CONCEVOIR UNE APPLICATION AU SERVICE DE LA SANTÉ PUBLIQUE

Présenté par Check KOUTAME

PLAN

I/ Mission & Description du projet

- Exploration des données du site Open Food Facts & explication de l'idée d'application

II/ Nettoyages des données

- Observations des données: Formes et qualités
- Nettoyages des données :
  - Filtre des données avec les variables d'interêts,
  - Traitements des valeurs aberrantes,
  - Traitements des duplicatas & des NaN
- Imputations des valeurs manquantes: knn, iterative imputer, médiane ou moyenne…

III/ Analyse exploratoire des données

- Analyse univariée & bivariée des données
- Analyse multivariée des données
  - Analyse en composantes principale des données numériques
  - ANOVA sur la corrélation entre les données numériques et catégorielle

Conclusion

I. MISSION & DESCRIPTION DU PROJET:

- Les données du projet
  - Site Open Food Facts
  - comportent plusieurs pays avec un certains nombres de variables d'informations sur les produits alimentaires: nom, date de modification, catégorie, nutriments pour 100 g, pays d'origine, etc…
- Mission
  - Proposer une idée d'application innovante: calcule le niutris-core et le nutri-grade de chaque produit de la base de données et d'insérer ce nutri-grade dans une proposition de conseil pour une **alimentation « saine ».**
- A partir des données d'open Fact Food, il faut dire:
  - La démarche pour traiter les données/ les nettoyer
  - La démarche pour faire une analyse exploratoire de ces données nettoyées
- Objectif:
  - Mise en pratique des démarches de nettoyage et d'analyse exploratoire des données brutes.

II. NETTOYAGE DES DONNÉES: OBSERVATIONS DES DONNES BRUTES

- A/ Observations des données brutes
- B/ Traitement des valeurs aberrantes
- C/Traitement des valeurs manquantes & NaN
- D/Imputations des restes des valeurs manquantes

II. NETTOYAGE DES DONNÉES: A/OBSERVATIONS DES DONNÉES BRUTES

```
1  #Chargement des données brutes
2
3  food_data = pd.read_csv('Data/food_fr.csv',sep='\t',low_memory=False)
```

product_name	generic_name	quantity	...	ph_100g	fruits-vegetables-nuts_100g	collagen-meat-protein-ratio_100g	cocoa_100g	chlorophyl_100g	carbon-footprint_100g	nutrition-score-fr_100g	nutrition-score-uk_100g	gl inde
Farine de blé noir	NaN	1kg	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
Banana Chips Sweetened (Whole)	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	14.0	14.0	
Peanuts	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	
Organic Salted Nut Mix	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	12.0	12.0	
Organic Polenta	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	
Tomato & ricotta	NaN	1	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
Mint Melange Tea A Blend Of Peppermint, Lemon ...	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	
乐吧泡菜味薯片	Leba pickle flavor potato chips	50 g	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

- **320772 produits et 162 variables**

II. NETTOYAGE DES DONNÉES: A/OBSERVATIONS DES DONNÉES BRUTES

- Définitions de quelques fonctions permettant d'observer les données brutes

```
def vue_general_variable(data):
    #Fonction qui prend en entrée un dataFrame et qui fait un résumé des Variables de cette dataFrame:
    #le type, etc...
```

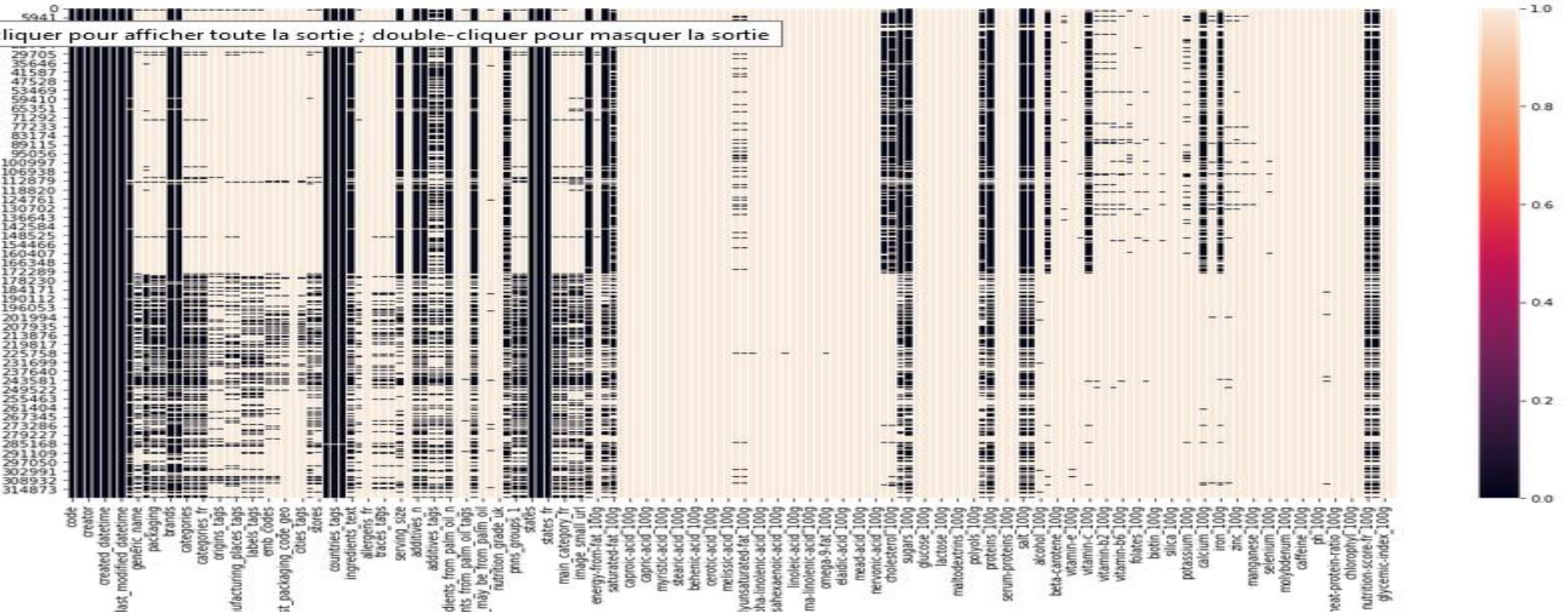
```
def plot_NA(data: pd.DataFrame):
    # Fonction qui prend en entrée un dataFrame et qui affiche en histogramme les pourcentages des valeurs
```

```
Entrée [32]:  1  vue_generale_data(food_data)# Caratéi
```

```
-----------------------------------------------
Donnée : ['_', 'food_data', '_20']
Nombre de variable : 162
Nomres de types de variables : float64    1(
object      56
dtype: int64
Nombre observation : 320772
Nombre de cellules manquantes : 39608589
% de cellules manquantes : 76.22%
Nombre de lignes dupliquées : 0
% de lignes dupliquées : 0.00%
```

