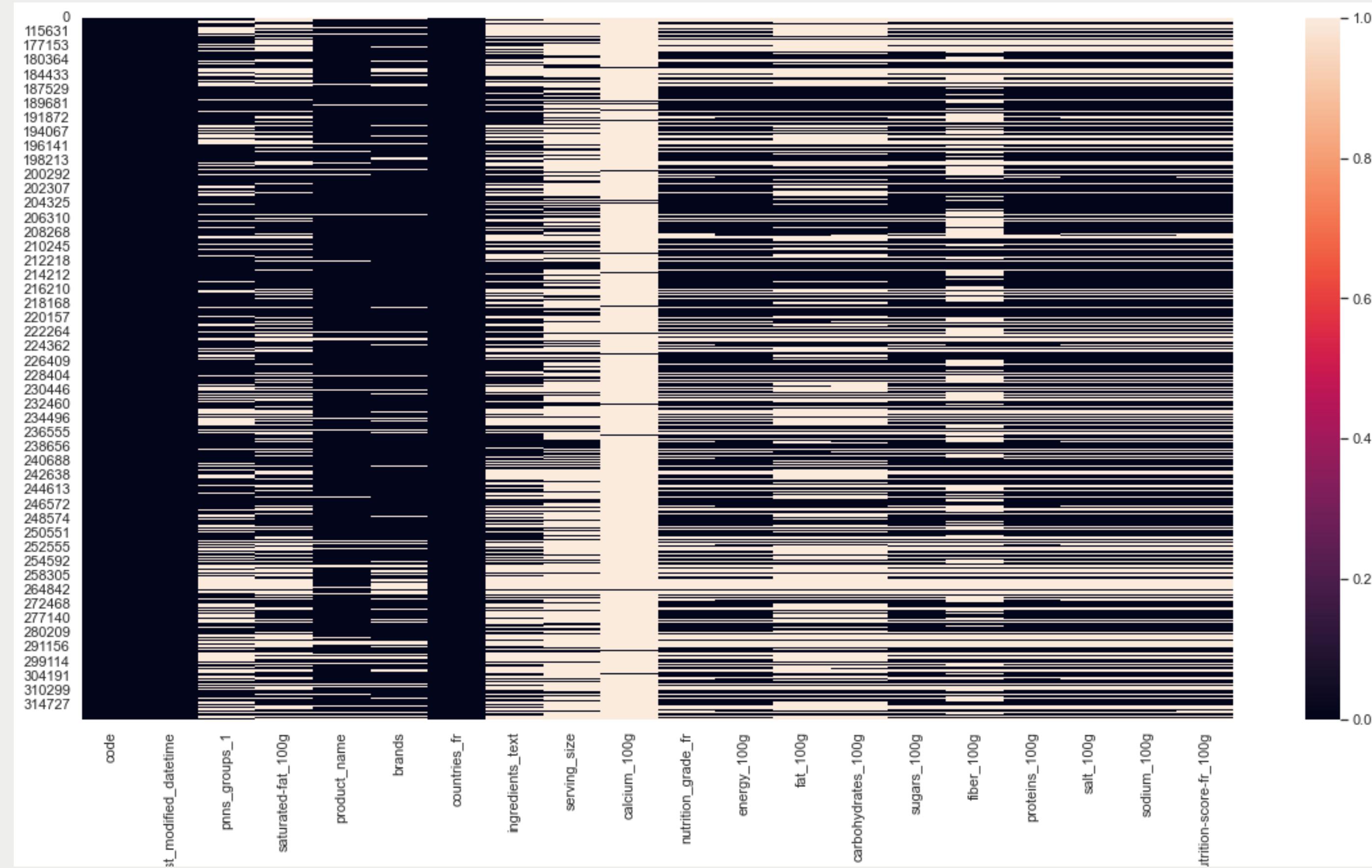
On va filtrer nos données pour conserver juste les produits vendus en France

```
produit_vendus_en_france= (data_food_global['countries_fr']=='France')
data_food_global[produit_vendus_en_france]
```

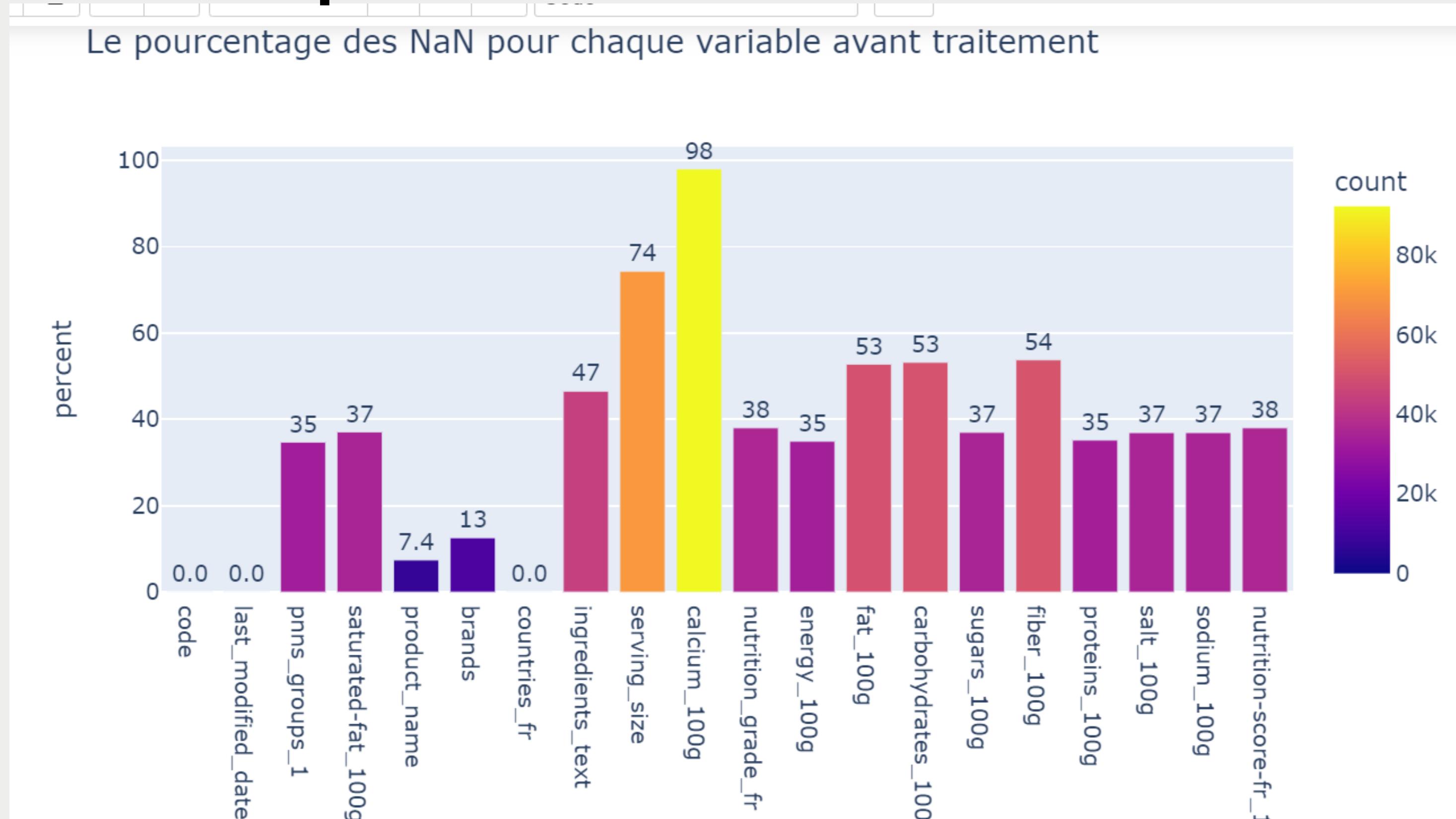
```
data_food_fr = data_food_global[produit_vendus_en_france]
data_food_fr.shape

(94392, 20)
```

Visualisation des valeurs manquantes après le filtrage des produits vendues en France et la suppression des valeurs qui ne sont pas assez remplis

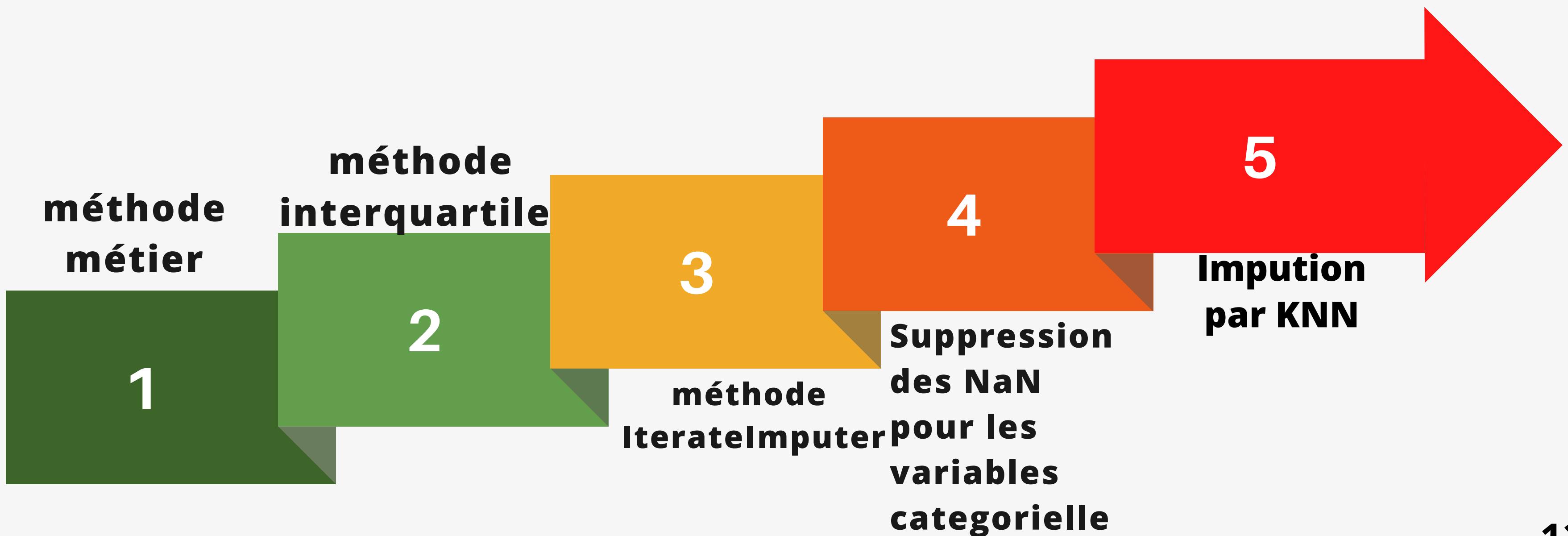


Histogramme qui représente le pourcentage des NaN pour chaque variable avant traitement



DETECTION DES VALEURS ABÉRRANTES

DETECTIONS DES VALEURS MANQUANTES



3.1.Détection et remplacement des valeurs abérantes par la méthode métier

Pour détecter les valeurs aberrantes on a définie une fonction qui prend en argument (dataframe et colonne)

```
def detect_outlier(dataframe, colonnes):

    for col in colonnes:
        # je calcule les valeurs abberantes avec la limite 100 et min = 0 pour tout les nutriment sauf l'energie
        nombre_aberrantes_sauf_energie = (dataframe[colonnes].drop(columns='energy_100g') > 100).sum() +
        #L'energie n'a pas la même limite que les autres, le max = 3700g et le min = 0
        nombre_aberrantes_sauf_energie['energy_100g'] = (dataframe['energy_100g'] < 0).sum() + (dataframe['energy_100g'] > 3700).sum()

    return nombre_aberrantes_sauf_energie
```

Pour remplacer les outliers détectées par la méthode métier aussi on a définie une fonction qui prend en argument notre dataframe et le colonne

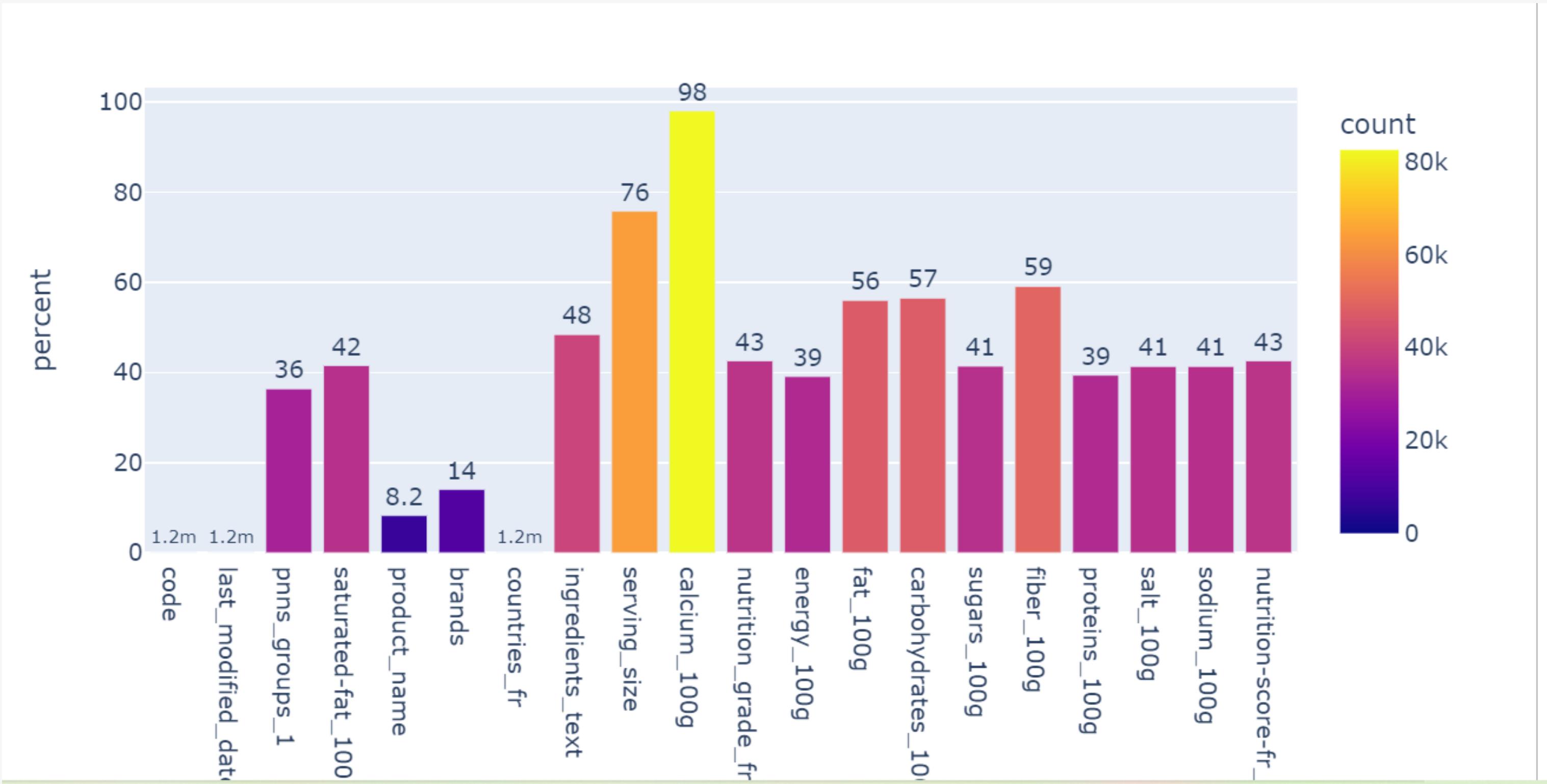
```
def suppre_outlier(dataframe, colonnes):
    for col in colonnes:
        if 'energy' not in col:
            dataframe.loc[dataframe[col] < 0] = np.nan
            dataframe.loc[dataframe[col] > 100] = np.nan

        else:

            dataframe.loc[dataframe[col] < 0] = np.nan
            dataframe.loc[dataframe[col] > 3700] = np.nan

    return dataframe
```

On remarque que le pourcentage des nan a augmenter



3.2.Détection et remplacement des valeurs abérantes par la méthode interquartille

Pour détecter les valeurs aberrantes par la méthode interquartile on fait appel a l'algorithme

```
def detect_outlier_interq(dataframe, colonnes):

    for col in colonnes:
        Q1 = dataframe[col].quantile(0.25)
        Q3 = dataframe[col].quantile(0.75)
        IQR = Q3-Q1
        Limite_inf = Q1 - (1.5 * IQR)
        Limite_sup = Q3 + (1.5 * IQR)
        nombre_aberrantes = (dataframe[colonnes] < Limite_inf).sum() + (dataframe[colonnes] > Limite_sup)

    return nombre_aberrantes
```

Expliquons ici la méthode interquartile

```
#Remplacement des outlier par NaN par La methode interquartile
def delete_outlier_quartile(dataframe,colonnes):

    for col in colonnes:
        Q1 = dataframe[col].quantile(0.25)
        Q3 = dataframe[col].quantile(0.75)
        IQR = Q3-Q1
        Limite_inf = Q1 - (1.5 * IQR)
        Limite_sup = Q3 + (1.5 * IQR)
        dataframe.loc[dataframe[col] < Limite_inf] = np.nan
        dataframe.loc[dataframe[col] > Limite_sup] = np.nan

    return(dataframe)
```

L'IQR est utilisé pour mesurer la variabilité en divisant un ensemble de données en quartiles. Les données sont triées par ordre croissant et divisées en 4 parties égales. Q1, Q2, Q3 appelés premier, deuxième et troisième quartiles sont les valeurs qui séparent les 4 parties égales.

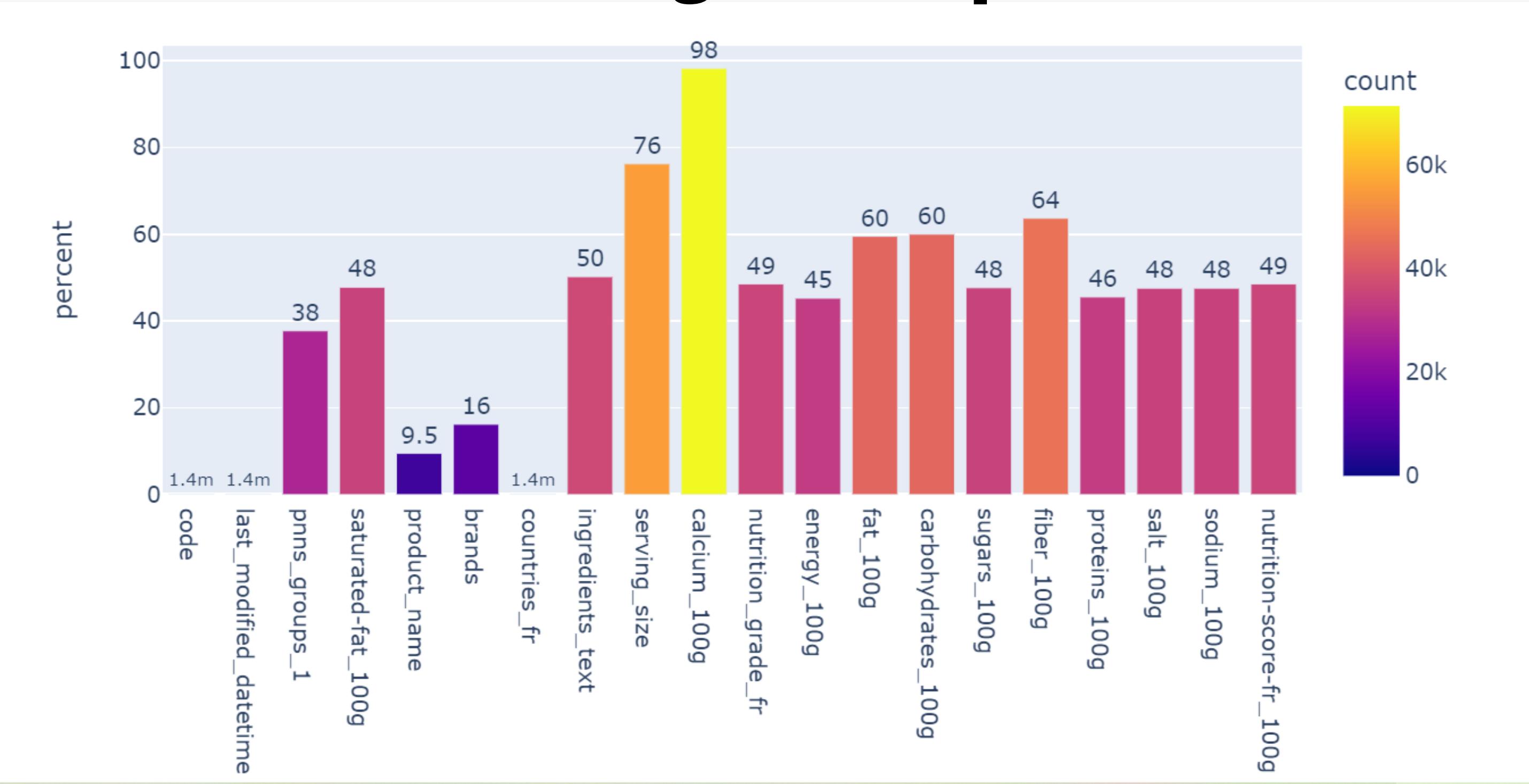
Q1 représente le 25e centile des données. Q2 représente le 50e centile des données. Q3 représente le 75e centile des données.