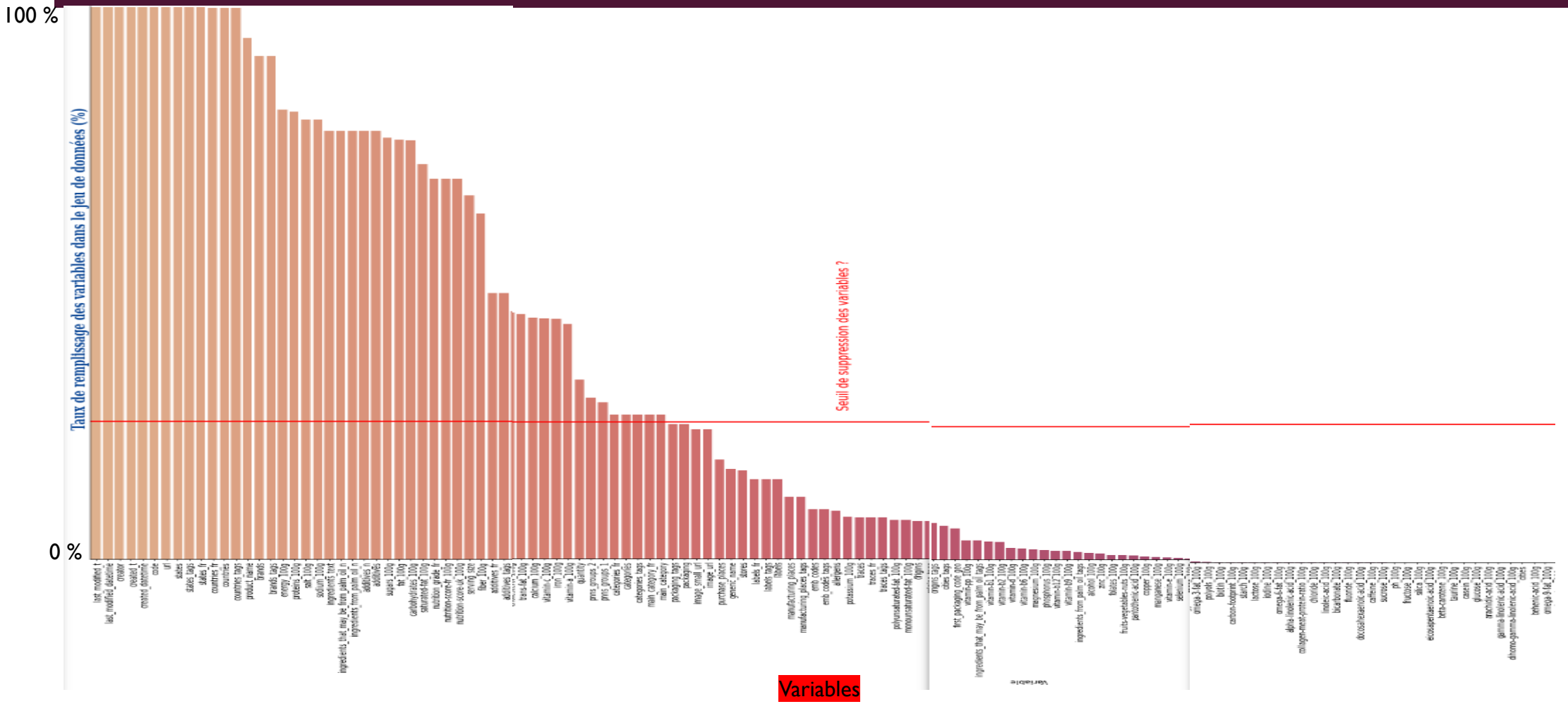
Nom variable	Type de la variable	nbre observation	Valeurs manquantes	% Valeurs manquantes	Moyenne	Mediane	Variance
code	object	320750	23	0.000072			
url	object	320750	23	0.000072			
creator	object	3536	2	0.000006			
created_t	object	189568	3	0.000009			
reated_datetime	object	189569	9	0.000028			
...	...	...	...	...	...	...	...
carbon-footprint_100g	float64	203	320504	0.999165	341.700764	195.75	180130.123183
nutrition-score-fr_100g	float64	56	99562	0.310382	9.165535	10.0	82.009007
nutrition-score-uk_100g	float64	56	99562	0.310382	9.058049	9.0	84.33793

II. NETTOYAGE DES DONNÉES:
A/OBSERVATIONS DES DONNÉES BRUTES



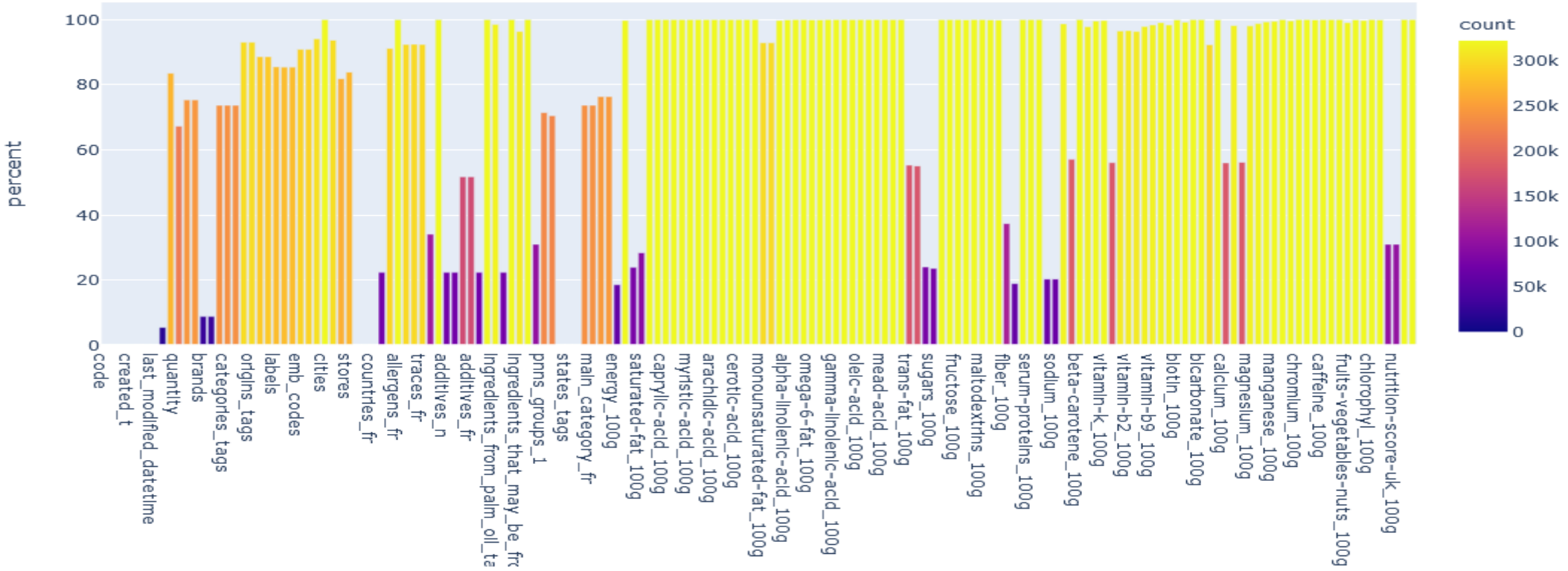
Heatmaps: aperçu de l'ensemble des données

II. NETTOYAGE DES DONNÉES:
A/OBSERVATIONS DES DONNÉES BRUTES



Taux de remplissage des données: aperçu de l'ensemble des données

II. NETTOYAGE DES DONNÉES:
A/OBSERVATIONS DES DONNÉES BRUTES



% des données manquantes: aperçu de l'ensemble des données

10 catégories sont représentées dans la variable pnns_group_1.

II. NETTOYAGE DES DONNÉES: A/OBSERVATIONS DES DONNÉES PAR CATÉGORIES

- 2 variables importantes qui permettent de voir les type de produits dans le jeu de données

```
# nombres de catégories dans les groupes pnns 1 & 2
pnns_groups_1 = split_words(df = food_data, column = 'pnns_groups_1')
pnns_groups_2 = split_words(df = food_data, column = 'pnns_groups_2')
print("{} catégories sont représentées dans la variable pnns_group_1.".format(len(pnns_groups_1)))
print("{} catégories sont représentées dans la variable pnns_group_2.".format(len(pnns_groups_2)))
```

- **14 catégories sont représentées dans la variable pnns_group_1.**
- **42 catégories sont représentées dans la variable pnns_group_2.**

Groupes pnns_groupe_1

```
['sugary-snacks',
 'Cereals and potatoes',
 'fruits-and-vegetables',
 'Composite foods',
 'salty-snacks',
 'Salty snacks',
 'Fish Meat Eggs',
 'unknown',
 'Beverages',
 'Fat and sauces',
 'Sugary snacks',
 'Milk and dairy products',
 'cereals-and-potatoes',
 'Fruits and vegetables']
```

- certaines catégories sont présentes plusieurs fois mais orthographiées différemment :
  - 'Cereals and potatoes' et 'cereals-and-potatoes
  - 'fruits-and-vegetables' et 'Fruits and vegetables', etc…

- corriger le problème en passant le texte en minuscule et en remplaçant les caractères spéciaux par un espace :

```
food_data["pnns_groups_1"] = food_data["pnns_groups_1"].str.lower().str.replace('-', ' ')
pnns_groups_1 = split_words(df = datas, column = 'pnns_groups_1')
print("{} catégories sont représentées dans la variable pnns_group_1.".format(len(pnns_groups_1)))
print(pnns_groups_1)
```

-> 10 catégories sont représentées dans la variable pnns_group_1.