Si un jeu de données a $2n / 2n+1$ points de données, alors Q1 = médiane du jeu de données. Q2 = médiane des n plus petits points de données. Q3 = médiane des n points de données les plus élevés.

L'IQR est l'intervalle entre le premier et le troisième quartile à savoir Q1 et Q3 : $IQR = Q3 - Q1$. Les points de données qui tombent en dessous de $Q1 - 1,5 \text{ IQR}$ ou au-dessus de $Q3 + 1,5 \text{ IQR}$ sont des valeurs aberrantes

Les points de données trouvés 1.5 fois IQR au-dessus de Q3 et en dessous de Q1 sont des valeurs aberrantes

Le pourcentage des NaN pour chaque variable après filtrage interquartile



On observe une augmentation de NaN

Comparaison de deux méthodes

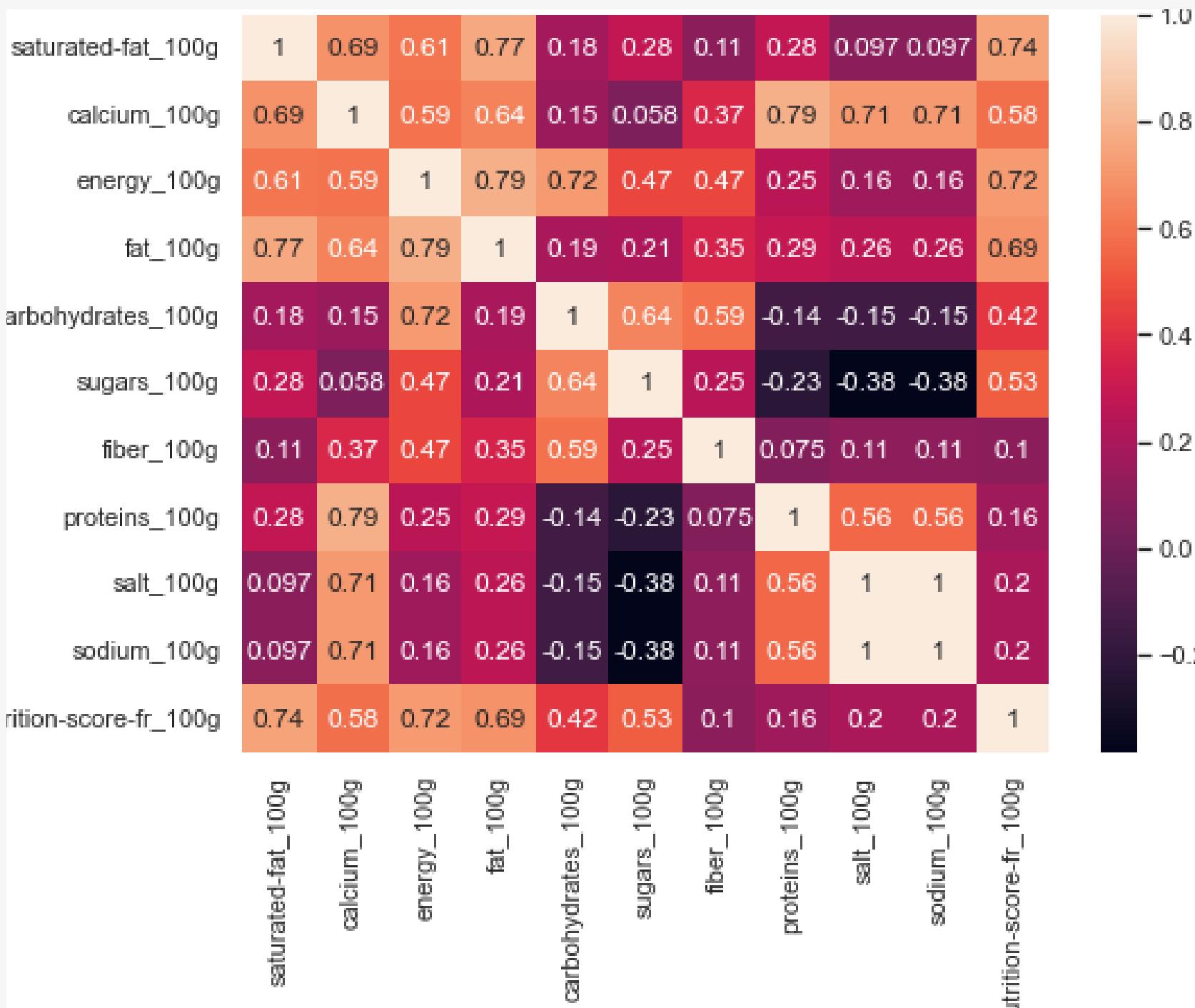
```
# Pourcentage de valeur aberrantes detecter par le metier  
Data_per_outlier = nombre_valeur_aberrantes*100/data.shape[0]  
Data_per_outlier  
  
saturated-fat_100g          0.001059  
calcium_100g                 0.000000  
fat_100g                     0.002119  
carbohydrates_100g           0.008475  
sugars_100g                   0.005297  
fiber_100g                    0.001059  
proteins_100g                 0.000000  
salt_100g                     0.002119  
sodium_100g                   0.000000  
nutrition-score-fr_100g       10.442622  
energy_100g                   0.268031  
dtype: float64
```

```
(nombr_outlier_interquartile*100/data.shape[0]).sort_values()  
  
nutrition-score-fr_100g      0.001059  
calcium_100g                  0.002119  
sodium_100g                   0.045555  
fiber_100g                     0.061446  
salt_100g                      0.206585  
proteins_100g                  0.243665  
saturated-fat_100g            0.618696  
fat_100g                       3.321256  
sugars_100g                     7.889440  
carbohydrates_100g             13.579541  
energy_100g                     53.383761  
dtype: float64
```

En calculant la limite inf et sup de chaque variable (méthode inetrquartile) on remarque que ecarbohydrates_100g
Limite_inf -64.74999999999999
Limite_sup 120.44999999999999
alors qu'elle ne doit pas dépasser 100 en limite sup.
On peut donc conclure que cette méthode n'est pas précise et je continuera mon analyse avec la méthode métier

3.3.Traitement des valeurs manquantes par la méthode iteratelImputer

La methode IteratelImputer se base sur les variables corrélées entre eux



On remarque ici que :

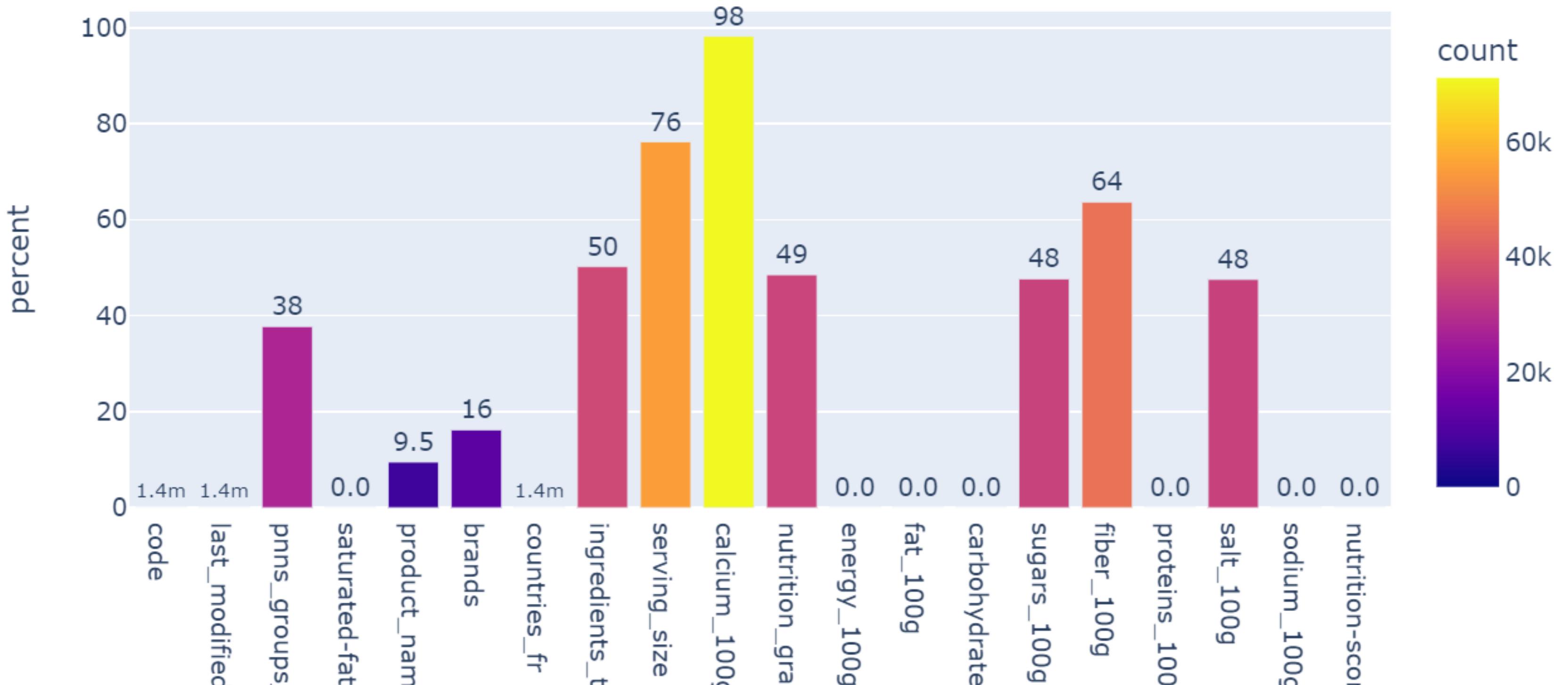
- La variables energy est fortement corrélée avec : saturated, fat, carbohydrate, et nutriscore
- La variable fat_100 est fortement corrélée avec : saturated, energiy, et nutriscore
- La variable carbohydrate est fortement corrélée avec: sucre
- La variable surce est fortement corrélée avec: carbohydrate, nutriscore,calcium
- La variable nutriscore est fortement corrélée avec: satured, energy, fat, sucre,calcium
- la variable calcium est fortement corrélée avec: nutriscore,

```

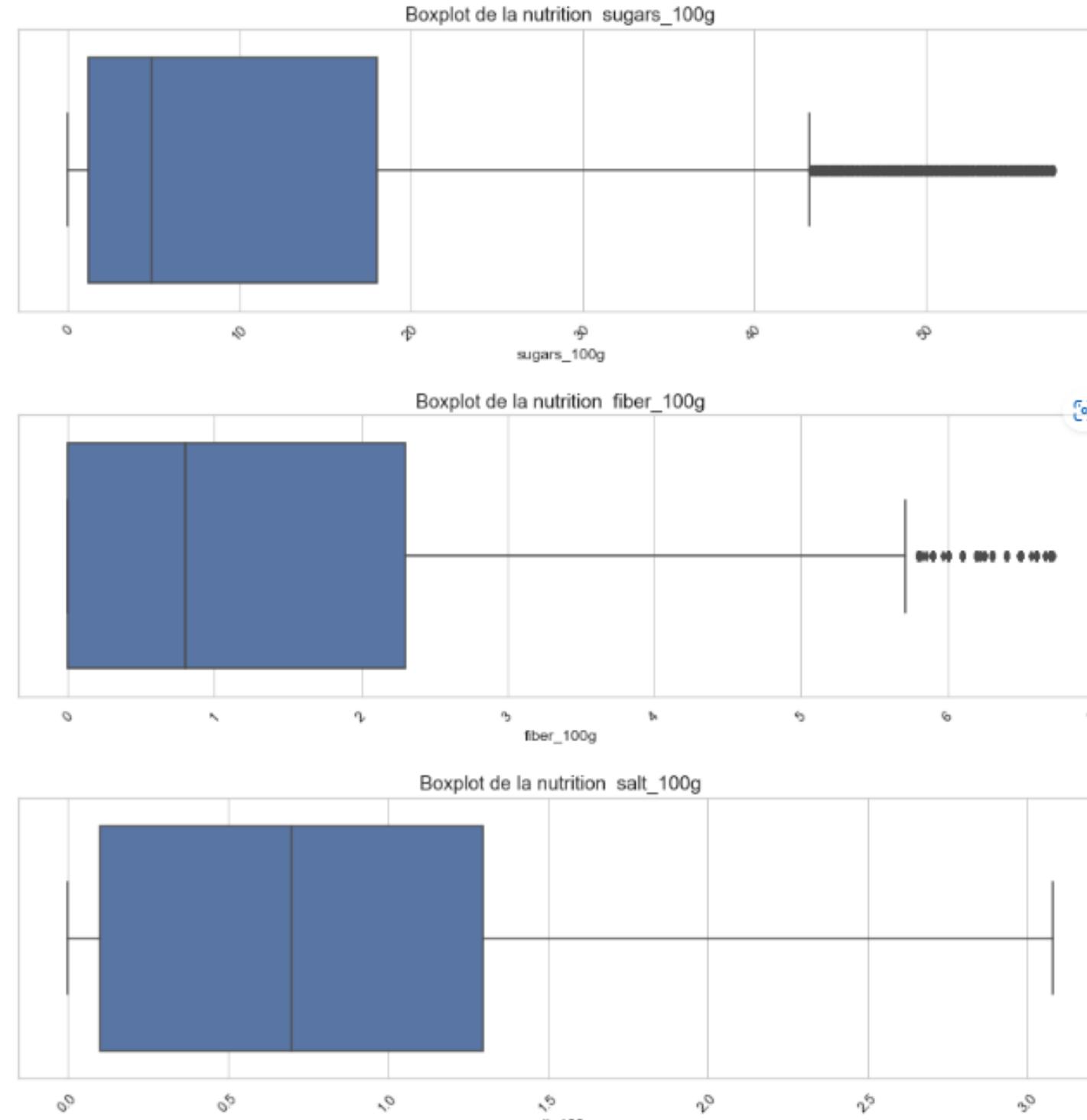
: colum corre = ['fat_100g', 'saturated-fat_100g', 'energy_100g', 'proteins_100g', 'carbohydrates_100g', 'so
imputerimputer = IterativeImputer()
data_sans_outlier[colum corre] = imputerimputer.fit_transform(data_sans_outlier[colum corre])

```

Pourcentage des NaN après filtrage interquartile



Remplacement des NaN par la médiane des variables



Comme la valeur moyenne est fortement influencée par les valeurs aberrantes, il est recommandé de remplacer es valeurs aberrantes par la valeur mediane

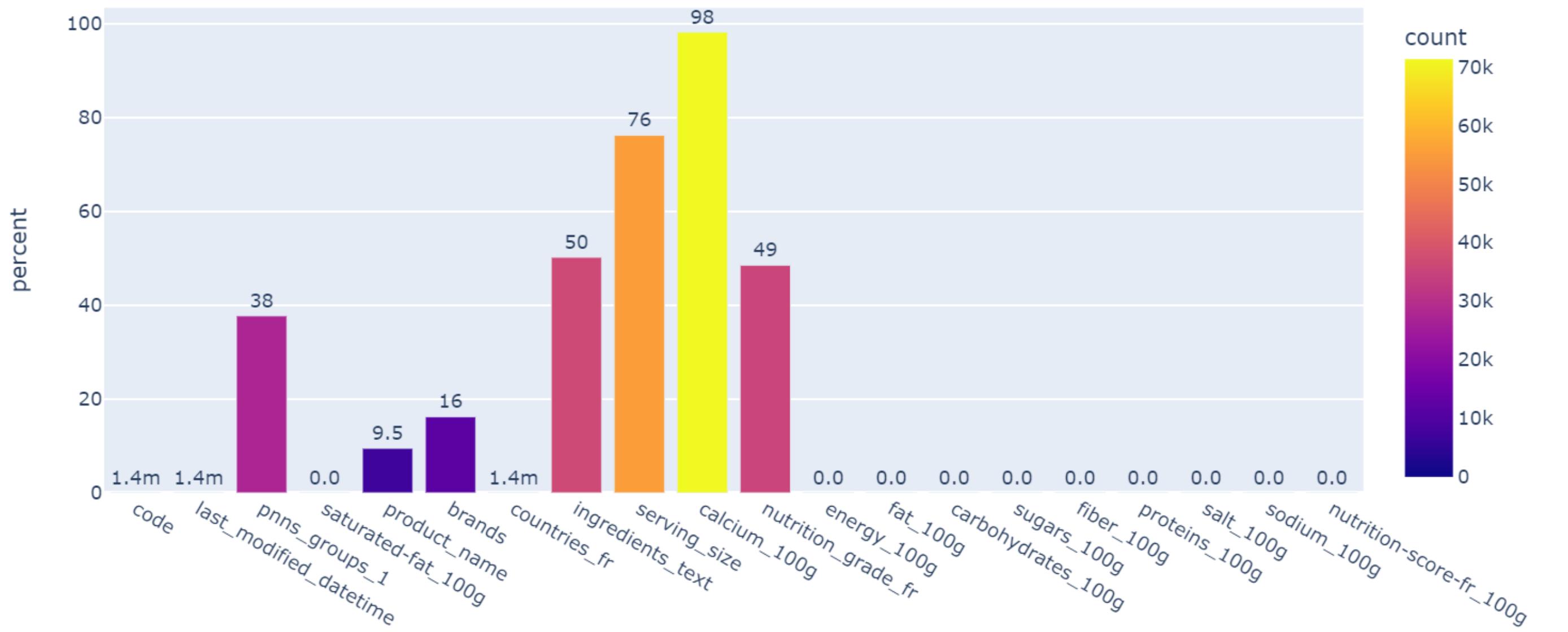
On observe le boxplot du sel, sucre, et fibre , on observe que la mediane n'est pas nul alors on remplace les NaN de :

- sel,
- sucre
- fibre

par leur median

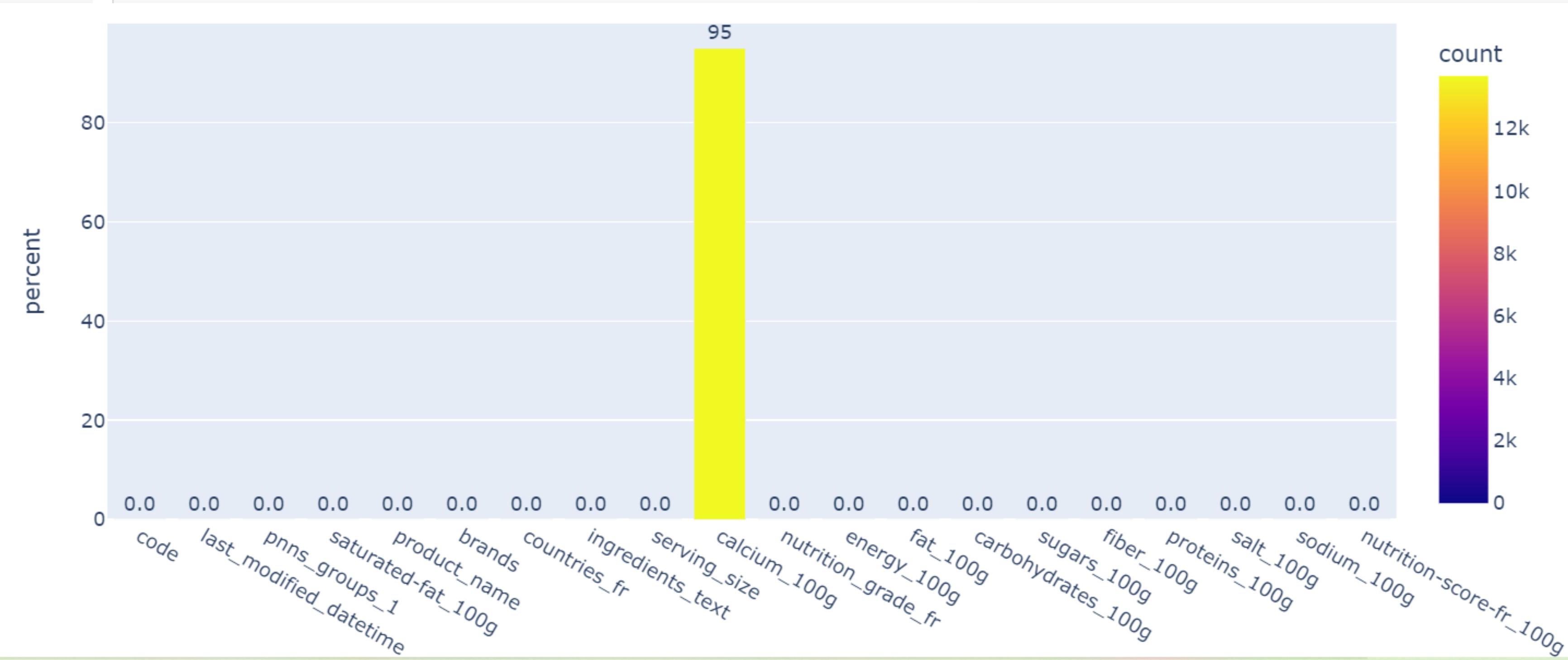
```
: data_sans_outlier[Nutritive] = data_sans_outlier[Nutritive].fillna(data_sans_outlier[Nutritive].median())
```