['beverages', 'fish meat eggs', 'sugary snacks', 'fruits and vegetables', 'salty snacks', 'cereals and potatoes', 'unknown', 'milk and dairy products', 'composite foods', 'fat and sauces']

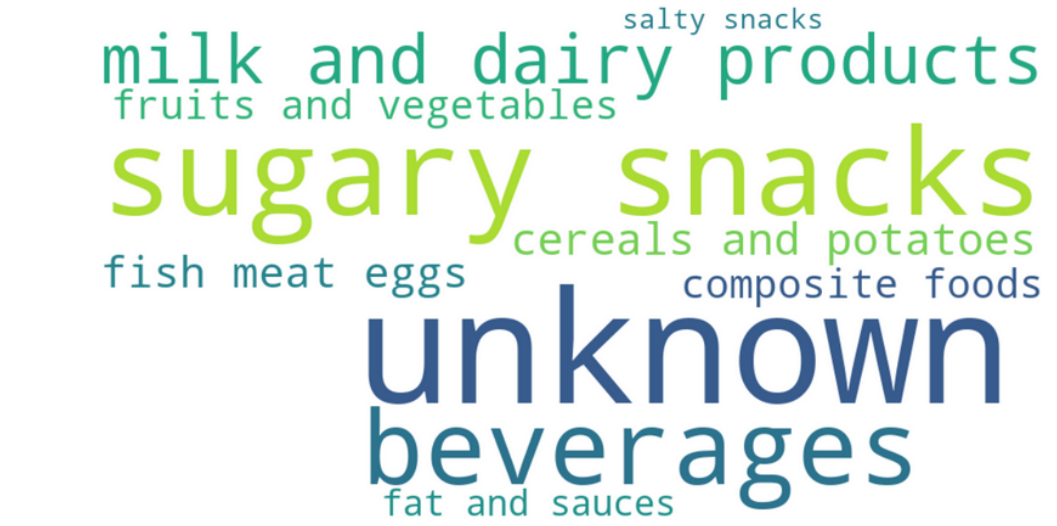
10 catégories sont représentées dans la variable pnns_group_1.

II. NETTOYAGE DES DONNÉES: A/OBSERVATIONS DES DONNÉES PAR CATÉGORIES

Groupes pnns_groupe_1

```
Entrée [46]: 1 plot_world_cloud(df=datas,column="pnns_groups_1",nb_top=10)
```

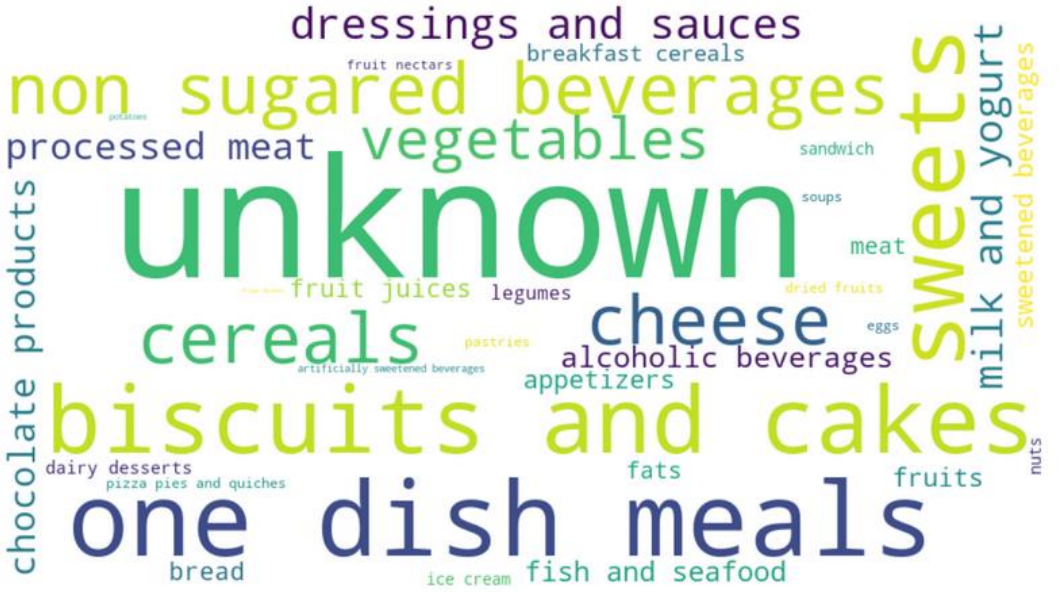
Nuage de mots des 10 meilleures pnns_groups_1



Groupes pnns_groupe_2

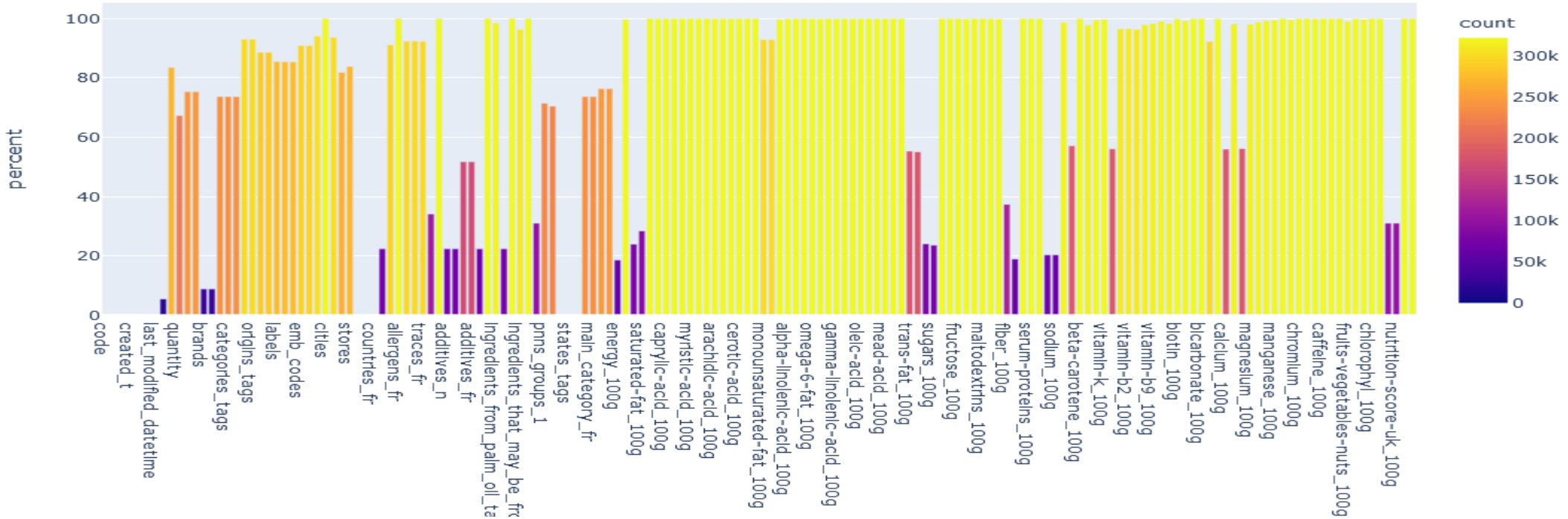
```
Entrée [47]: 1 plot_world_cloud(df=datas,column="pnns_groups_2",nb_top=40)
```

Nuage de mots des 40 meilleures pnns_groups_2



II. NETTOYAGE DES DONNÉES:
B/ FILTRE DES DONNÉES ET VARIABLES D'INTÉRÊT

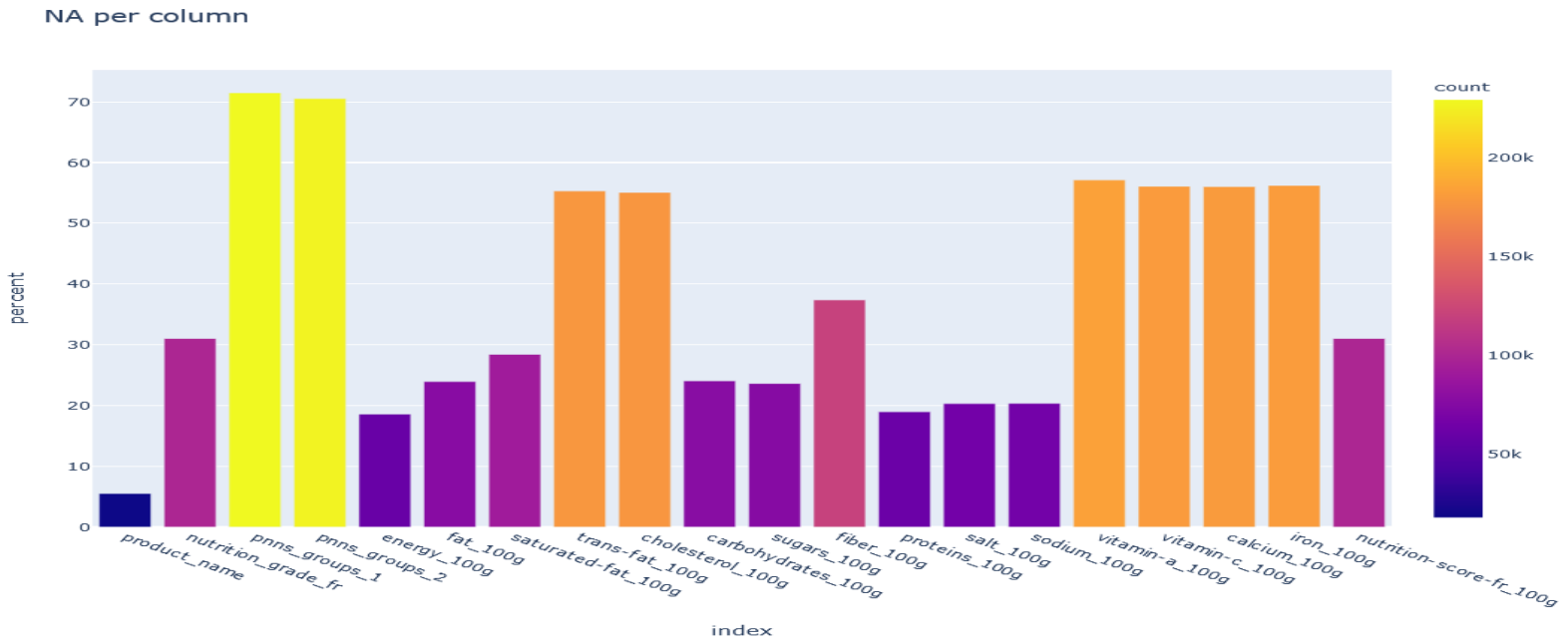
- L'application permet de noter un aliment sur sa composition nutritionnelle sur 100 g
  - Garder les variables finissant par _100g et
  - Appliquer un premier filtre qui enlèvera toutes les variables qui ont + de 80% de données manquantes



% des données manquantes: aperçu de l'ensemble des données

II. NETTOYAGE DES DONNÉES:
B/ FILTRE DES DONNÉES ET VARIABLES D'INTÉRÊT

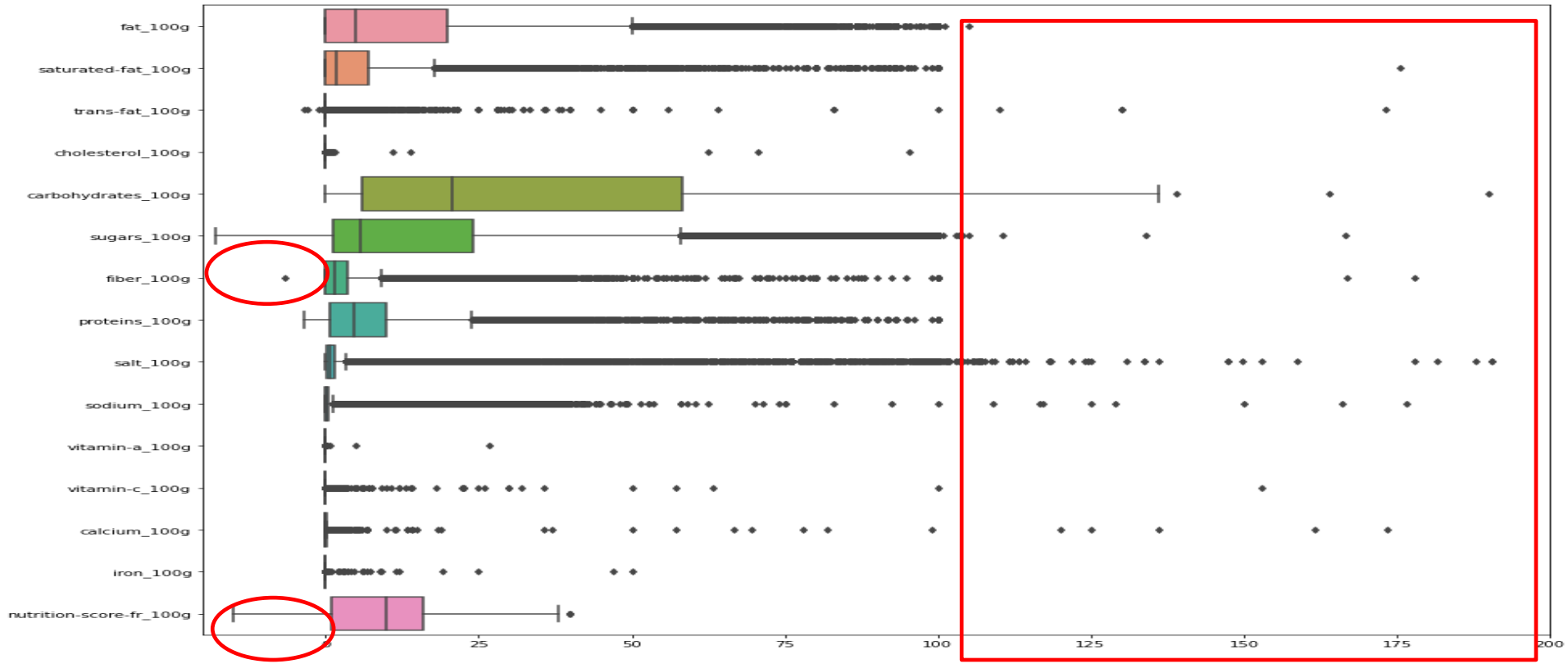
- L'application permet de noter un aliment sur sa composition nutritionnelle sur 100 g
  - Garder que les variables finissant par _100g et
  - Appliquer un premier filtre qui enlèvera toutes les variables qui ont + de 80% de données manquantes



% des données manquantes: aperçu des données après un premier filtre

II. NETTOYAGE DES DONNÉES:
B/ TRAITEMENT DES VALEURS ABERRANTES

- Affichage des variables d'interêts pour observer toutes nos données



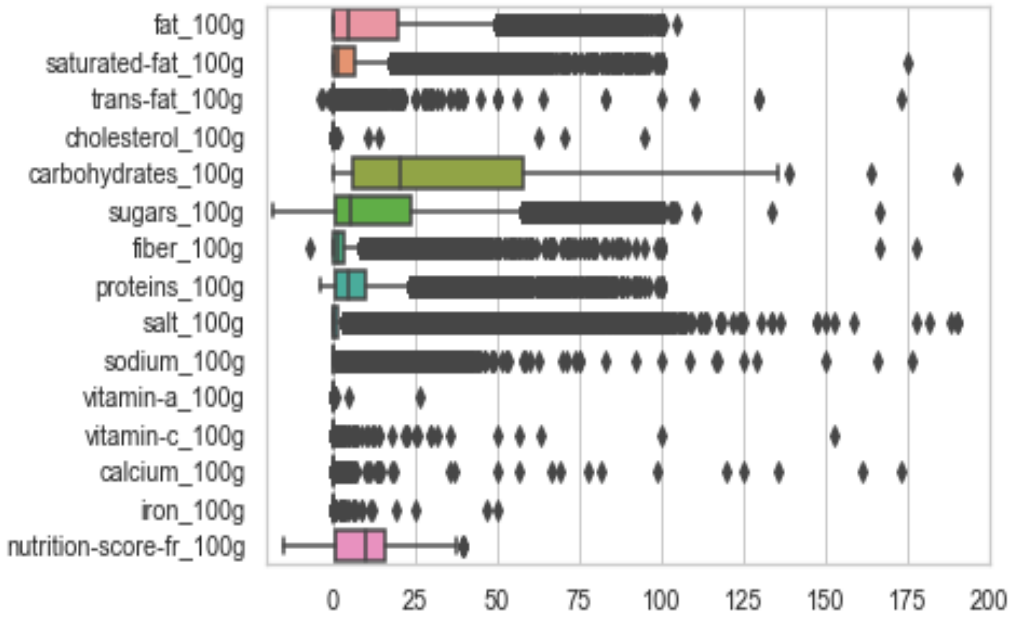
Boxplot de nos variables d'interêts: aperçu de l'ensemble des données

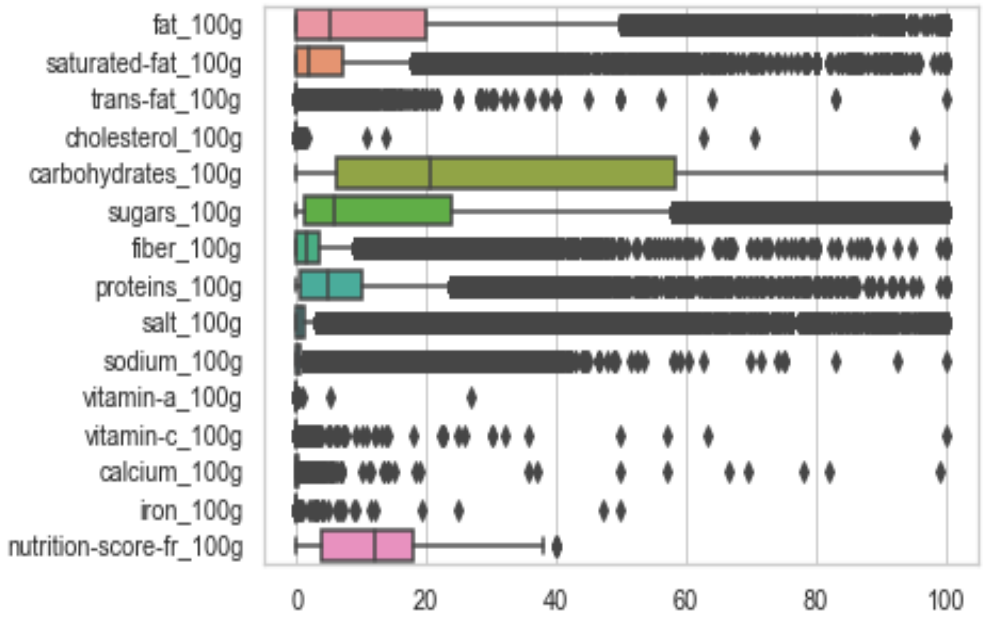
II. NETTOYAGE DES DONNÉES:
B/ TRAITEMENT DES VALEURS ABERRANTES

- Pour le traitement des valeurs aberrantes, nous allons comparer 2 méthodes:
  - La méthode basée sur la règle métier
  - La méthode des interquartiles