Remplacement des NaN par le median des variables



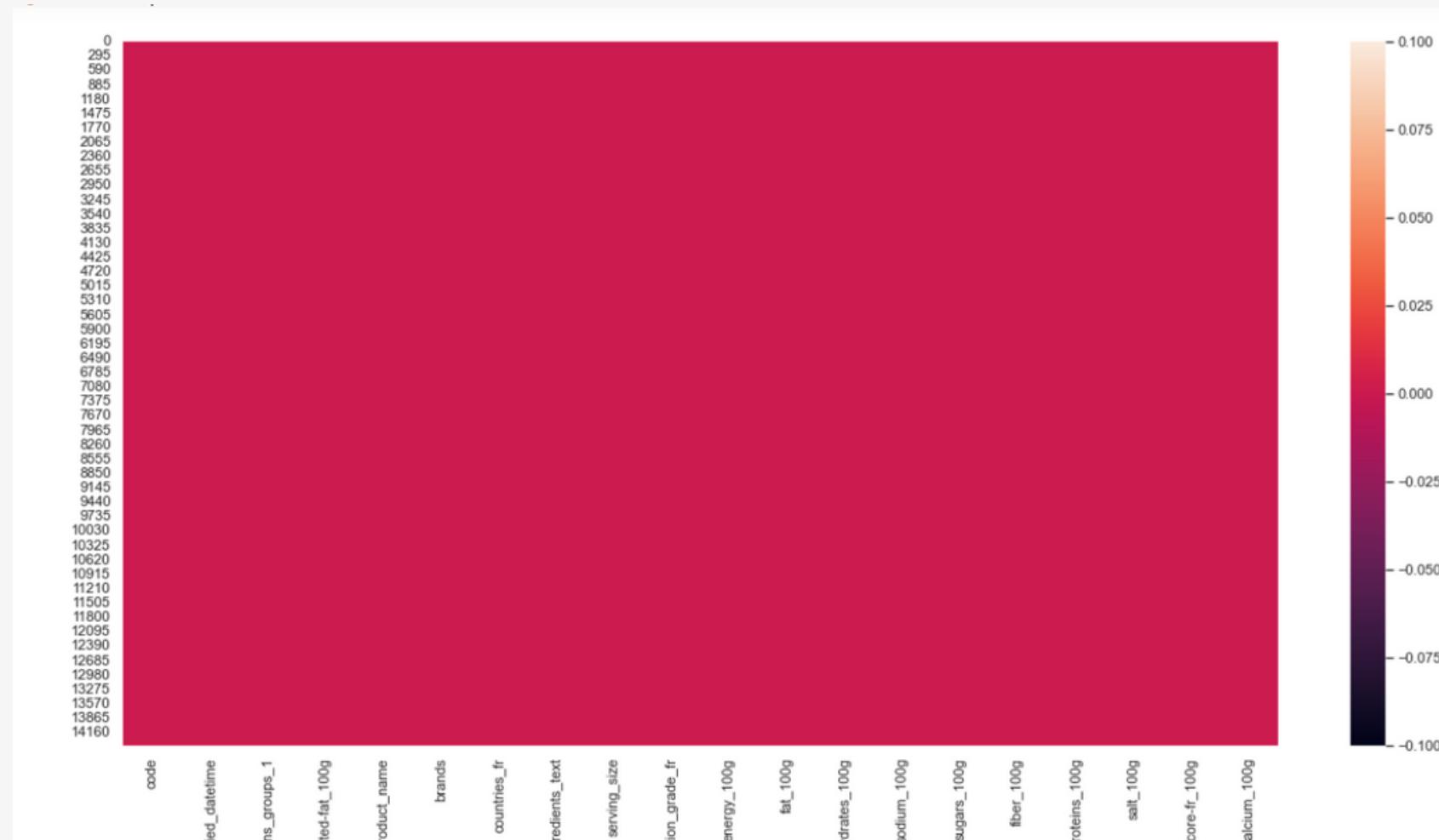
3.4. Suppression des Nan variables catégorielles

```
: data_sans_outlier.dropna(subset= Nutritive_nan_sup, inplace=True)
```



3.5.Imputation par la méthode KNN

```
: imputer = KNNImputer(n_neighbors=3)
imputed = imputer.fit_transform(data_sans_outlier.filter(regex='calcium_100g'))
df_imputed = pd.DataFrame(imputed, columns = data_sans_outlier.filter(regex='calcium_100g').columns)
```



on remarque au final qu'on a bien remplis notre base de données et on a plus des valeurs manquantes

4. ANALYSE EXPLORATOIRE DES VARIABLES IMPORTANTES

1

Analyse univariée

2

Analyse bivariée

3

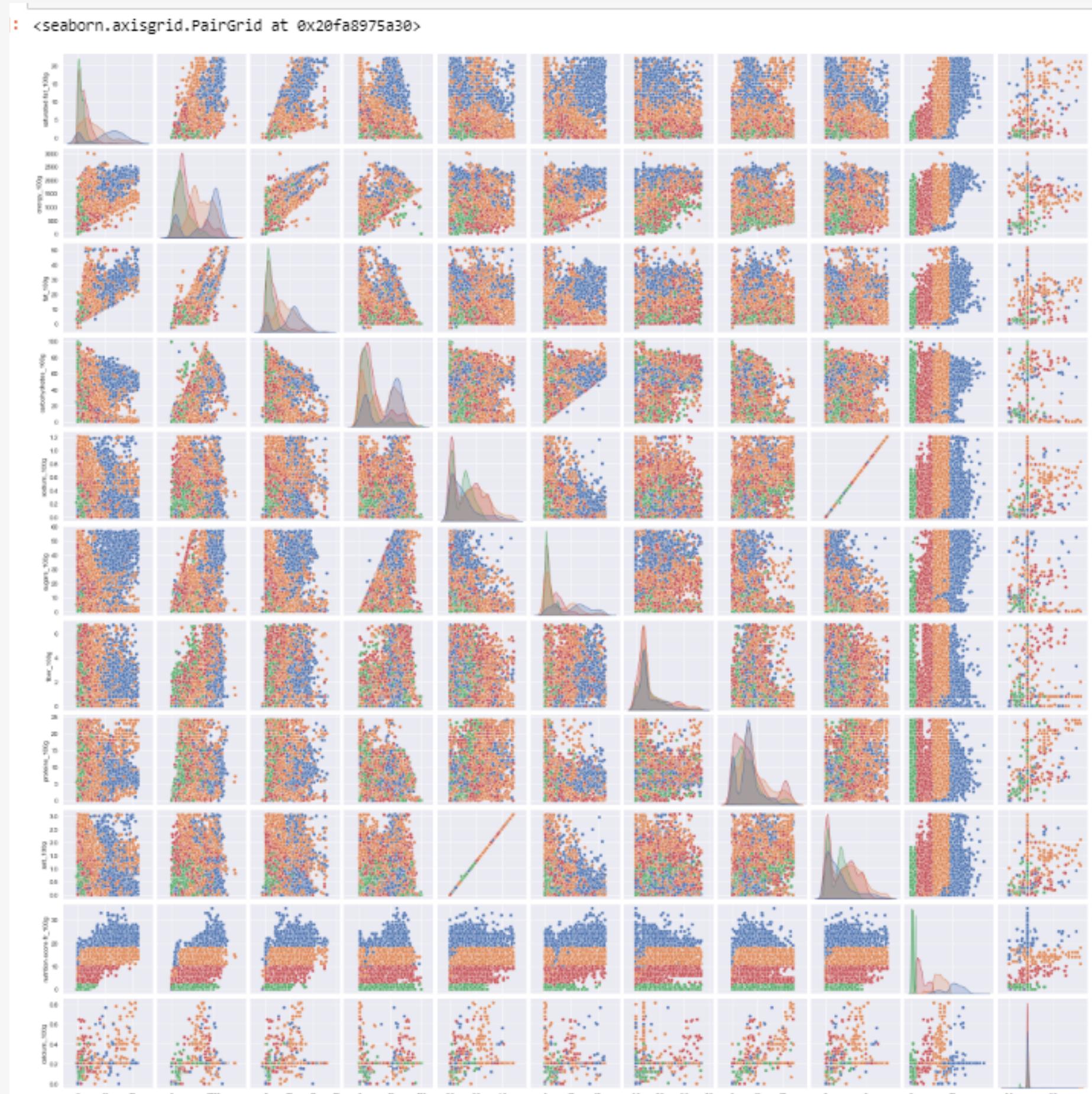
Test de normalité avec shapiro

4

Analyse explicative Kruskal Wallis

5

Analyse multivariable ACP



Ce graphe montre la répartition du nutri_grade par rapport a toute les variables

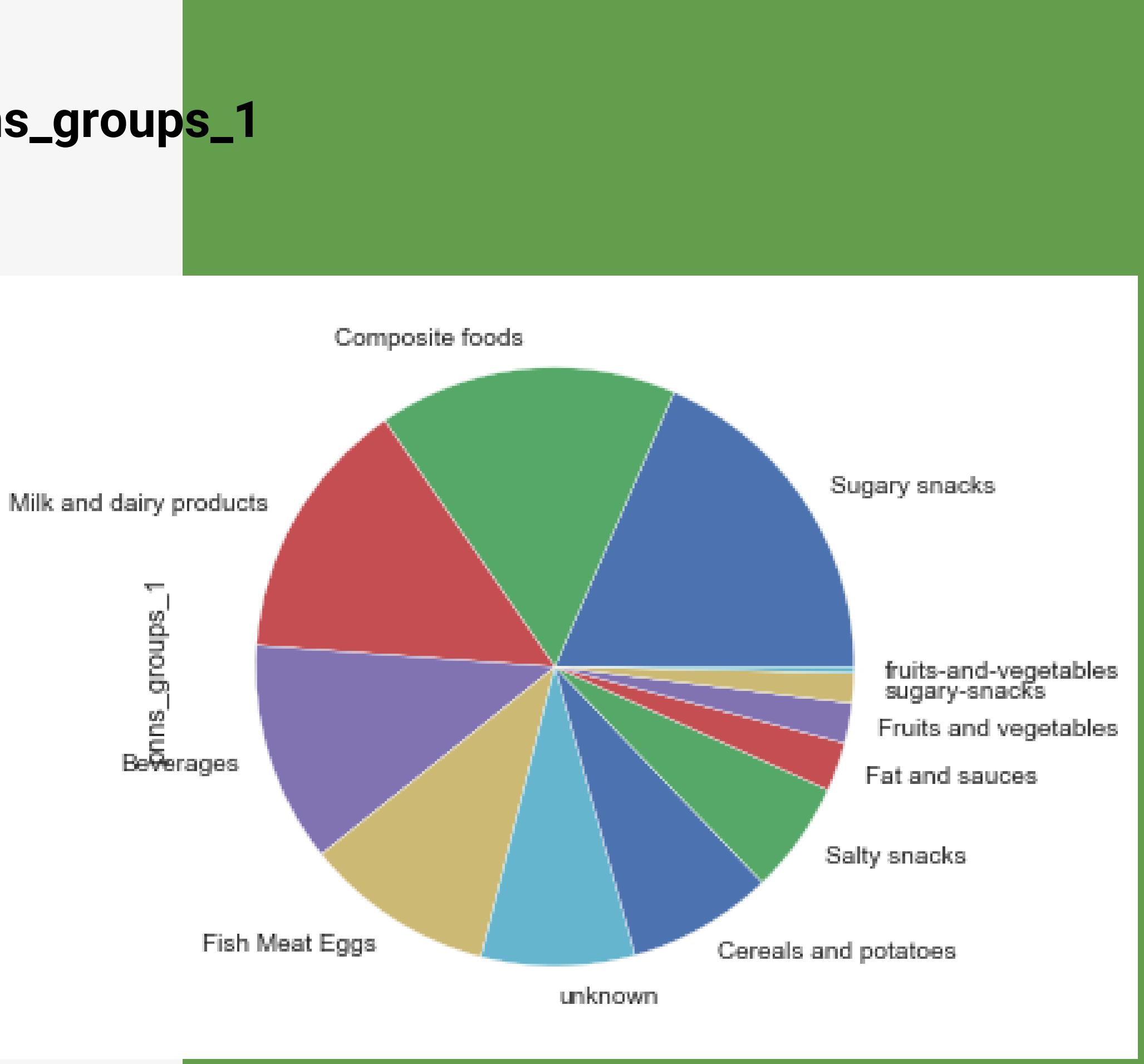
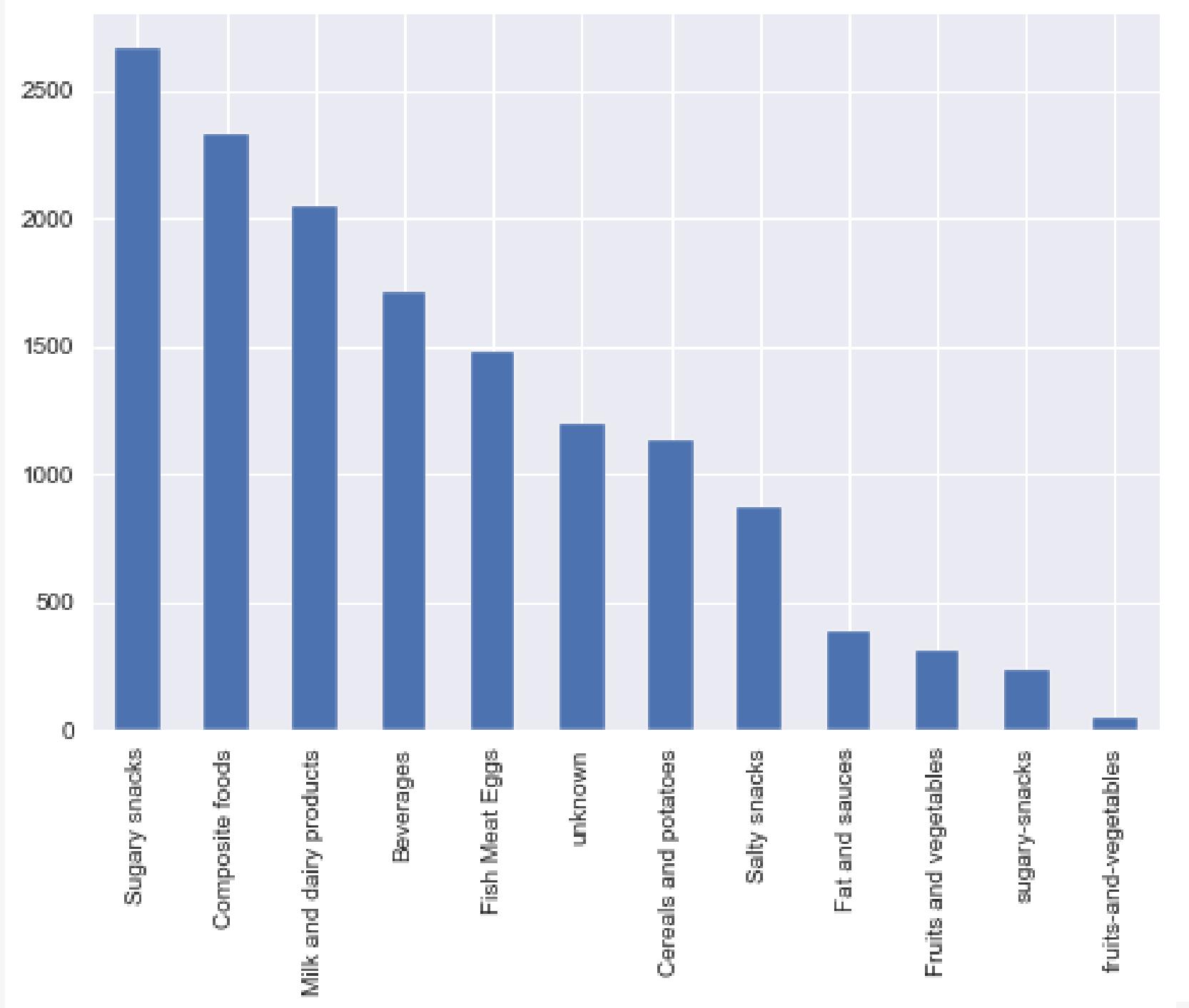
3.1.Analyse univariée

3.1.1.variable catégorielle

nutrition_grade_fr

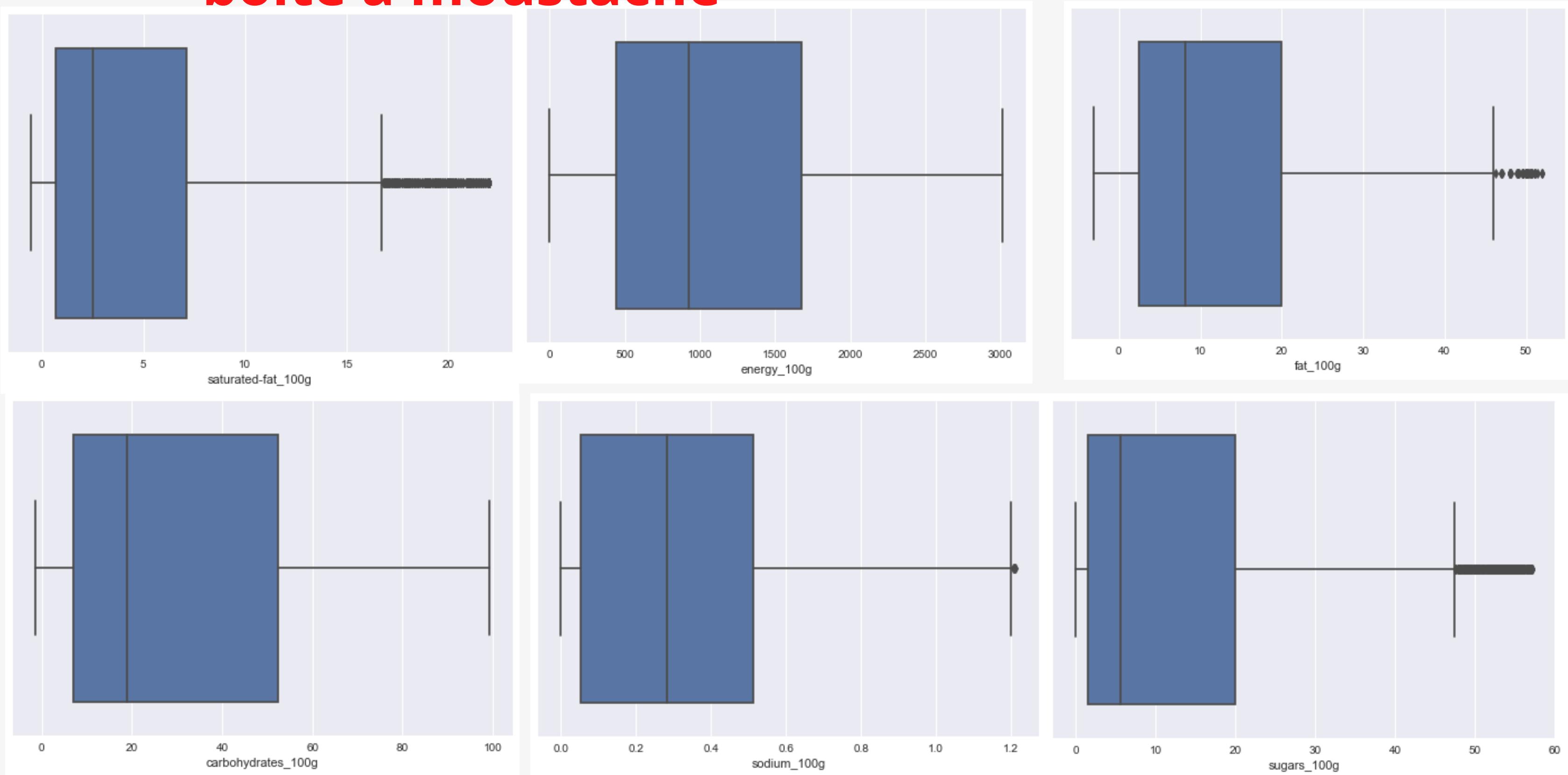


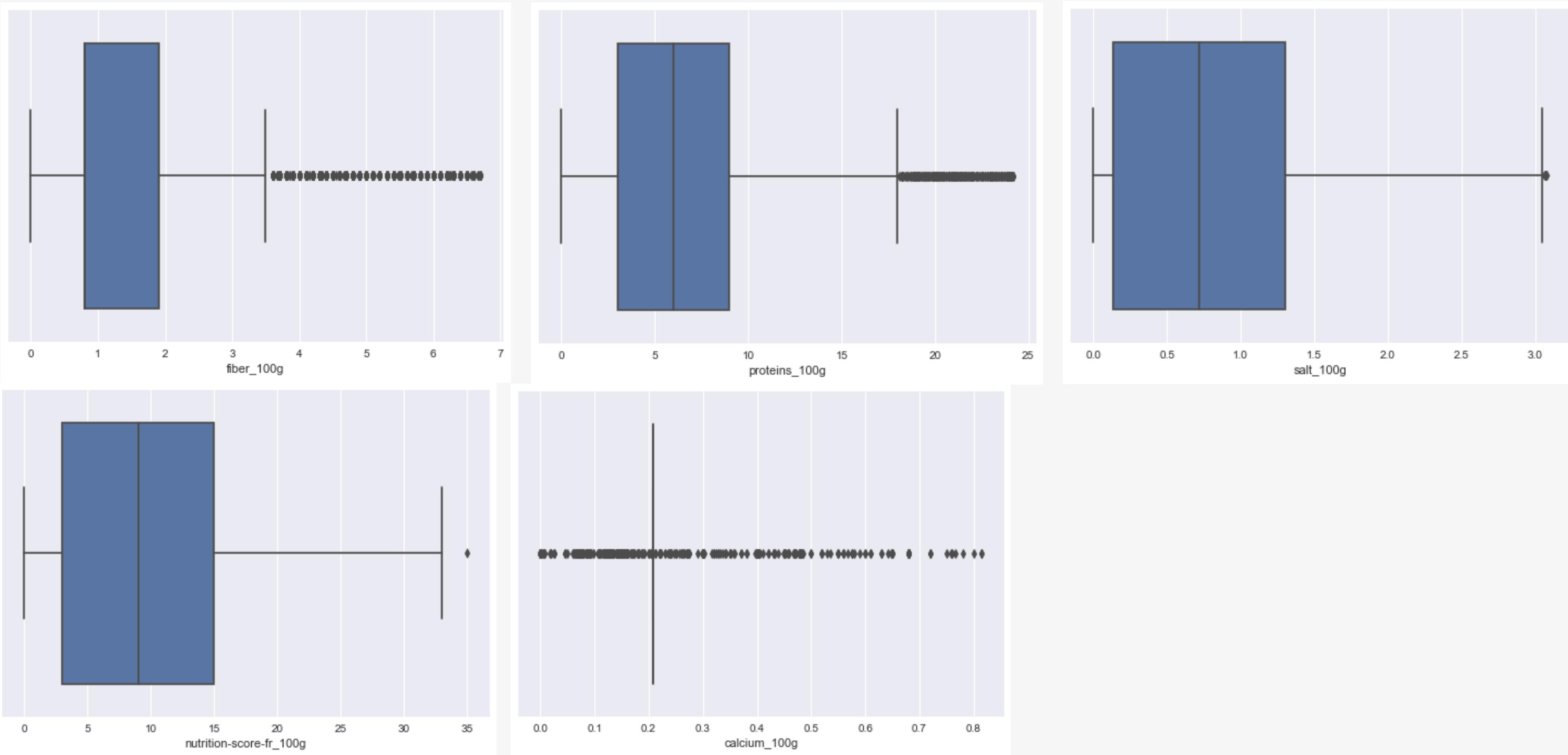
pnns_groups_1



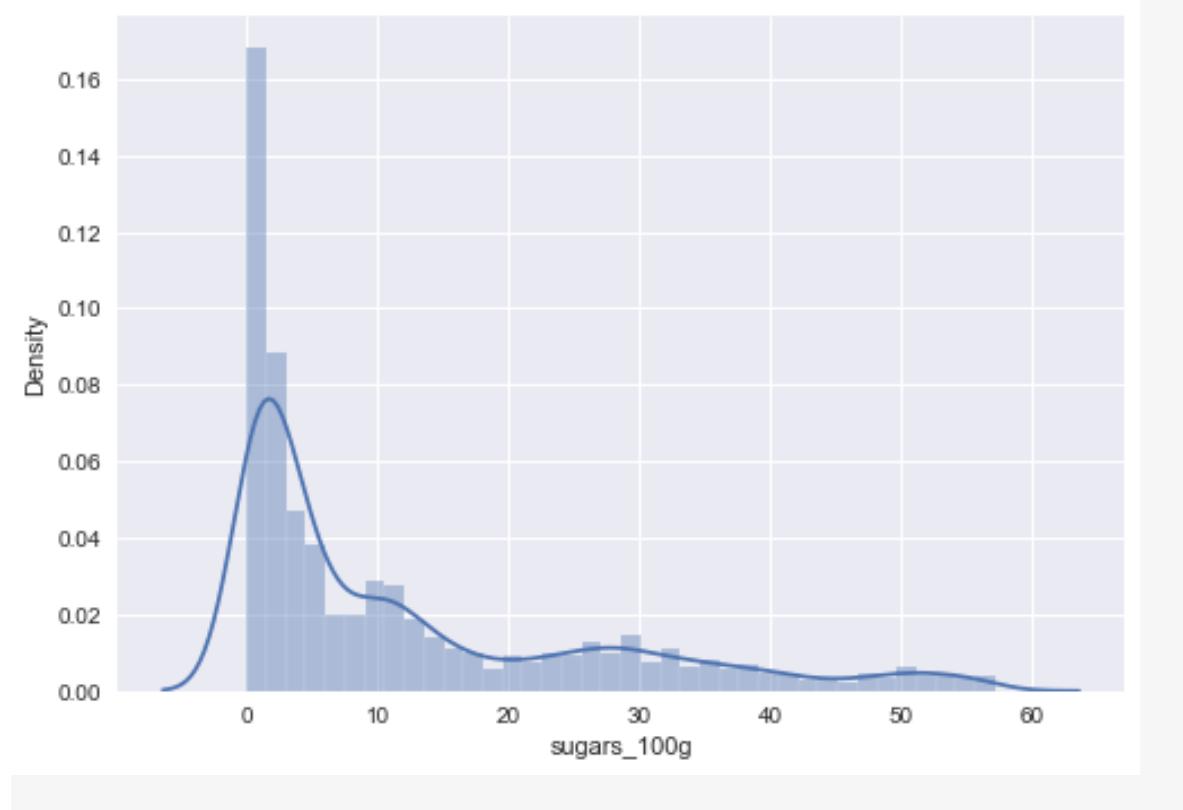
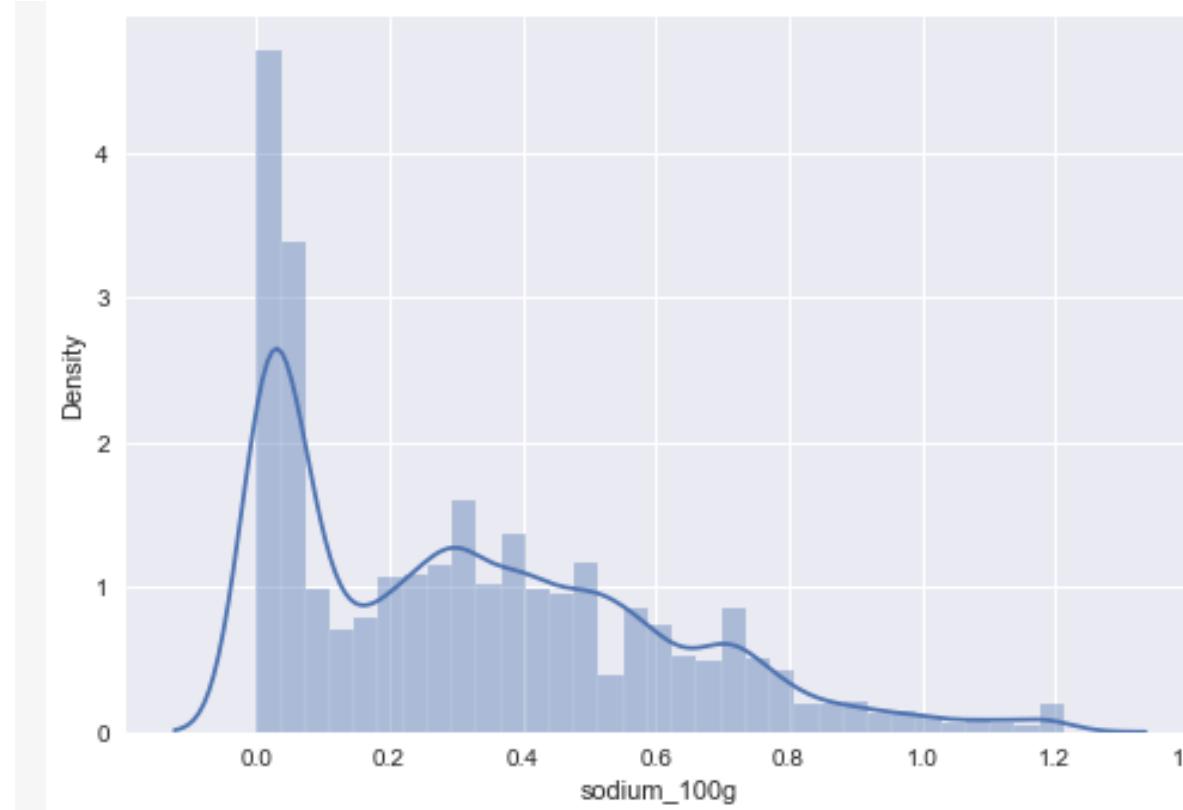
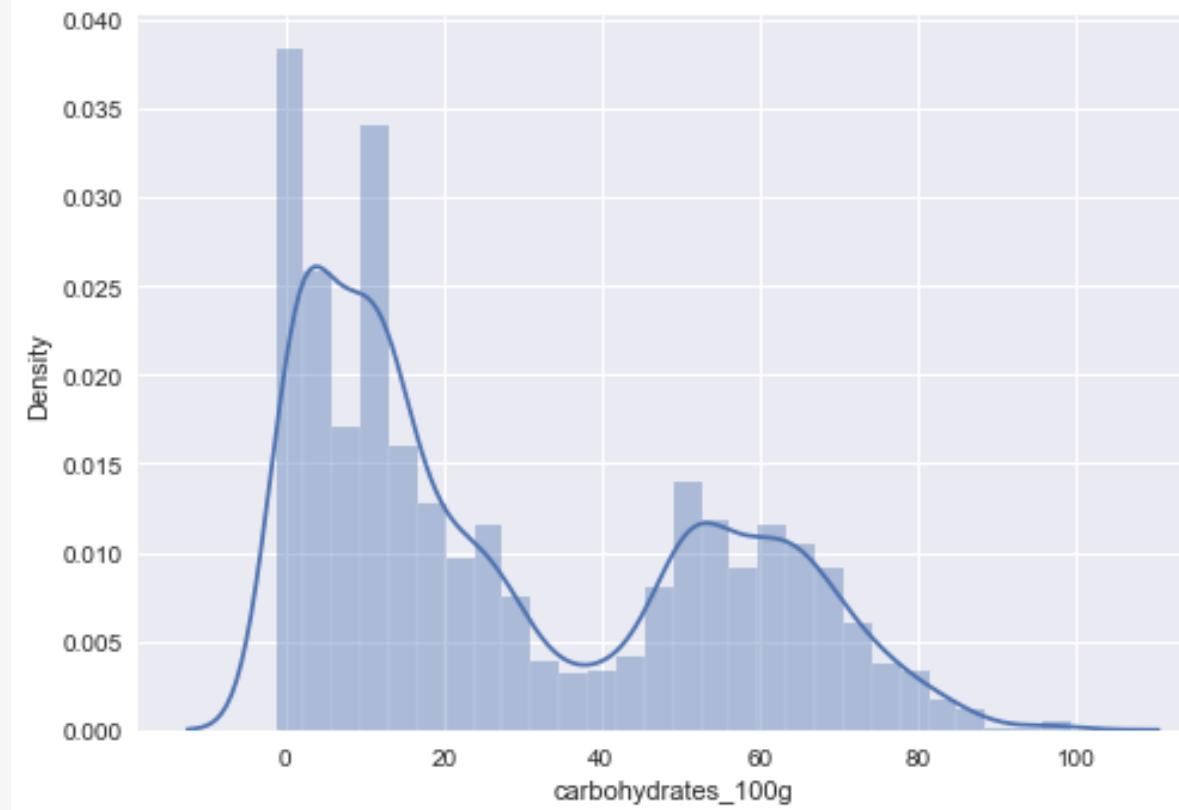
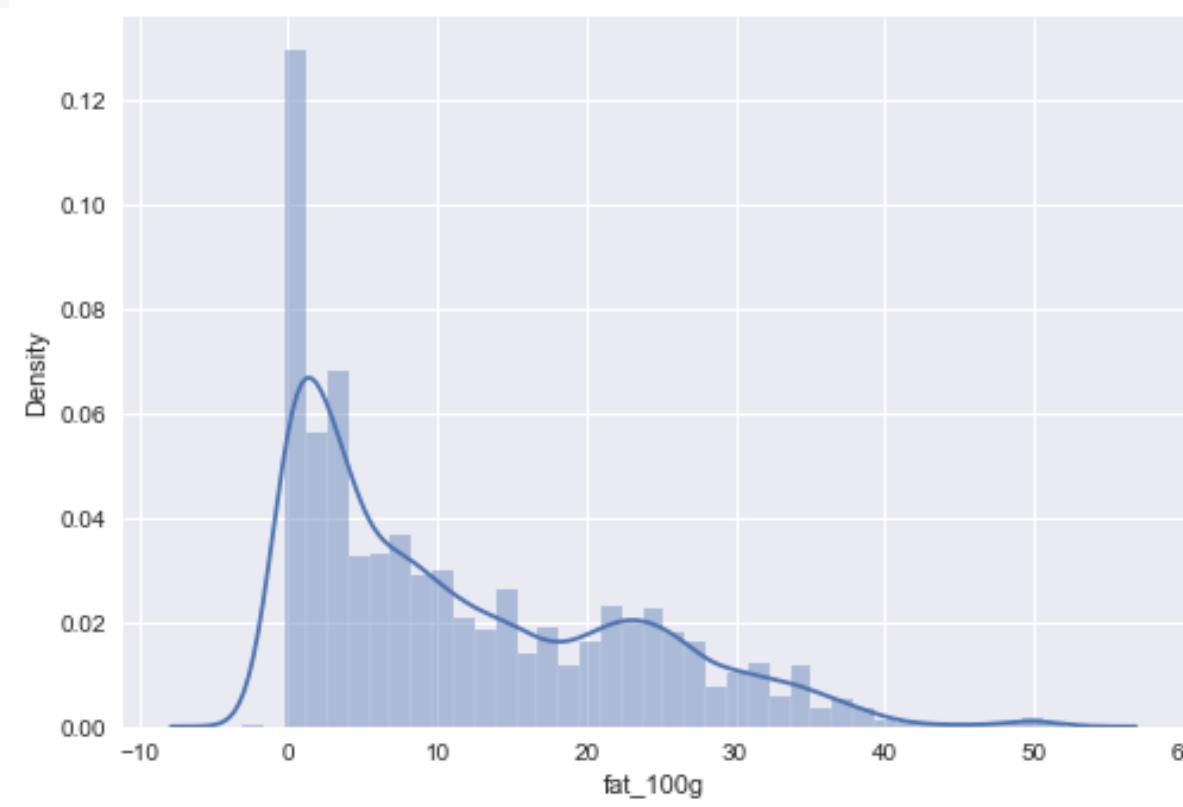
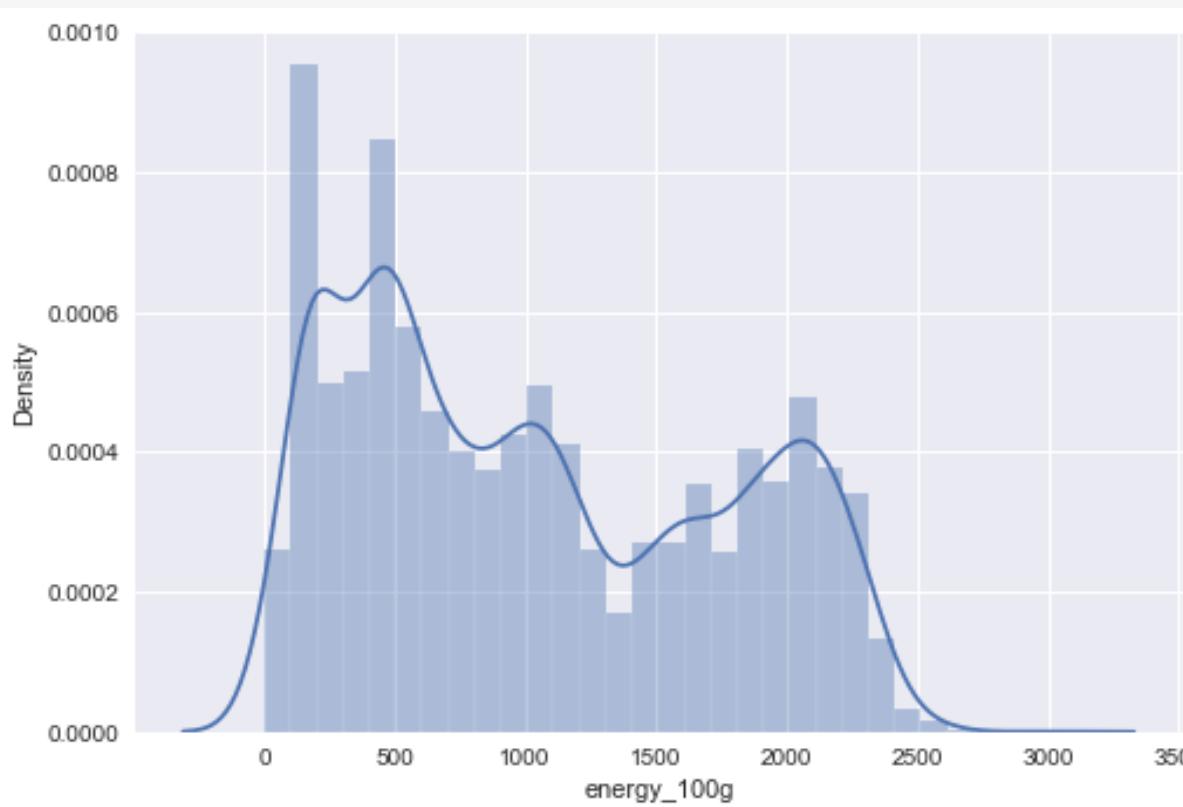
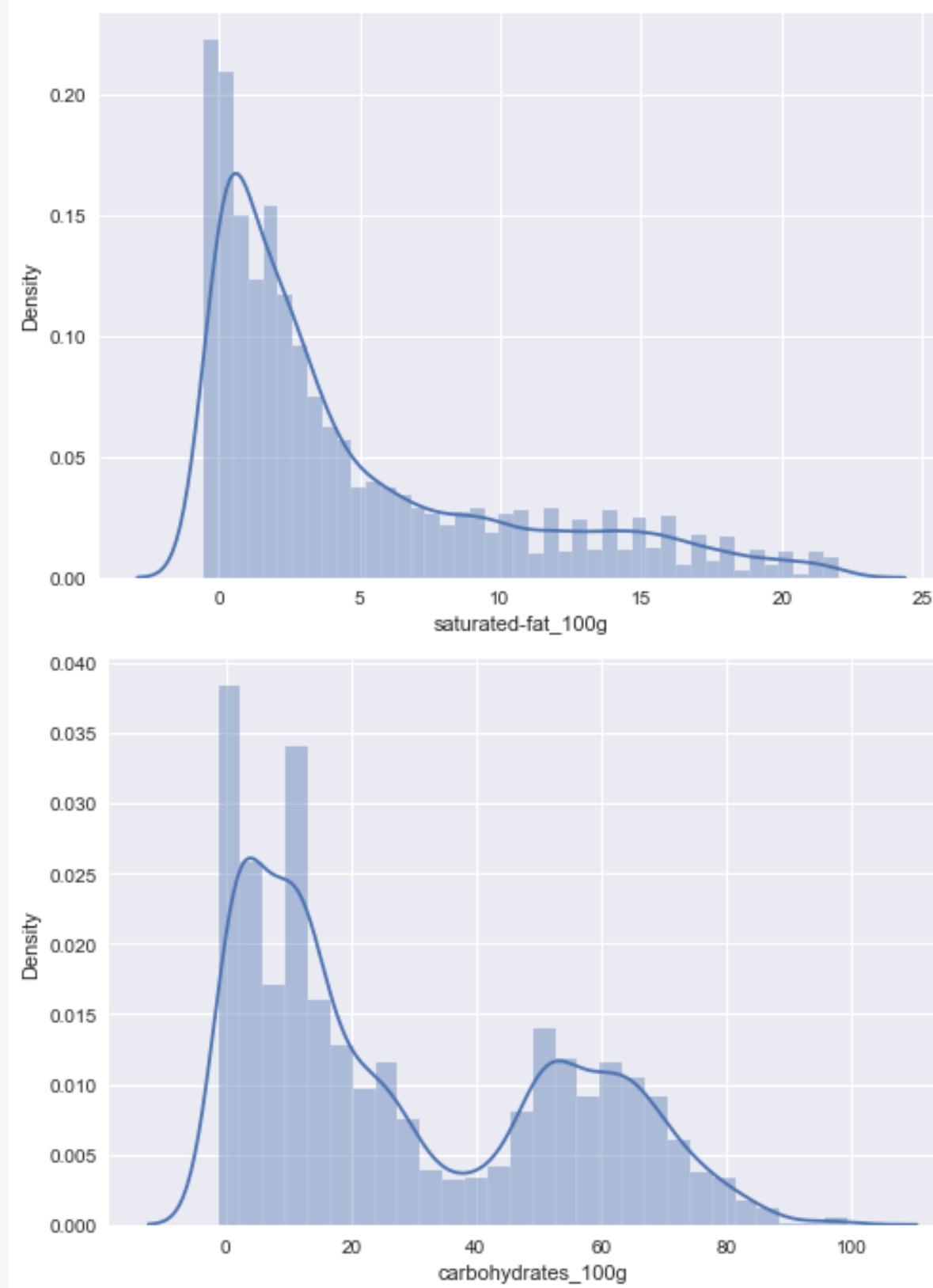
Repartition des catégories