II. NETTOYAGE DES DONNÉES: B/ TRAITEMENT DES VALEURS ABERRANTES

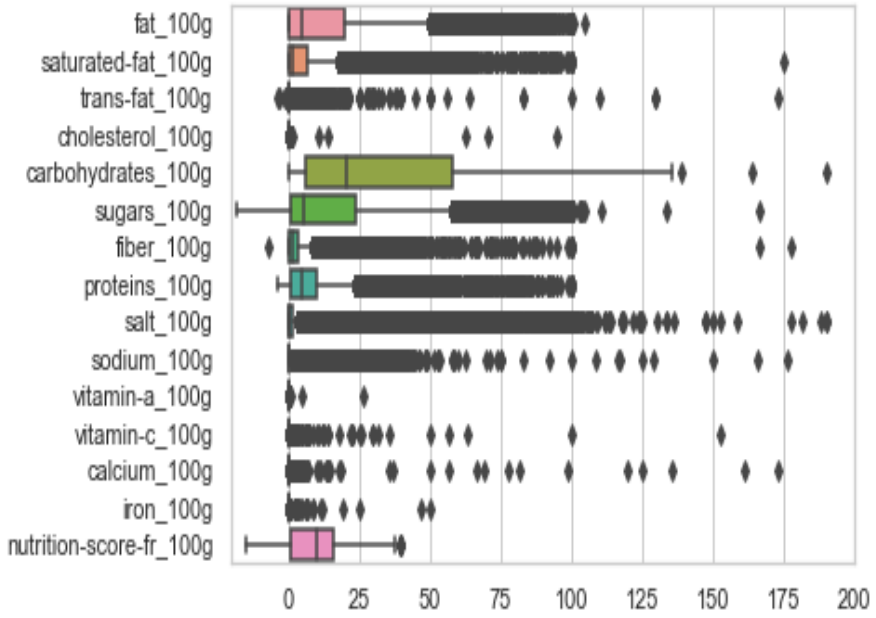
- Pour le traitement des valeurs aberrantes, nous allons comparer 2 méthodes:
  - **La méthode basée sur la règle métier**
    - Il s'agit ici de supprimer toutes les valeurs des nutriments qui sont inférieures à 0 et supérieures à 100 g
    - Création d'une fonction permettant de ne garder que les valeurs respectant cette règle (delete_Outliers)

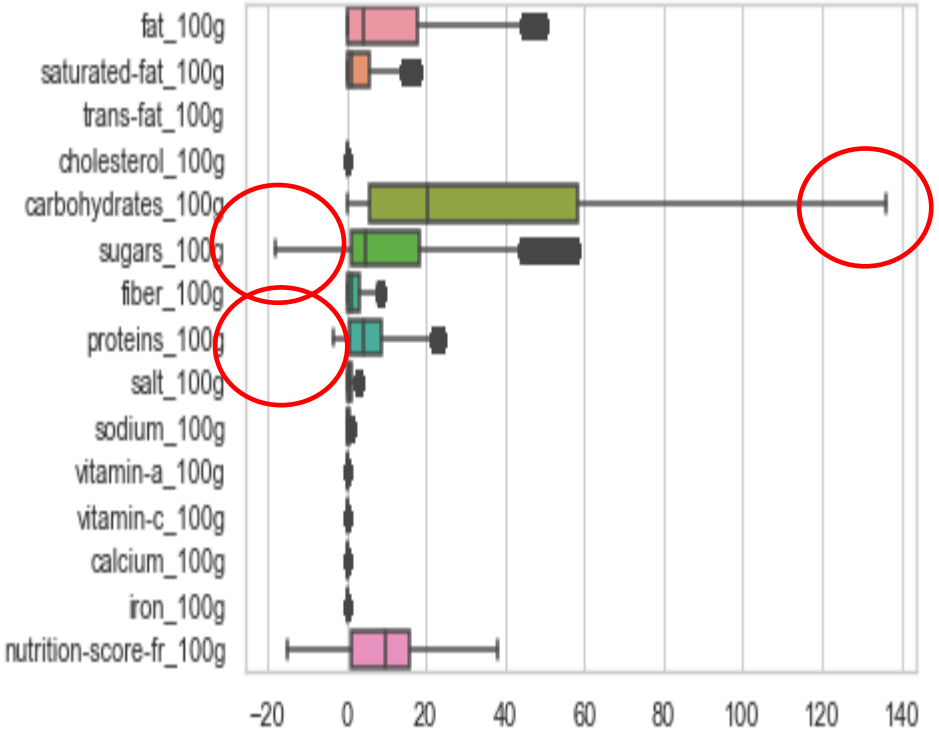




II. NETTOYAGE DES DONNÉES: B/ TRAITEMENT DES VALEURS ABERRANTES

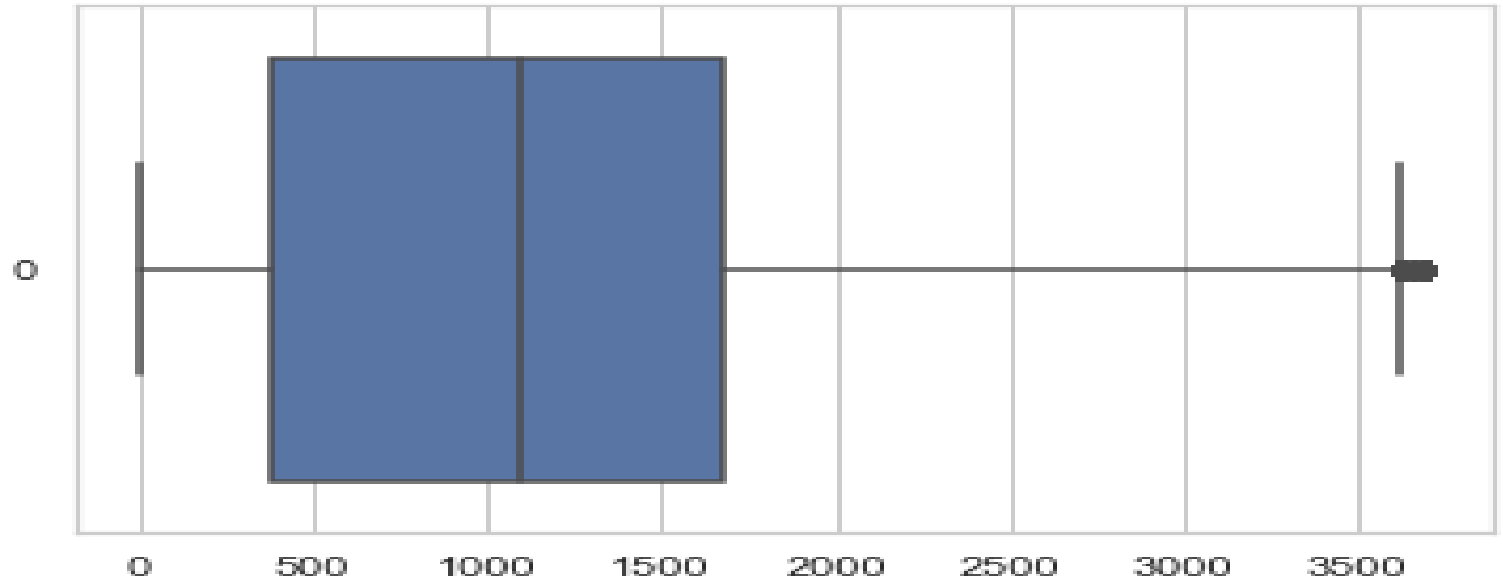
- Pour le traitement des valeurs aberrantes, nous allons comparer 2 méthodes:
  - **La méthode basée sur l'interquartile**
    - Il s'agit ici de supprimer toutes les valeurs des nutriments qui sont inférieure à 1,5*IQ et supérieure à 1,5*IQ
    - Création d'une fonction permettant de ne garder que les valeurs respectant cette règle (delete_Interquartile)