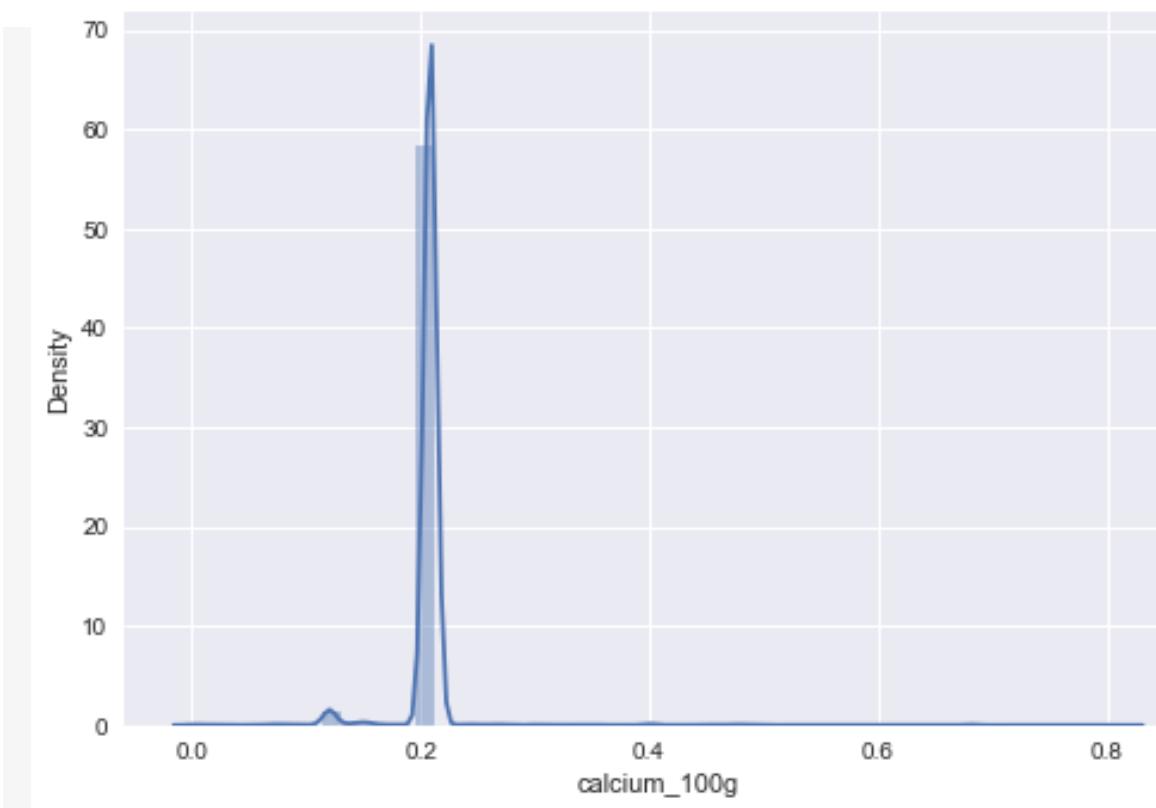
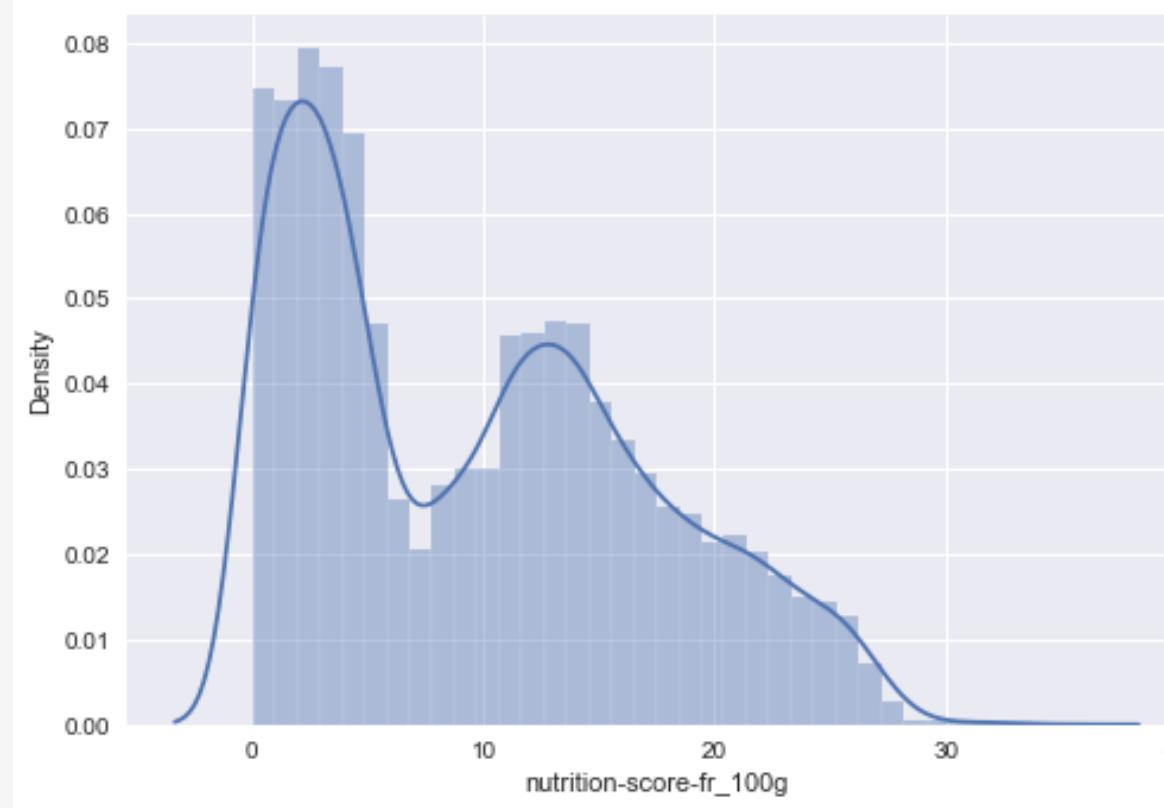
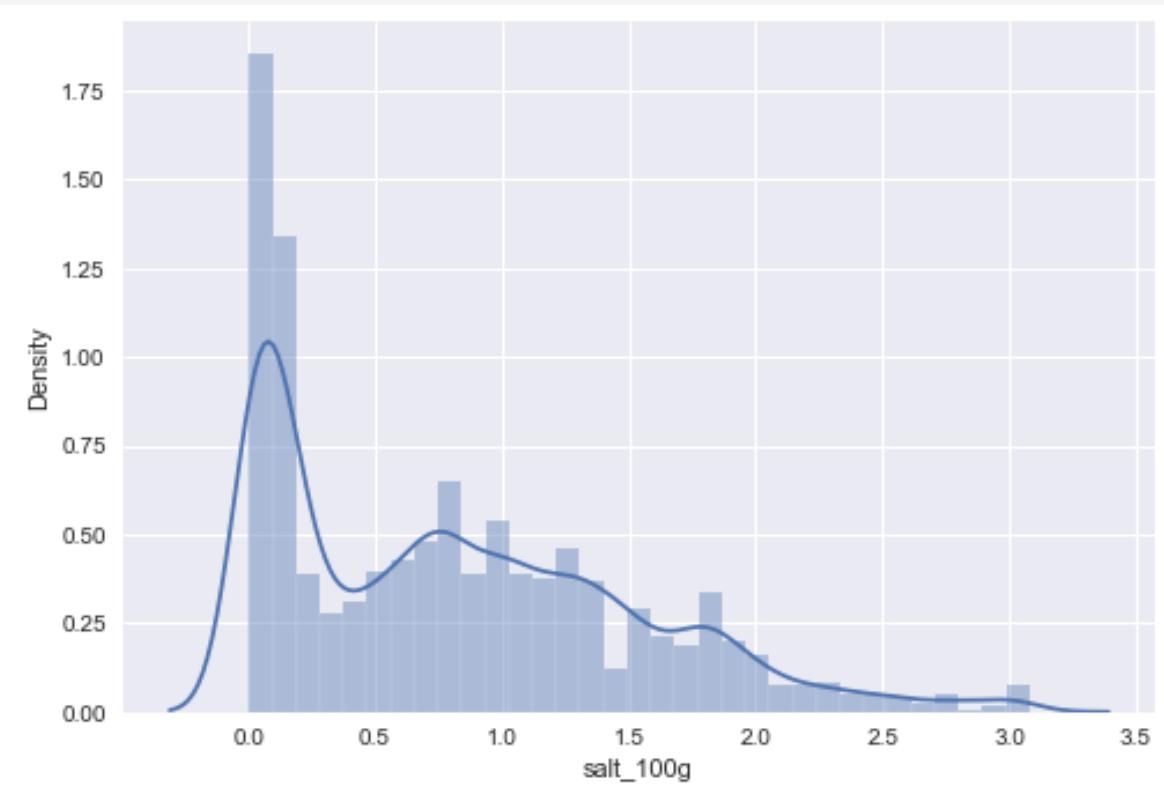
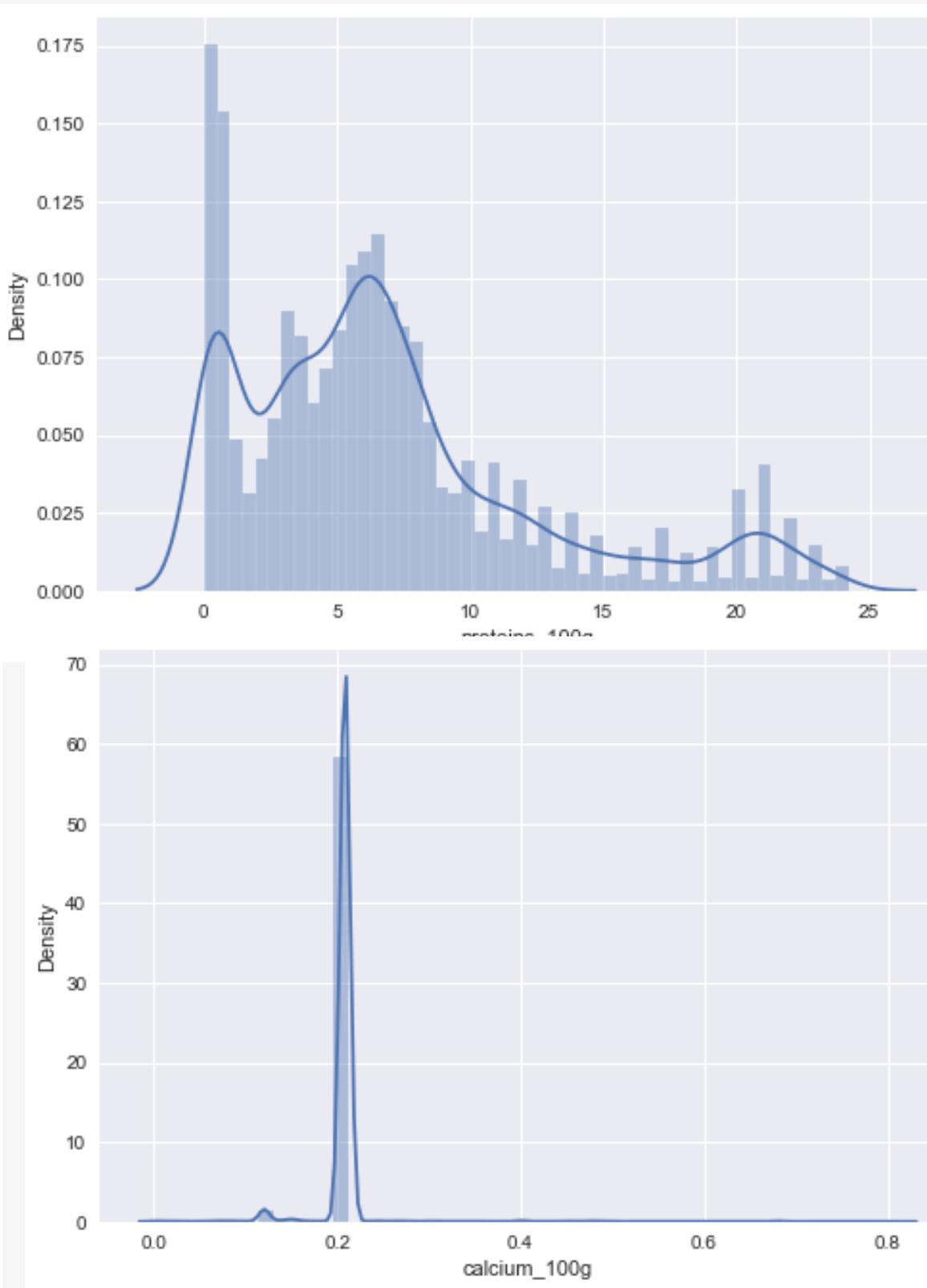
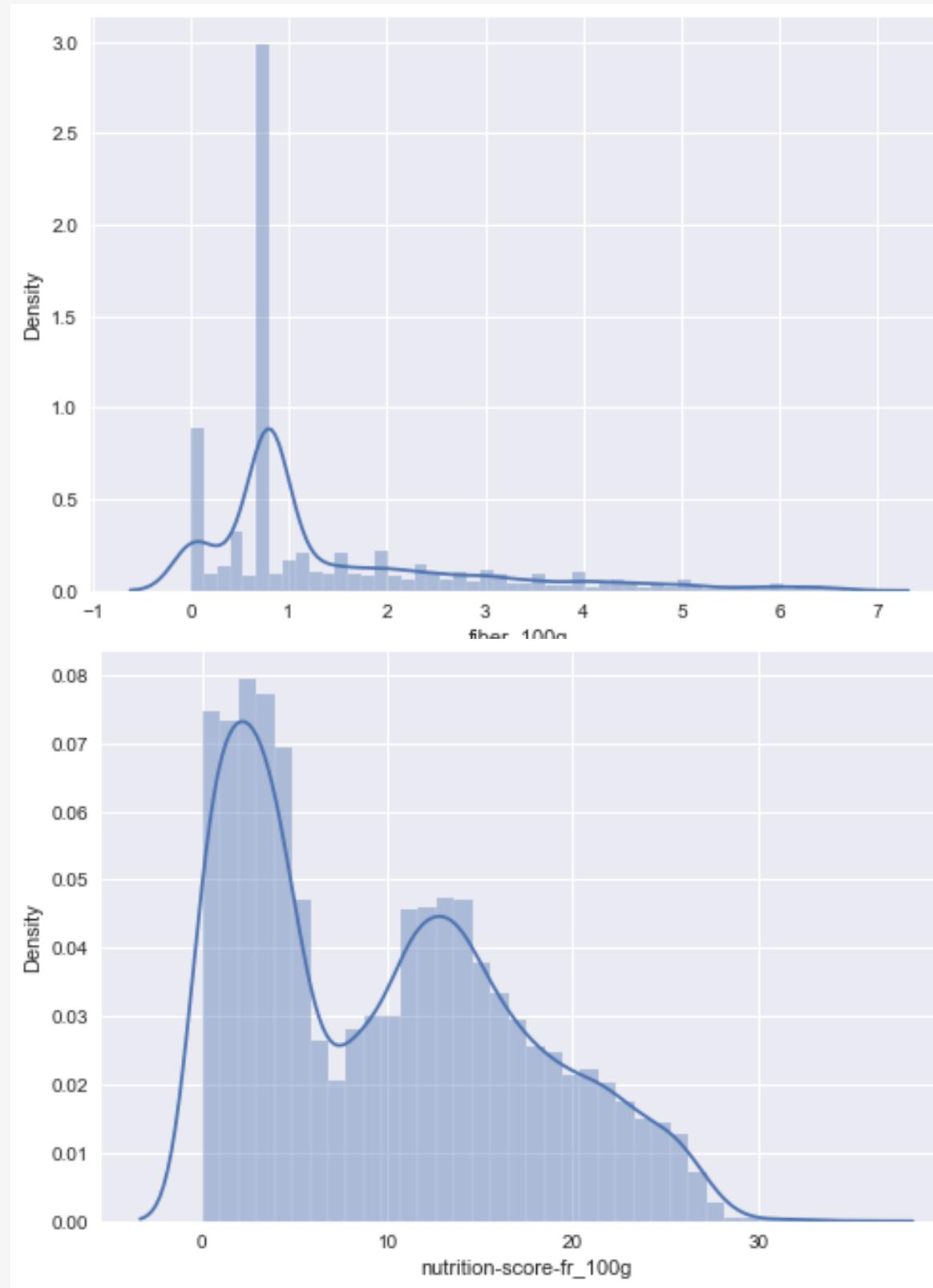
3.1.2. Variables Numériques boite à moustache