II. NETTOYAGE DES DONNÉES: B/ TRAITEMENT DES VALEURS ABERRANTES

- Pour le traitement des valeurs aberrantes, nous allons comparer 2 méthodes:
  - **On garde donc la méthode basée sur la règle metier**
    - **Et on fait un filtre sur la valeur de l'énergie sur 100g en sachant que la valeur maximale doit être de 3700 J**



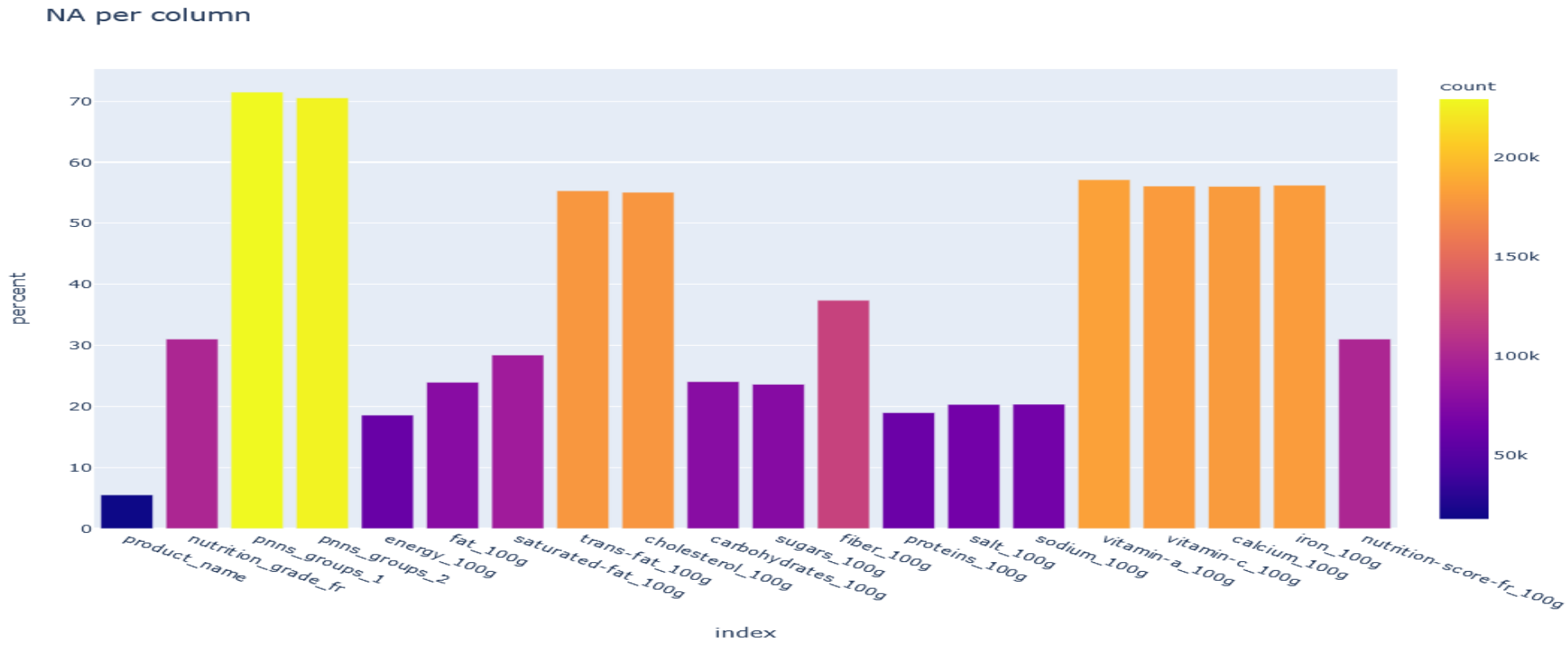
Energie_100g

II. NETTOYAGE DES DONNÉES: C/ TRAITEMENT DES VALEURS DUPLIQUÉES

- Utilisation de la fonction drop_duplicates pour supprimer tous les produits dupliqués, en utilisant comme index la variables « created_t » qui ne comporte aucune valeur manquante.
  - Passage des 320139 produits à 220931

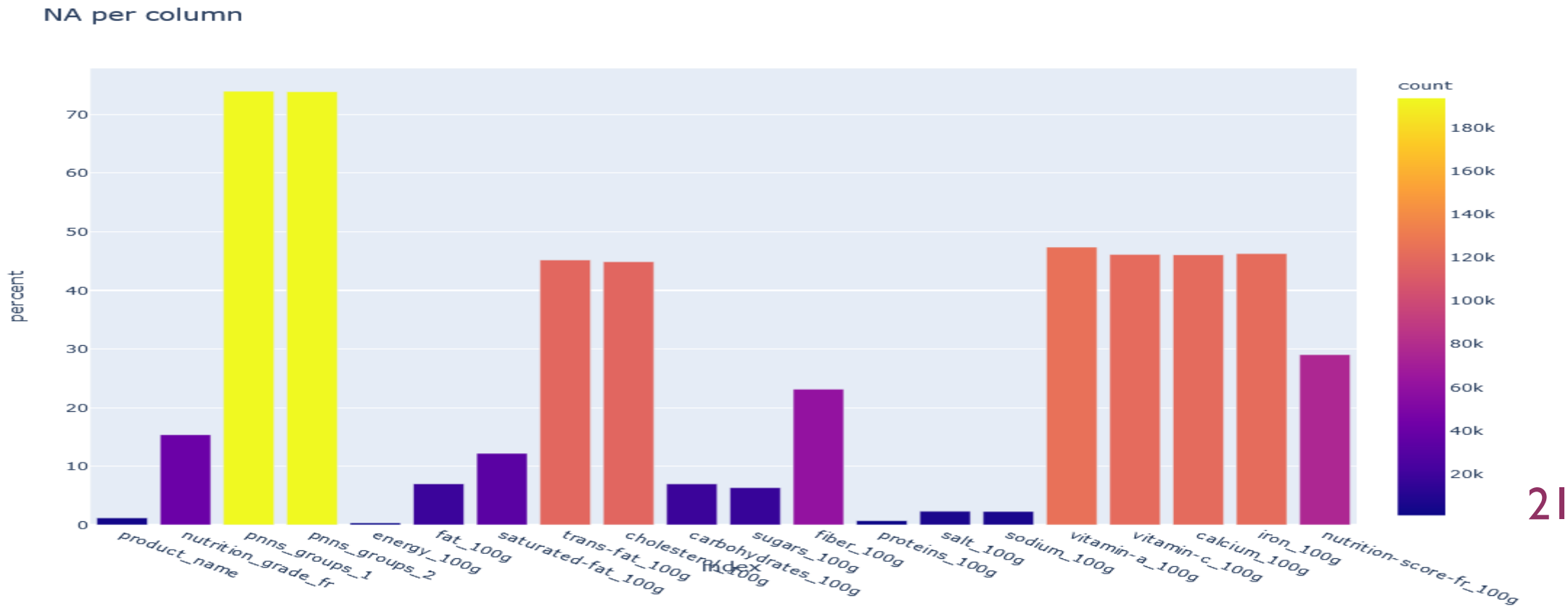
II. NETTOYAGE DES DONNÉES: C/ TRAITEMENT DES VALEURS MANQUANTES (LIGNE ET COLONNES)

- Supprimer les produits sans categories et sans Nutriscores
- Supprimer les lignes ou colonnes avec 80% de valeurs manquantes



II. NETTOYAGE DES DONNÉES:
C/ TRAITEMENT DES VALEURS MANQUANTES (LIGNES ET COLONNES)

- Supprimer les produits sans catégories et sans Nutriscores
- Supprimer les lignes ou colonnes avec 80% de valeurs manquantes