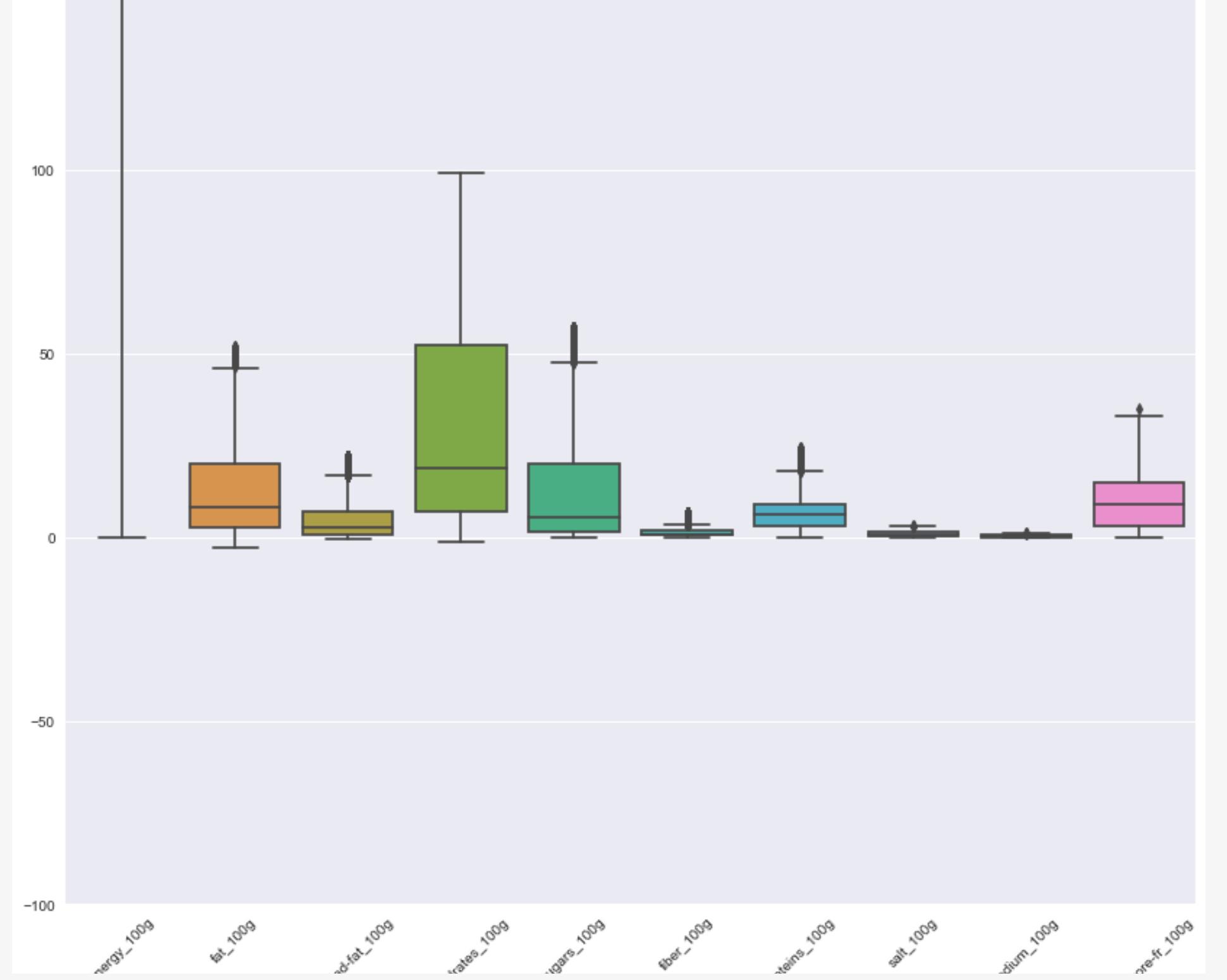




distribution de la densité de chaque variable numérique



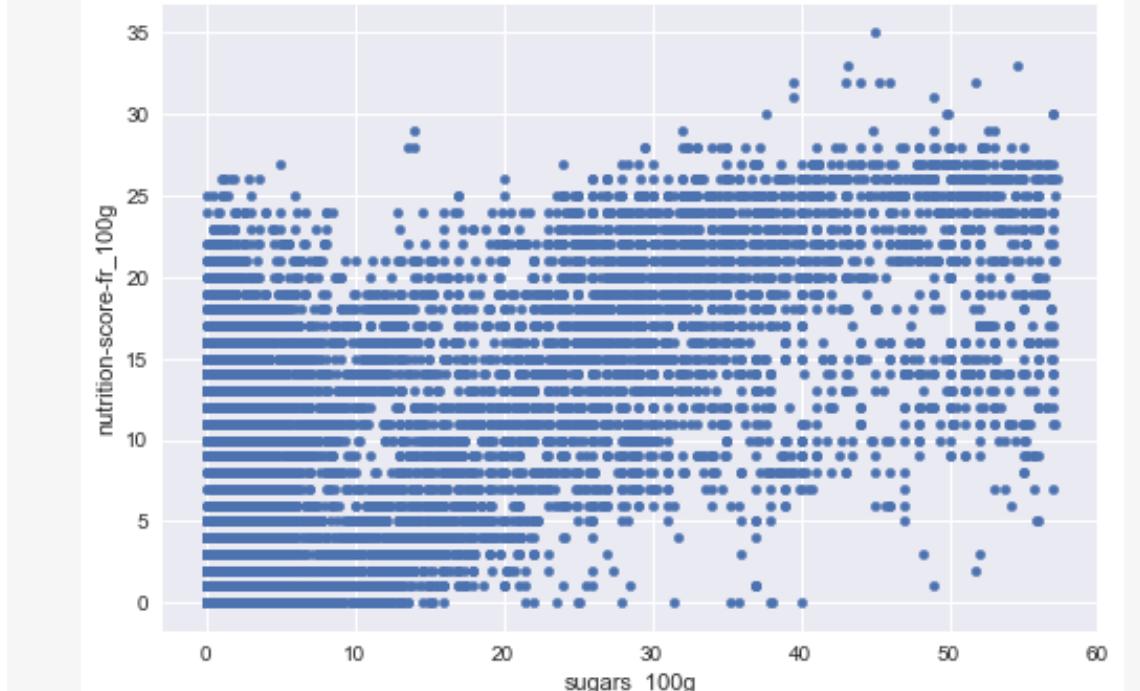
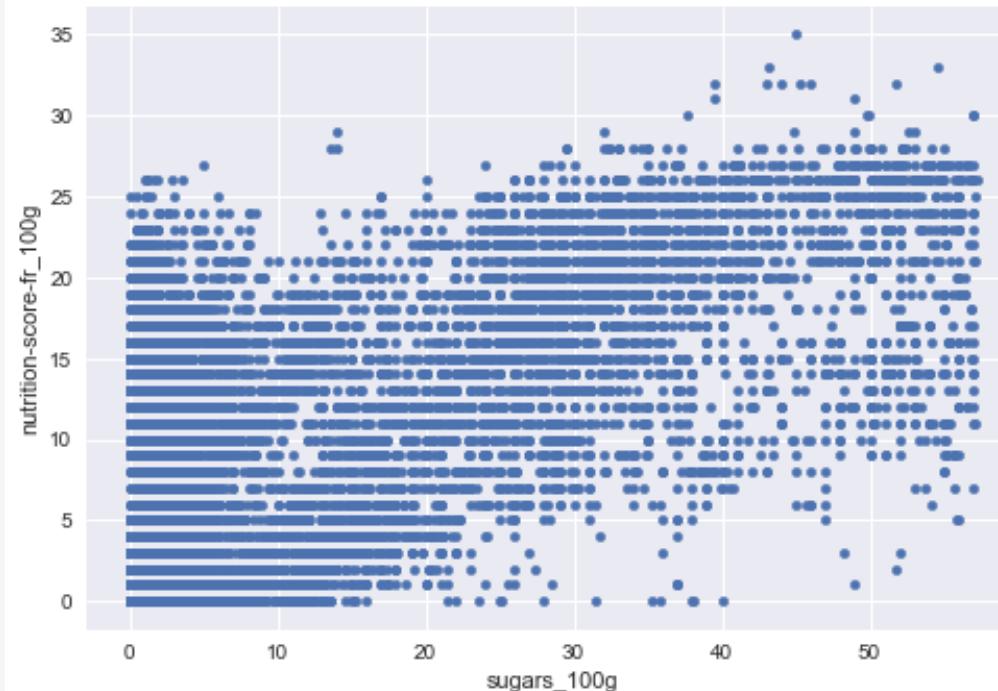
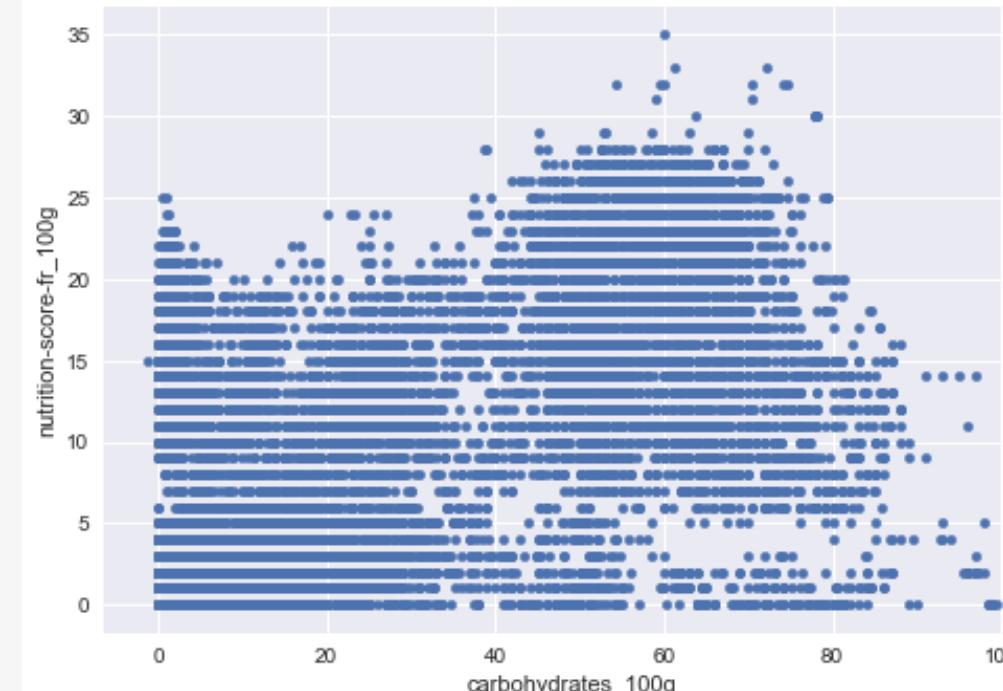
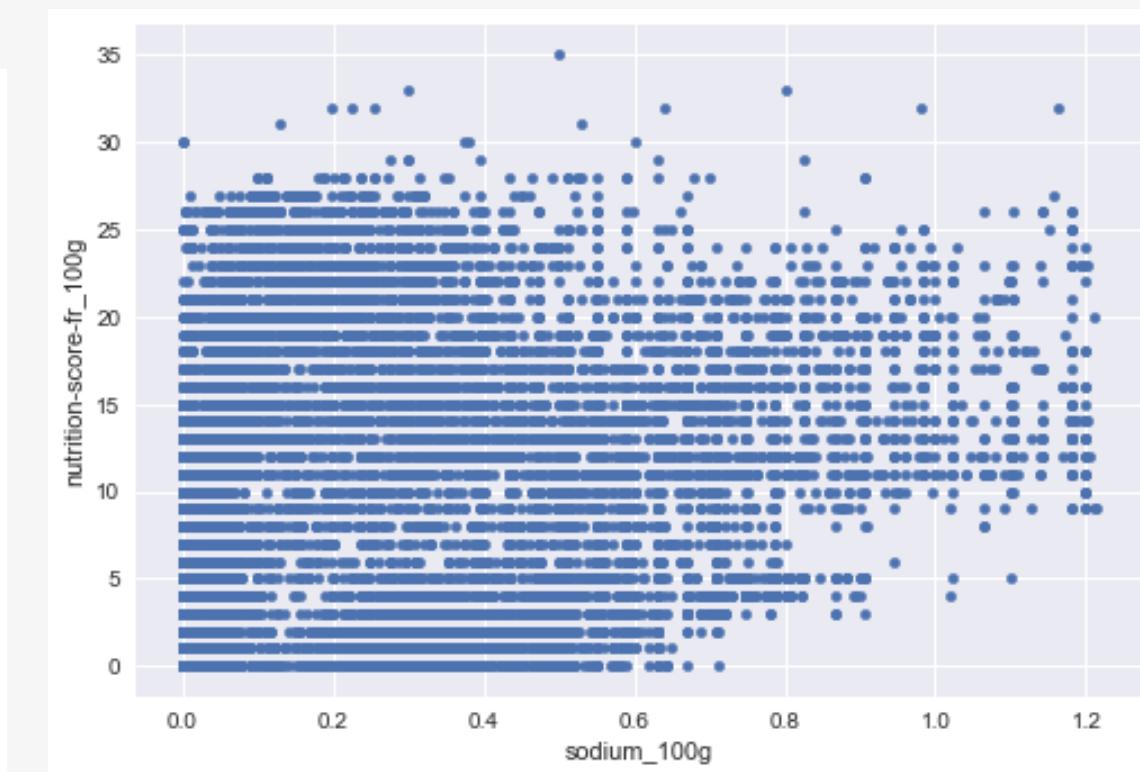
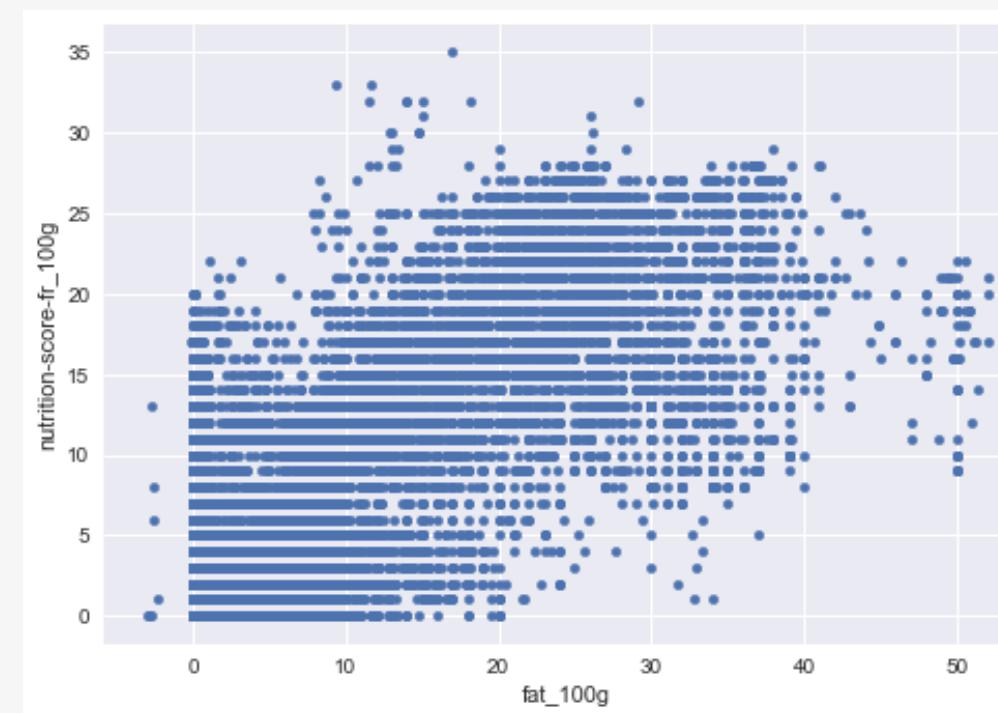
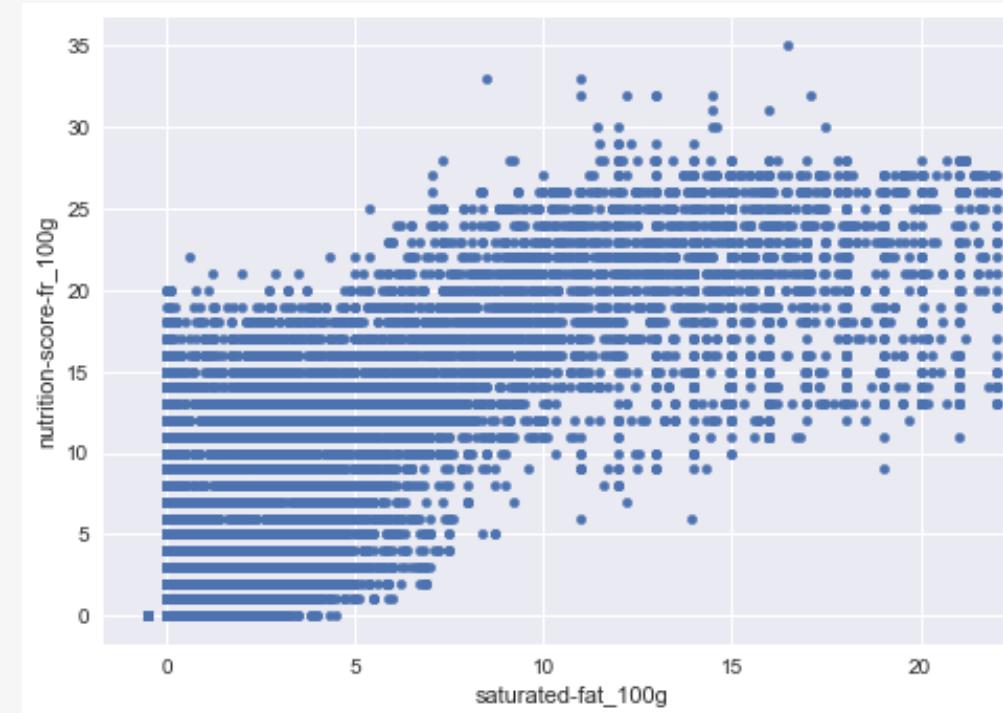


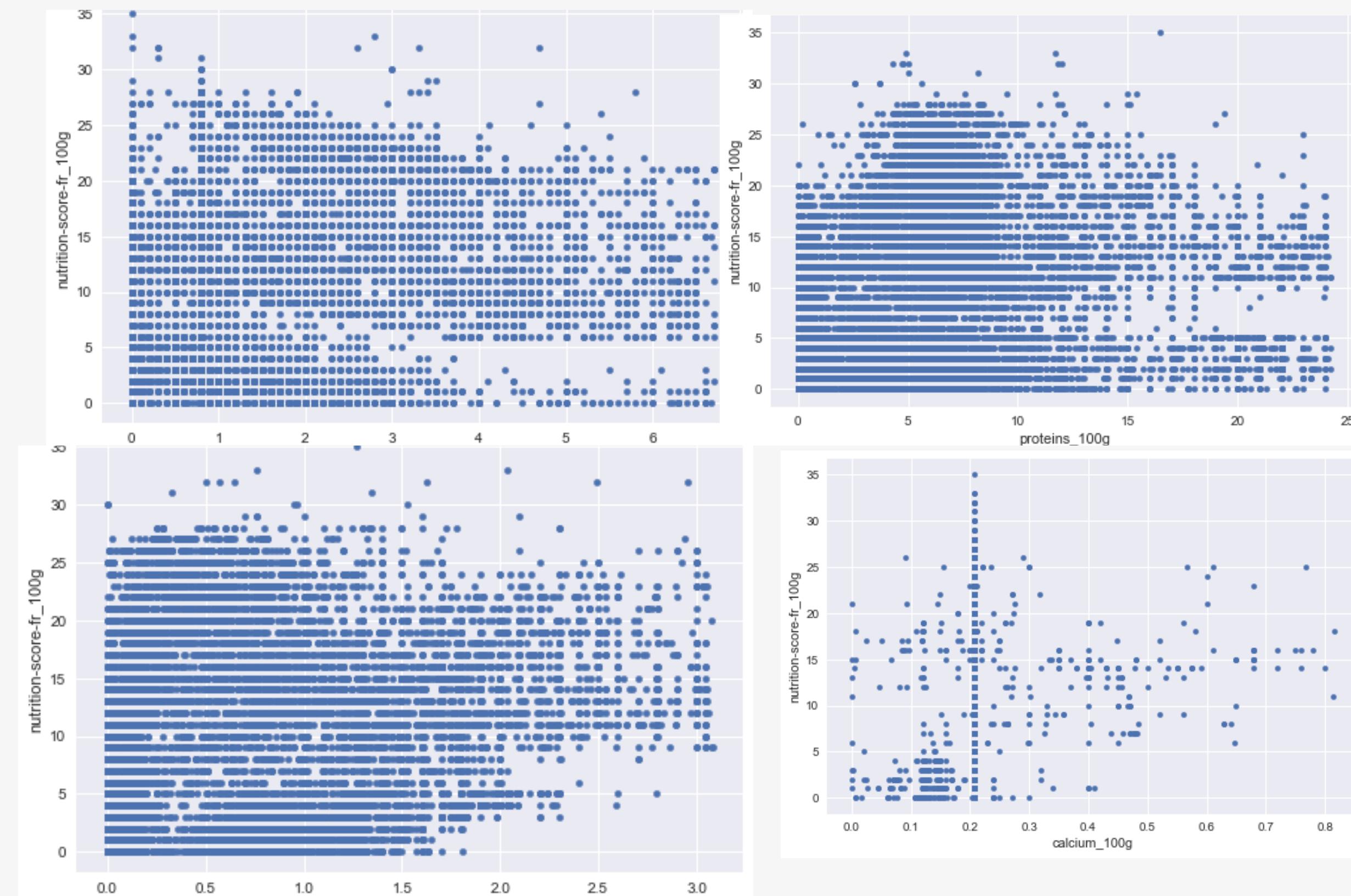


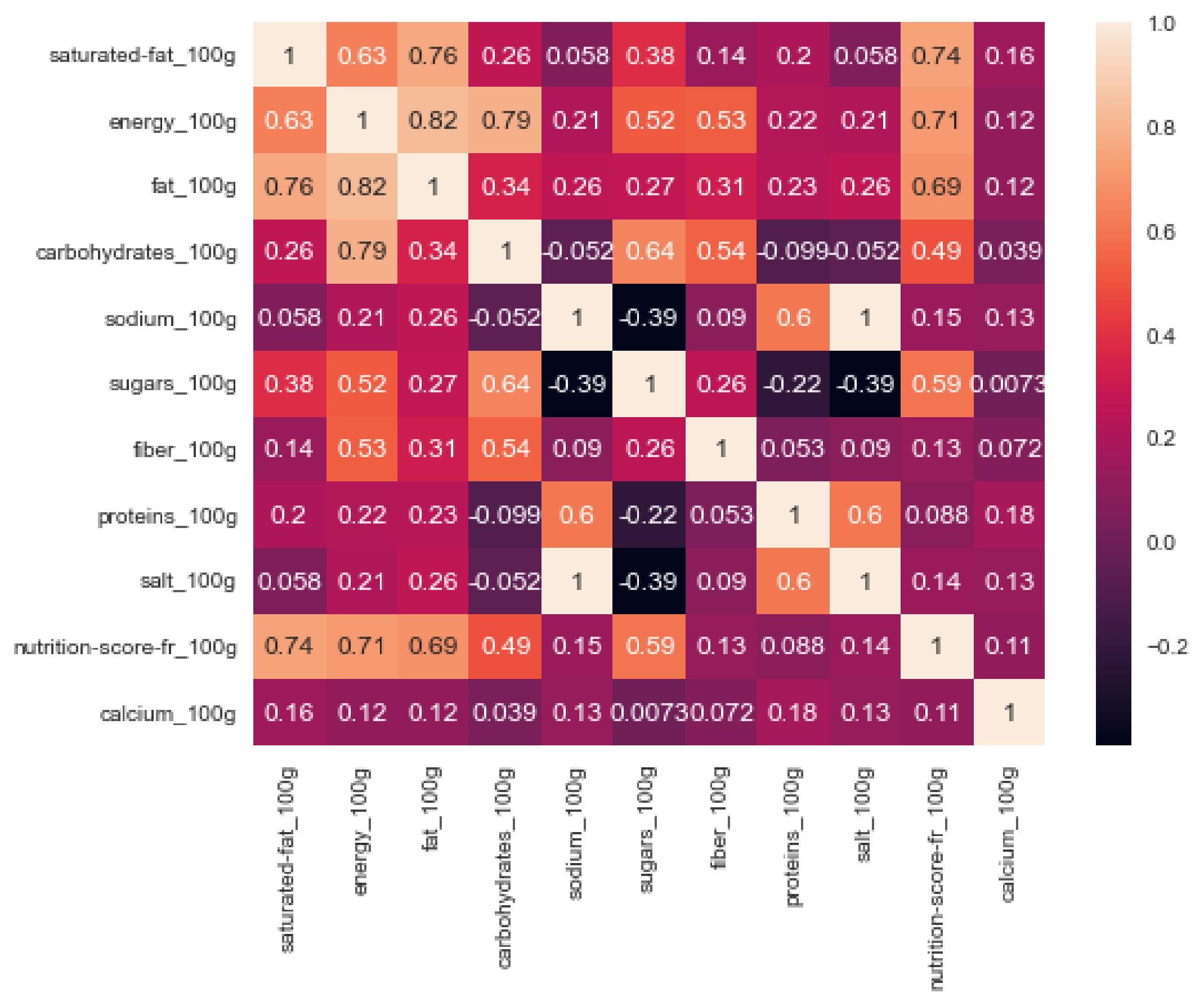
3.2.Analyse Bivariée

3.2.1.Analyse bivariée de deux variables numériques (Numérique-Numérique)

Nuages des points







energy_100g a une forte corrélation avec :

- carbohydrates_100g
- nutrition-score-fr_100g

fat_100g a une forte corrélation avec

- saturated-fat_100g
- energy_100g

nutrition-score-fr_100g a une forte corrélation avec

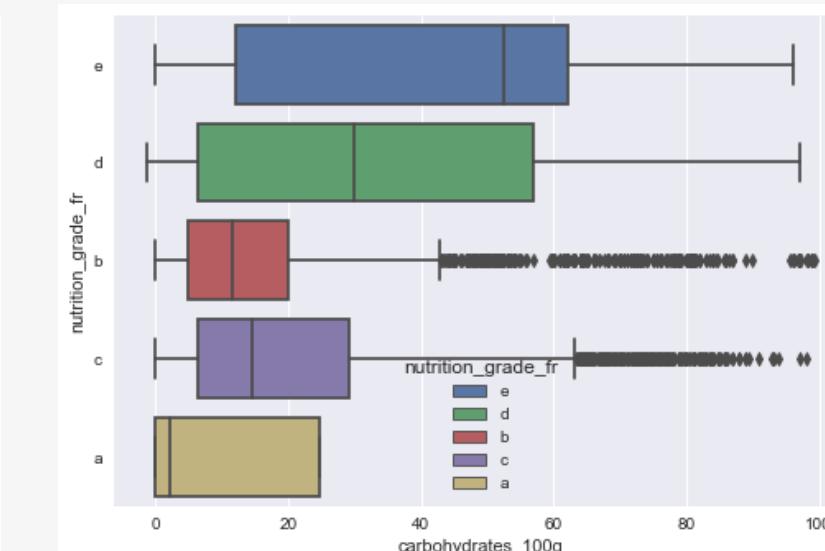
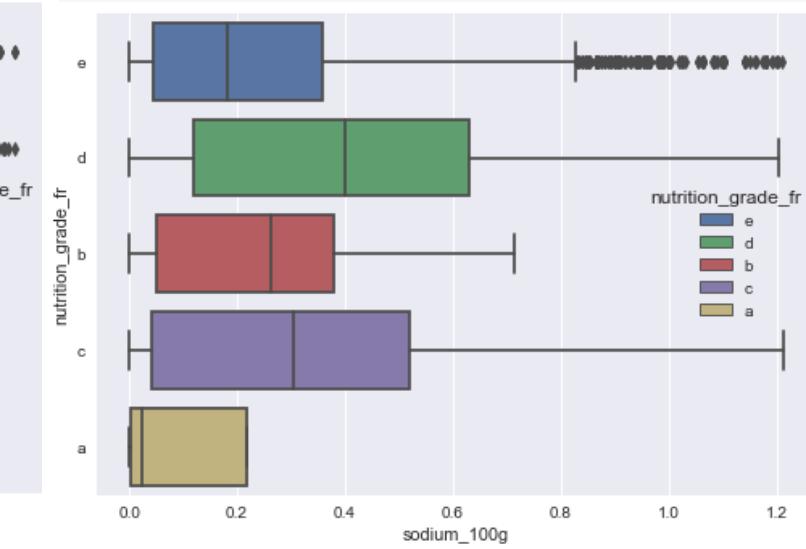
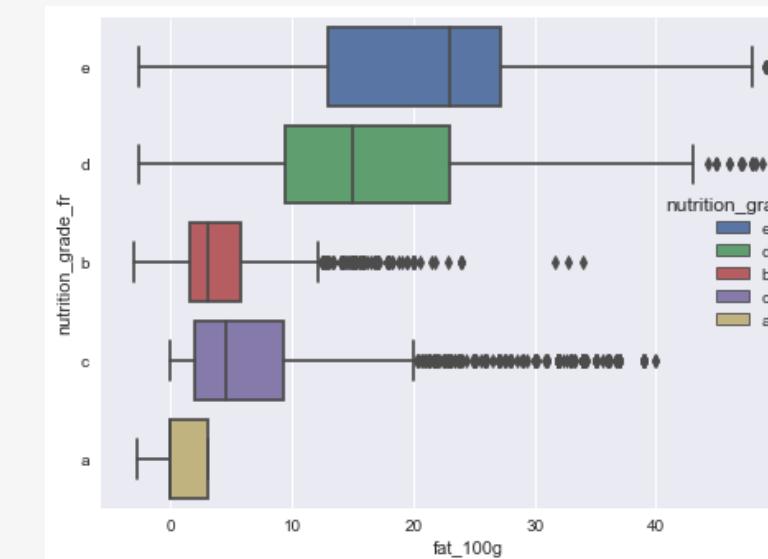
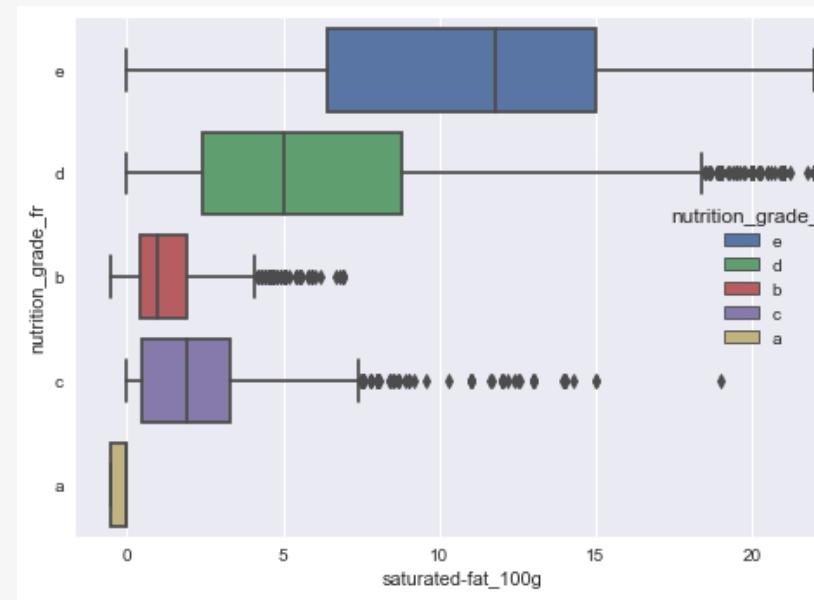
- energy_100g
- saturated_fat_100g

carbohydrates_100g est fortement corrélées avec energy_100g

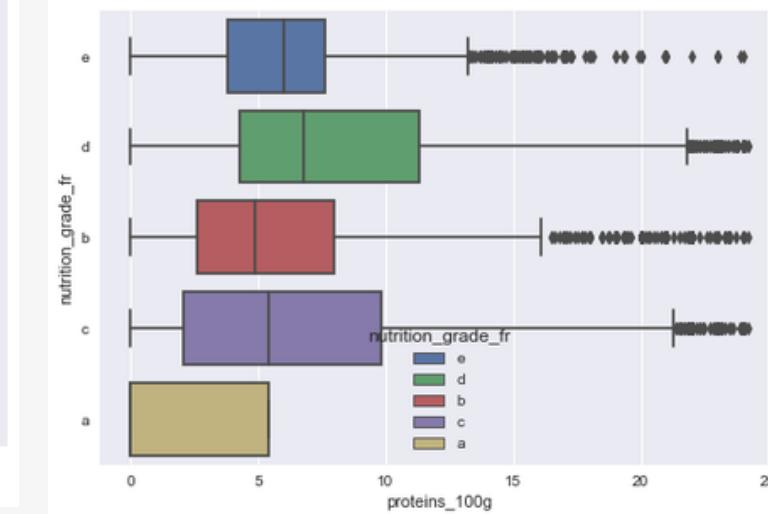
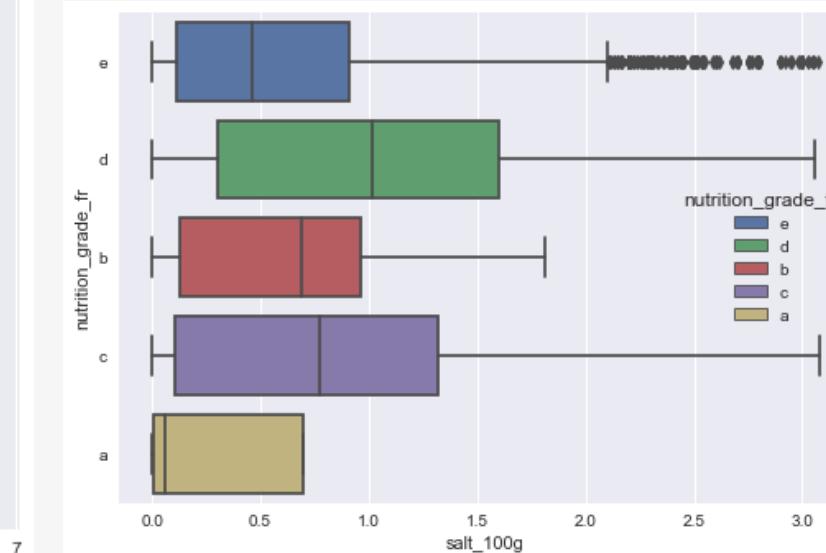
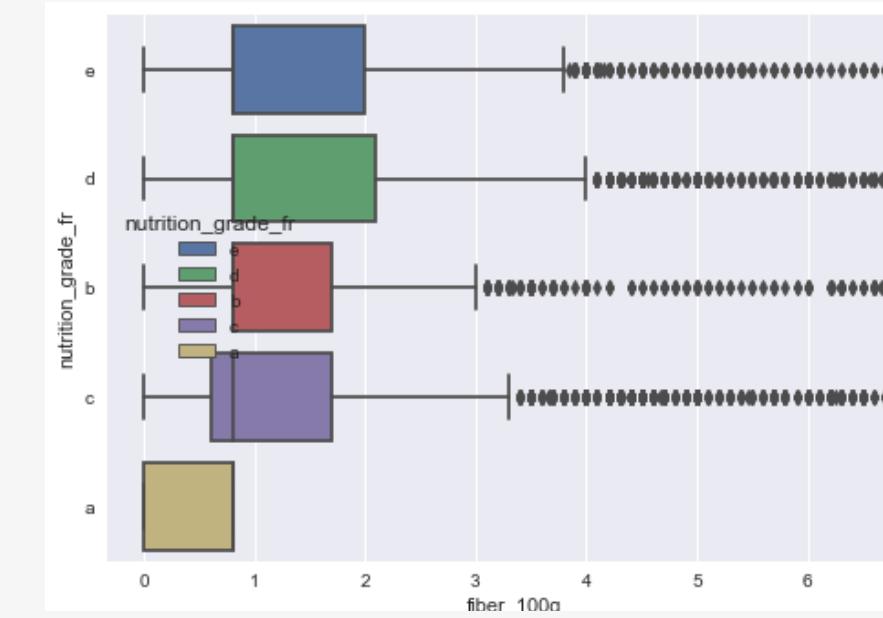
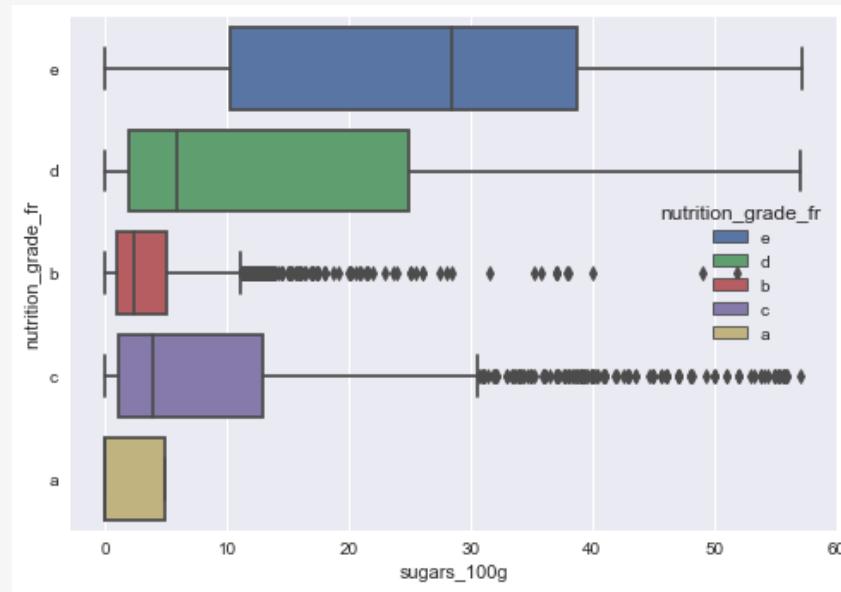
saturated_fat_100ga une forte corrélation avec

- fat_100g
- nutrition-score-fr_100

3.2.2. Analyse Bivariée d'une variable Numérique et d'une variable catégorielle (Numérique-Catégorique)



ici on remarque que le nutrigrade d et e est élevé en carbohydrate



On constate qu'il y avait une différence entre le type de nutrigrade (A,B,C,D, E) et les différentes variables (voir le boxplot)

Pour investiguer si différence entre les groupes sont statistiquement significatif => (shapiro: tester la normalité , et ANOVA / Kruskal wallis)

3.3.Test de normalité avec shapiro

```
# normality test

for column in numeric_columns:
    print('_____ \n{}'.format(column))
    stat, p = shapiro(data_clean[column])
    print('Statistics=% .3f, p=% .3f' % (stat, p))

# interpret
alpha = 0.05    if p > alpha:
    print("H0 ne peut être rejetée :{}, on considère l'hypothèse de normalité, l'échantillon semble normal".format(column))
else:
    print("H0 est rejetée : {} n'est pas une distribution normale, l'échantillon n'a pas l'air gaussien".format(column))
```

```
saturated-fat_100g
Statistics=0.816, p=0.000
H0 est rejetée : saturated-fat_100g n'est pas une distribution normale, l'échantillon n'a pas l'air gaussien

energy_100g
Statistics=0.928, p=0.000
H0 est rejetée : energy_100g n'est pas une distribution normale, l'échantillon n'a pas l'air gaussien

fat_100g
Statistics=0.892, p=0.000
H0 est rejetée : fat_100g n'est pas une distribution normale, l'échantillon n'a pas l'air gaussien

carbohydrates_100g
Statistics=0.885, p=0.000
H0 est rejetée : carbohydrates_100g n'est pas une distribution normale, l'échantillon n'a pas l'air gaussien

sodium_100g
Statistics=0.917, p=0.000
H0 est rejetée : sodium_100g n'est pas une distribution normale, l'échantillon n'a pas l'air gaussien

sugars_100g
Statistics=0.801, p=0.000
H0 est rejetée : sugars_100g n'est pas une distribution normale, l'échantillon n'a pas l'air gaussien

fiber_100g
Statistics=0.784, p=0.000
H0 est rejetée : fiber_100g n'est pas une distribution normale, l'échantillon n'a pas l'air gaussien

proteins_100g
Statistics=0.894, p=0.000
H0 est rejetée : proteins_100g n'est pas une distribution normale, l'échantillon n'a pas l'air gaussien

salt_100g
Statistics=0.917, p=0.000
H0 est rejetée : salt_100g n'est pas une distribution normale, l'échantillon n'a pas l'air gaussien

nutrition-score-fr_100g
Statistics=0.931, p=0.000
H0 est rejetée : nutrition-score-fr_100g n'est pas une distribution normale, l'échantillon n'a pas l'air gaussien
```

Suite au test de normalité avec shapiro qui montre que toute les nutritions ont une distribution anormale , de ce fait je ne pourrai pas faire l'ANOVA mais j'appliquerai le test de Kruskal Wallis (qui n'est pas liée à la normalité d'un échantillon)

3.4.Analyse explicative Kruskal Wallis

```
def kruskal_most_col(data):
    X = "nutrition_grade_fr" # qualitative
    Y = data.filter(regex='_100g').columns# quantitative
    p = []
    feature = []
    for col in Y:
        p.append(stats.kruskal(data_a[col], data_b[col], data_c[col], data_d[col], data_e[col])))
        feature.append(col)
    df = pd.DataFrame(np.column_stack([feature, p]), columns = ['feature', 'statistic', 'pvalue'])
return df
```