II. NETTOYAGE DES DONNÉES: D/ TRAITEMENT DES NAN

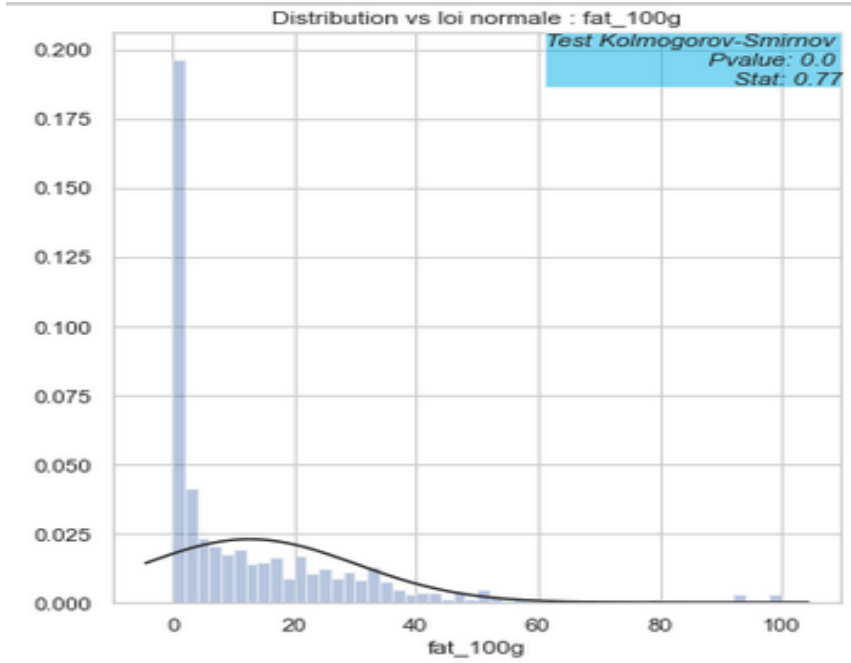
- Pour le traitement des Valeurs NaN restantes, plusieurs méthodes s'offrent à nous
  - **Remplacer par la moyenne,**
  - **Remplacer par la médiane**
  - **Ou encore faire des algorithme de remplissage**
    - **Knn**
    - **Iterative computer**

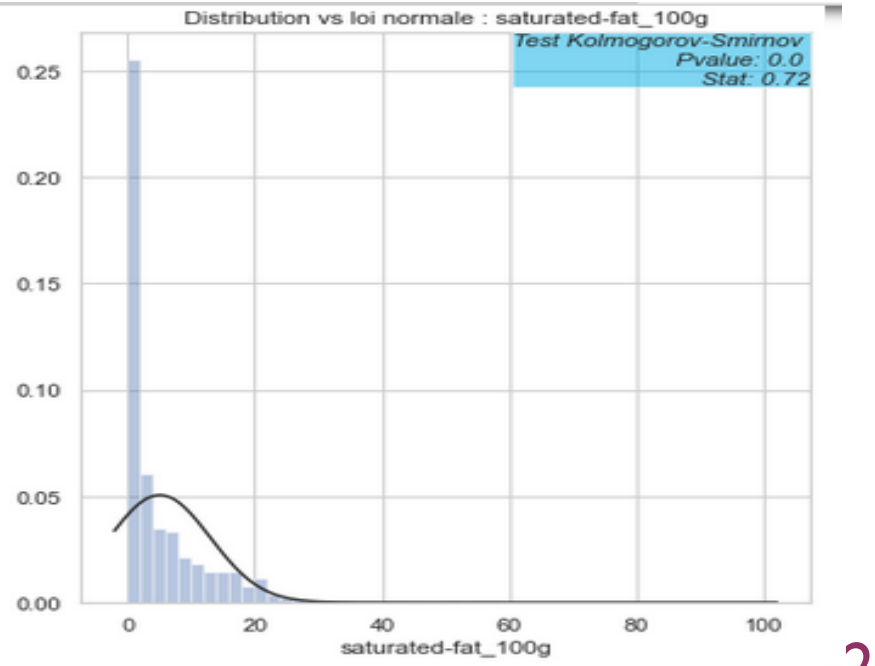
II. NETTOYAGE DES DONNÉES: D/ TRAITEMENT DES NAN: NORMALITÉS DES DISTRIBUTIONS

- Pour le traitement des Valeurs NaN restantes, plusieurs méthodes s'offrent à nous
  - **Test de normalités sur les données**

- Si la valeur p > alpha
L'échantillon suit une loi normale
(on ne rejète pas l'hypothèse H0)

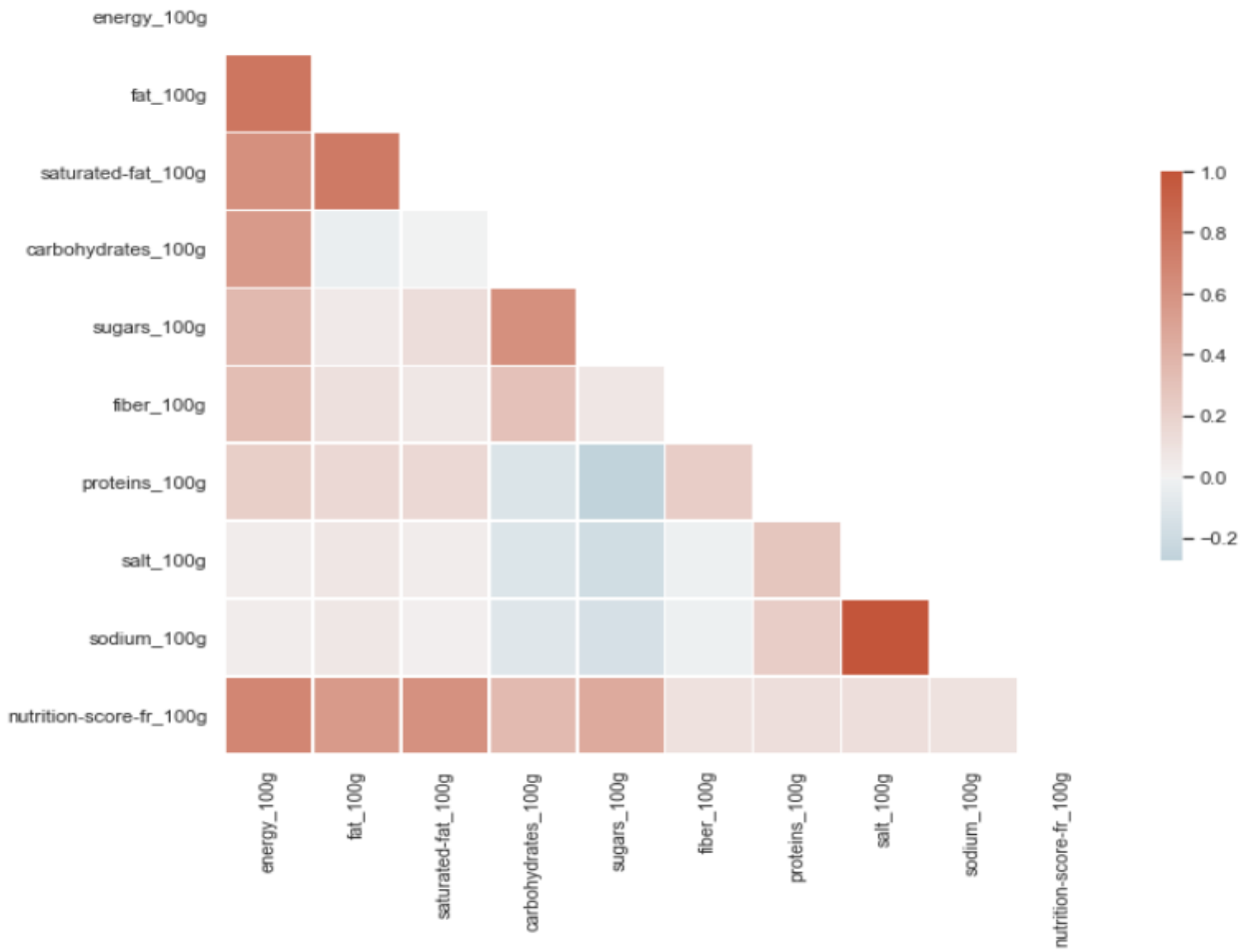
- Si la valeur p < alpha
L'échantillon suit une loi normale
(on rejète l'hypothèse H0)





Sur nos données les valeurs ne suivent pas une loi normale: On ne peut pas se permettre de remplacer par la moyenne

II. NETTOYAGE DES DONNÉES:
D/ TRAITEMENT DES NAN: CORRÉLATION DES VARIABLES



Triangle de corrélation des différentes variables

II. NETTOYAGE DES DONNÉES: D/ TRAITEMENT DES NAN

- Pour l'imputation des valeurs, nous allons utiliser
  - **La valeur médiane** sur les variables: 'carbohydrates_100, cholesterol_100g, fat_100g, fiber_100g, vitamine-c_100g… en remplaçant par la médiane de valeurs homogènes, c'est-à-dire sur le groupe pnns_groups_2.
  - **Un algorithme kNN** sur les variables : energy_100g, proteins_100g ,saturated-fat_100g ,sugars_100g
  - **Iteractive Imputer** sur les variables salt_100g et sodium_100g