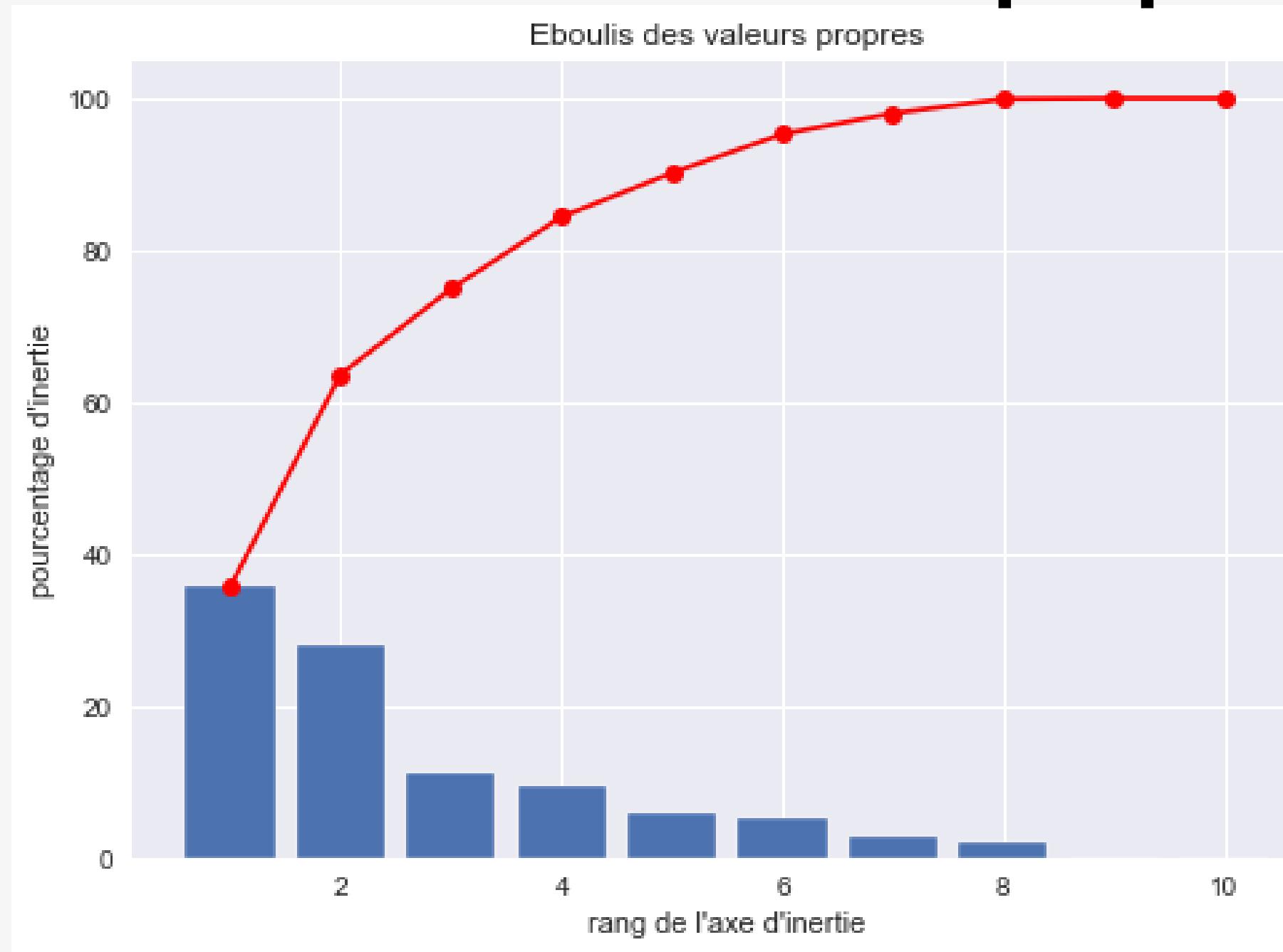
	feature	statistic	pvalue
0	saturated-fat_100g	4120.441018306976	0.0
1	energy_100g	4061.8157183717226	0.0
2	fat_100g	3785.7016517582733	0.0
3	carbohydrates_100g	1567.0977759761533	0.0
4	sodium_100g	767.5272326427312	8.294172707507511e-165
5	sugars_100g	3180.997565471611	0.0
6	fiber_100g	81.49614350891868	8.394091715329268e-17
7	proteins_100g	369.17627695722797	1.2674774737959873e-78
8	salt_100g	762.7417238978051	9.020360342560205e-164
9	nutrition-score-fr_100g	12783.484506754283	0.0
10	calcium_100g	703.5540283362672	5.924942861777134e-151

Valeur de $p \leq 0.05$:
les différences entre certaines médianes sont statistiquement significatives.

Dans mon cas tous les valeurs de P sont inferieur à 0.05 du coup je rejette l'hypothèse nulle et je conclu que toutes les médianes de mes nutritions sont différents. Au moins une médiane est différente

3.5.Analyse multivariable ACP

Eboulis des valeurs propres



Quand on additionne les inerties associées à tous les axes, on obtient l'inertie totale du nuage des individus.

figure : Diagramme des éboulis de valeurs propres avec somme cumulée

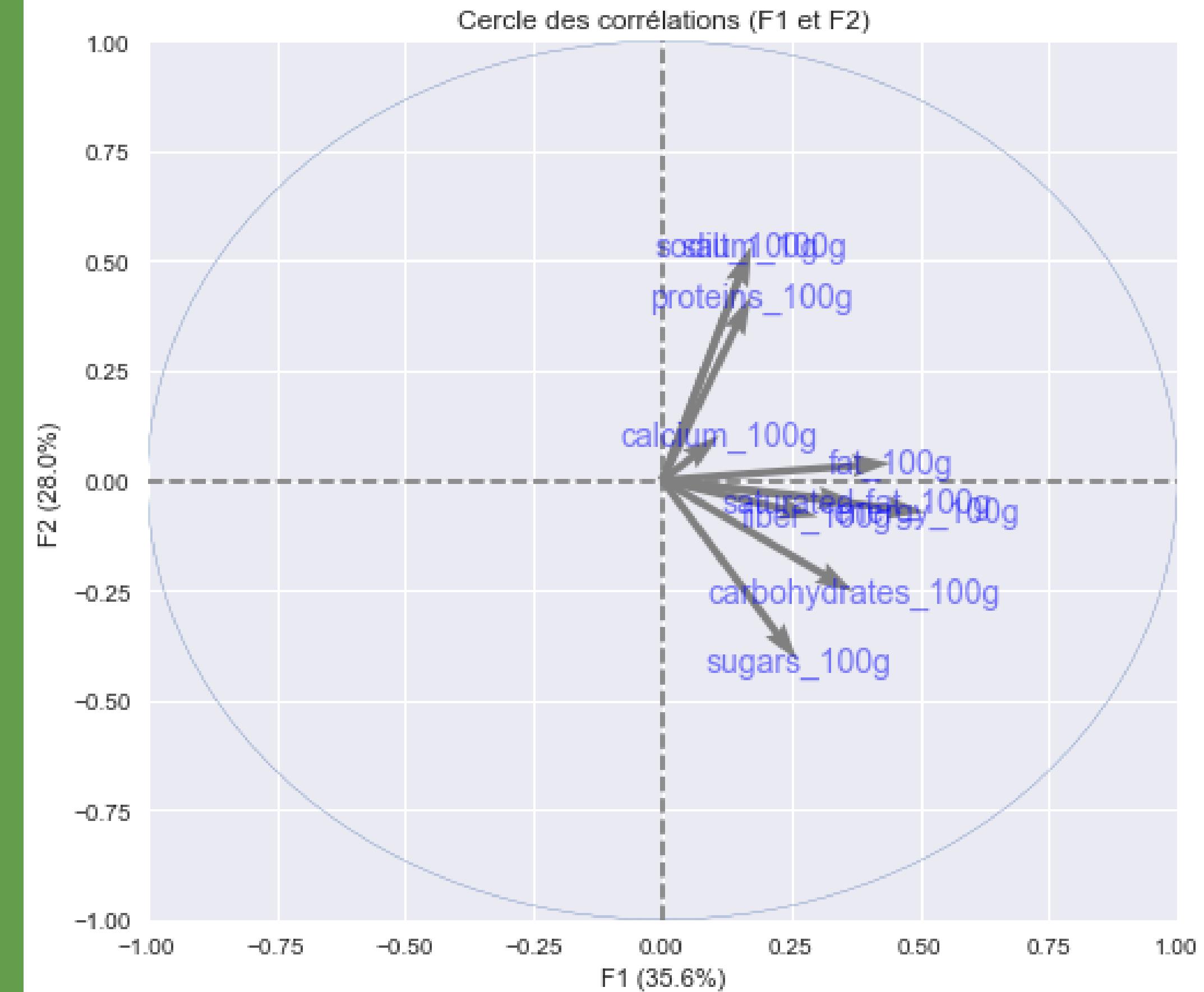
Le cercle des corrélations nous montre les variables qui sont très corrélées (ou anti corrélées.)

F1 peut être considéré comme une nouvelle variable qu'on peut l'ajouté sous forme d'une colonne à notre échantillon.

-F1 est très corrélée avec le gras, fibre, les gras saturés, carbohydrate

-F2 corrélée avec le sodium, les protéines, et le sel

-F2 corrélée négativement avec le sucre



-F3 est corrélées aux variables le fat, saturated fat, calcium

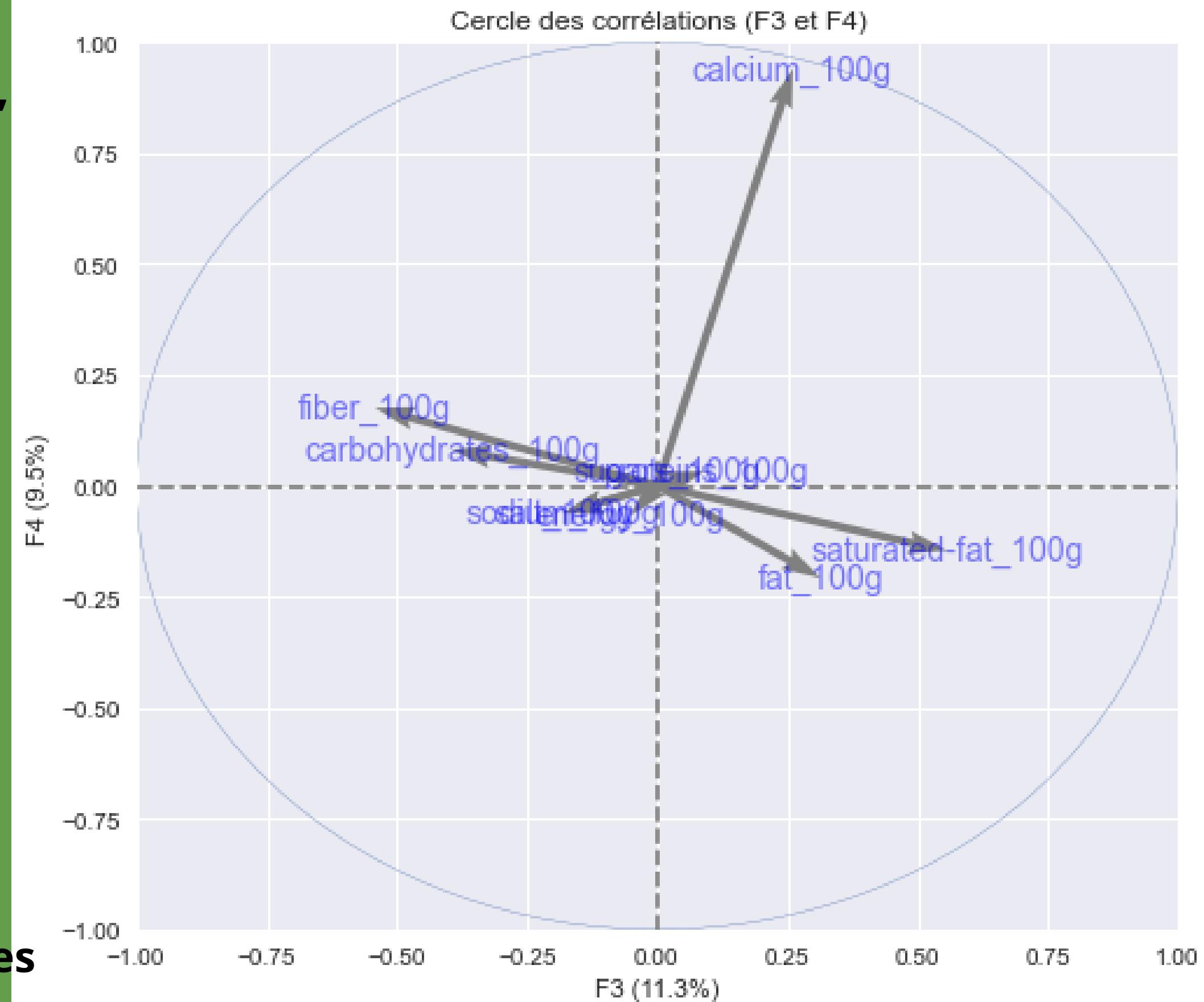
-La variable fibre, carbohydrate est négativement corrélé à F3

-Le calcium est fortement corrélé avec F4

Mes Observations

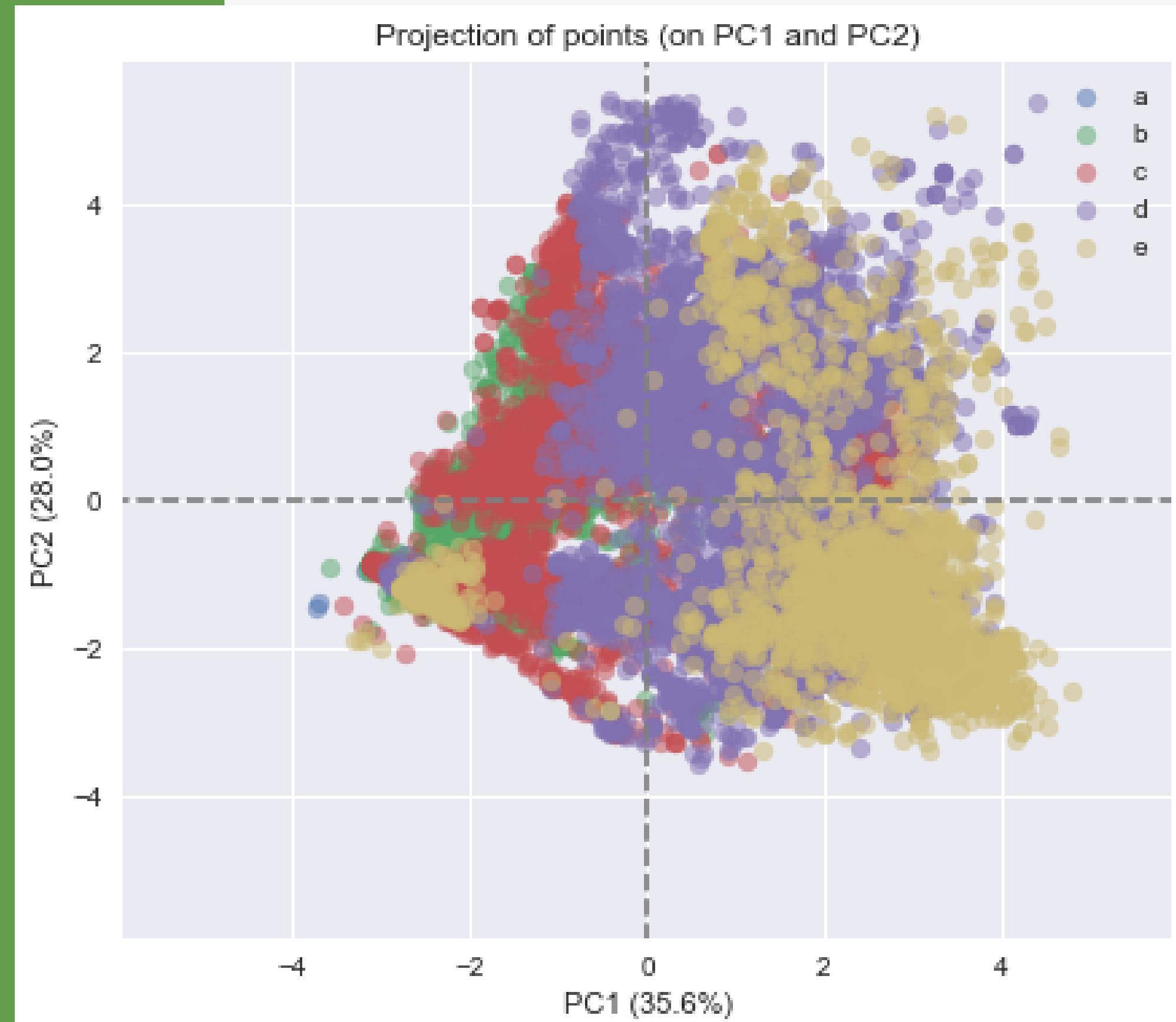
***Non indépendance des donnée**

***Corrélation forte de certaines variables avec le nutriscore**

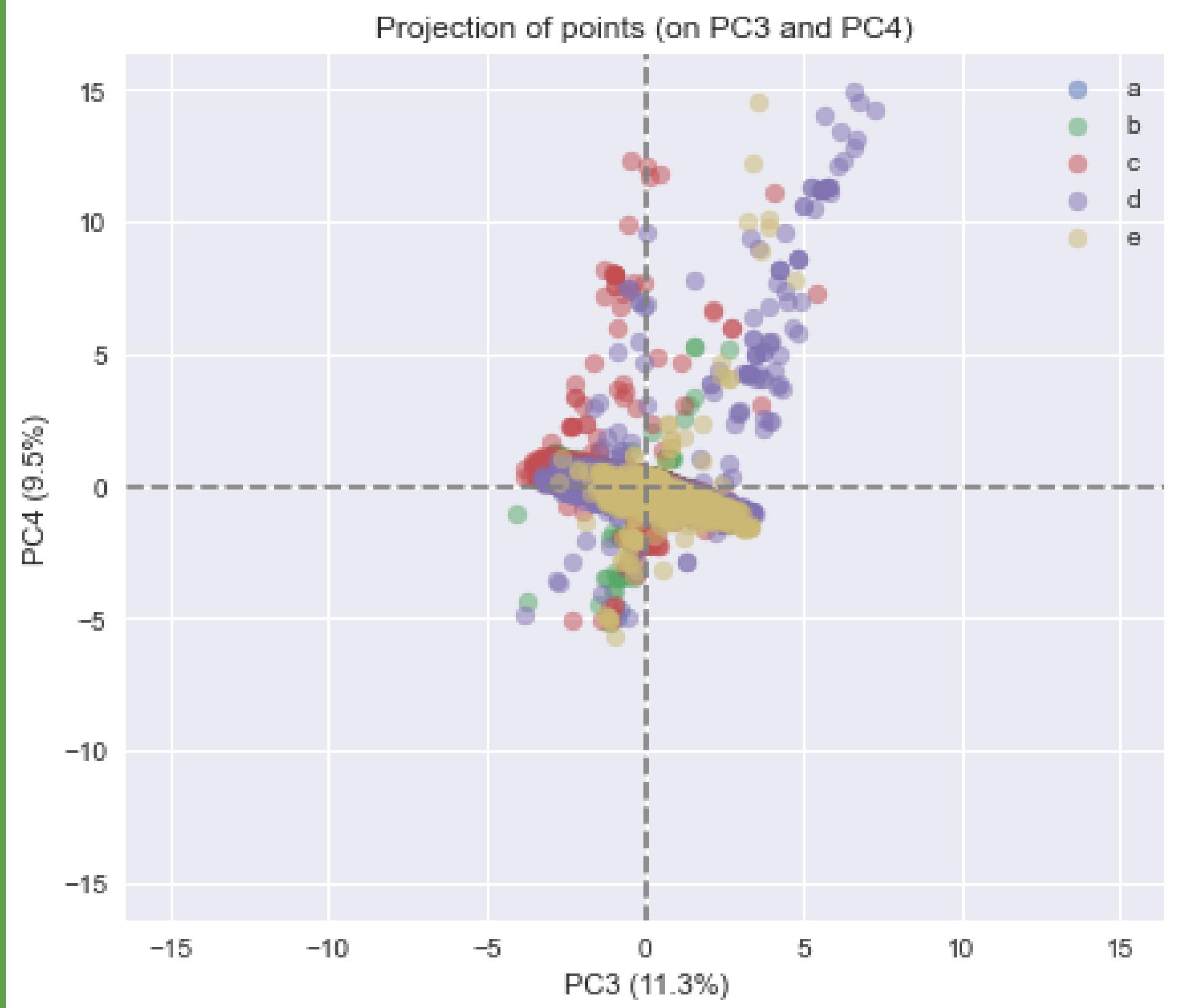


les nutrigrade E est représenté par la couleur jaune : on remarque qu'ils ont une tendance à se regrouper vers les abscisses positives (importantes) alors que les nutrigrade C représenté par la couleur rouge ont tendance à se regrouper vers les abscisses négatives (faibles).

La nutrigrade D représenté par la couleur violet ont tendance à se regroupement dans les deux parties des abscises, une partie vers les abscises faibles et l'autre partie vers l'abscises importantes



F1 représente le gras, le sucre , le sel , le carbohydrate qui sont représentés par des produits de mauvaise qualité (e)



Conclusion



**Dataset Open
Food Facts :**
données de très
bonne qualité
avec u peu de
valeur aberrantes.



Mes Observations
***Non indépendance**
des donnée
***Corrélation forte de**
certaines variables
avec le nutriscore



**Perceptive
(Application):**
Rescoring des
produits en se
basant sur les
features traitées.

MERCI POUR
VOTRE ATTENTION