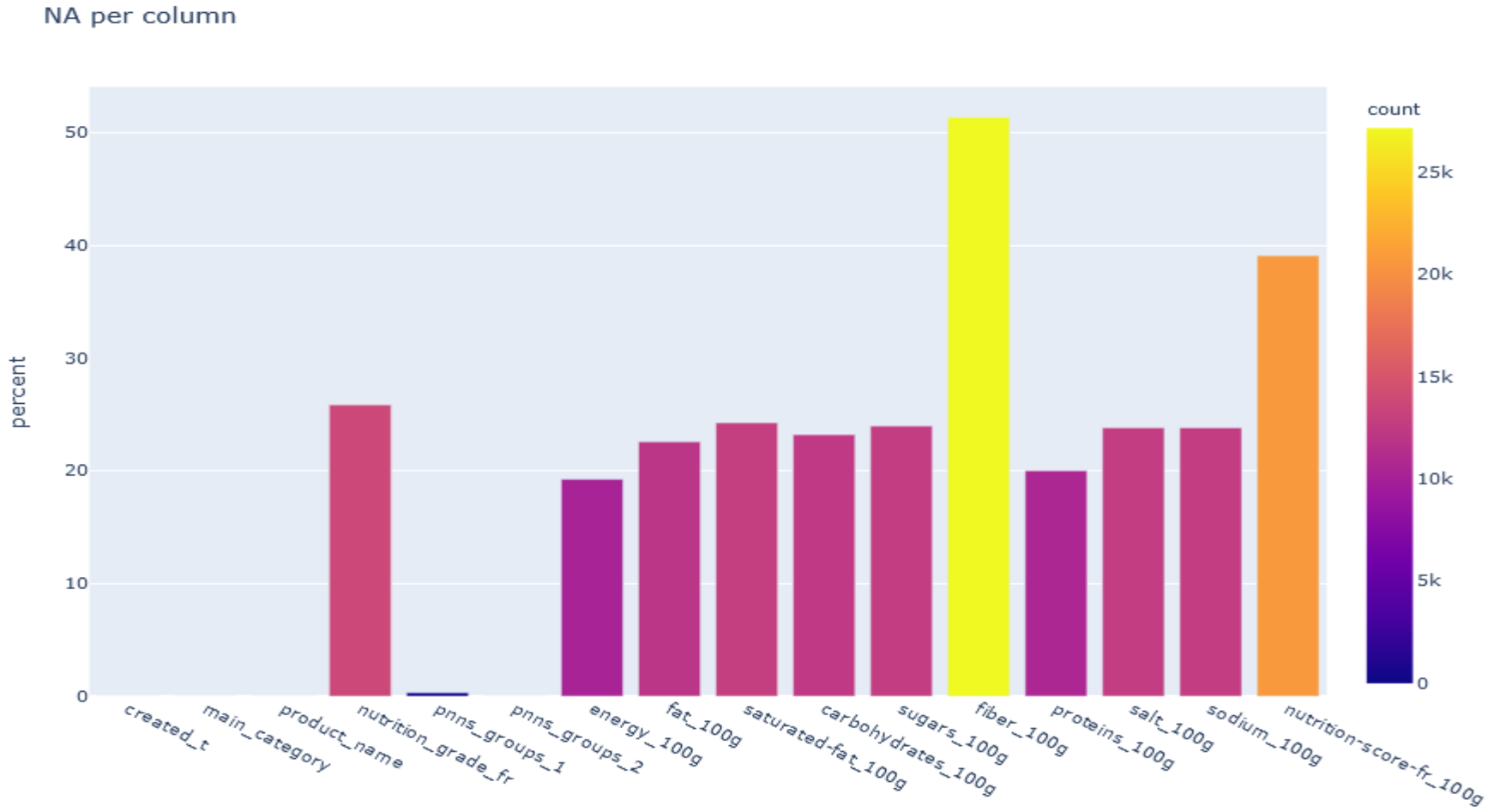
II. NETTOYAGE DES DONNÉES: D/ TRAITEMENT DES NAN:

- Pour l'imputation des valeurs, nous allons utiliser
  - **La valeur médiane** sur les variables: 'carbohydrates_100, cholesterol_100g, fat_100g, fiber_100g.
  - Dans ce cas, on utilise des données homogènes en se référant bien au groupe pnns_group2
    - On calcule la médiane des pour l'ensemble des produits ayant la même catégorie, et on remplace les produits ayant une NaN par cette médiane

```
# On complète les variables suivantes avec la médiane du groupe pnns 2
for col in ['carbohydrates_100g','fat_100g','fiber_100g','iron_100g','vitamin-c_100g',
                                    'vitamin-a_100g','calcium_100g','iron_100g']:
    data_avec_catg_Nut[col] = data_avec_catg_Nut.groupby('pnns_groups_2')[col].apply(lambda
                                                                    x:x.fillna(x.median()))
```

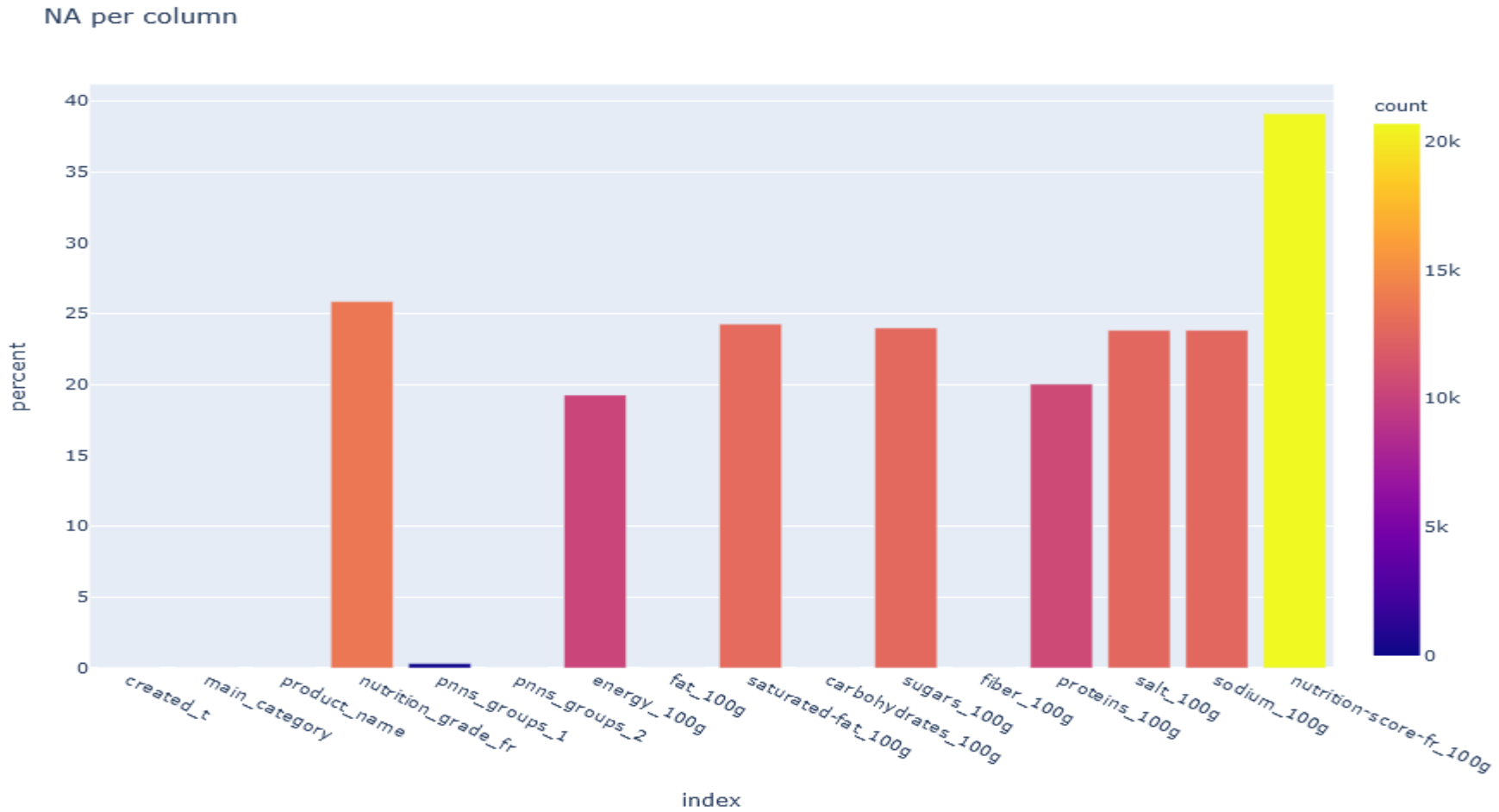
II. NETTOYAGE DES DONNÉES: D/ TRAITEMENT DES NAN:

- Imputation par la médiane: avant



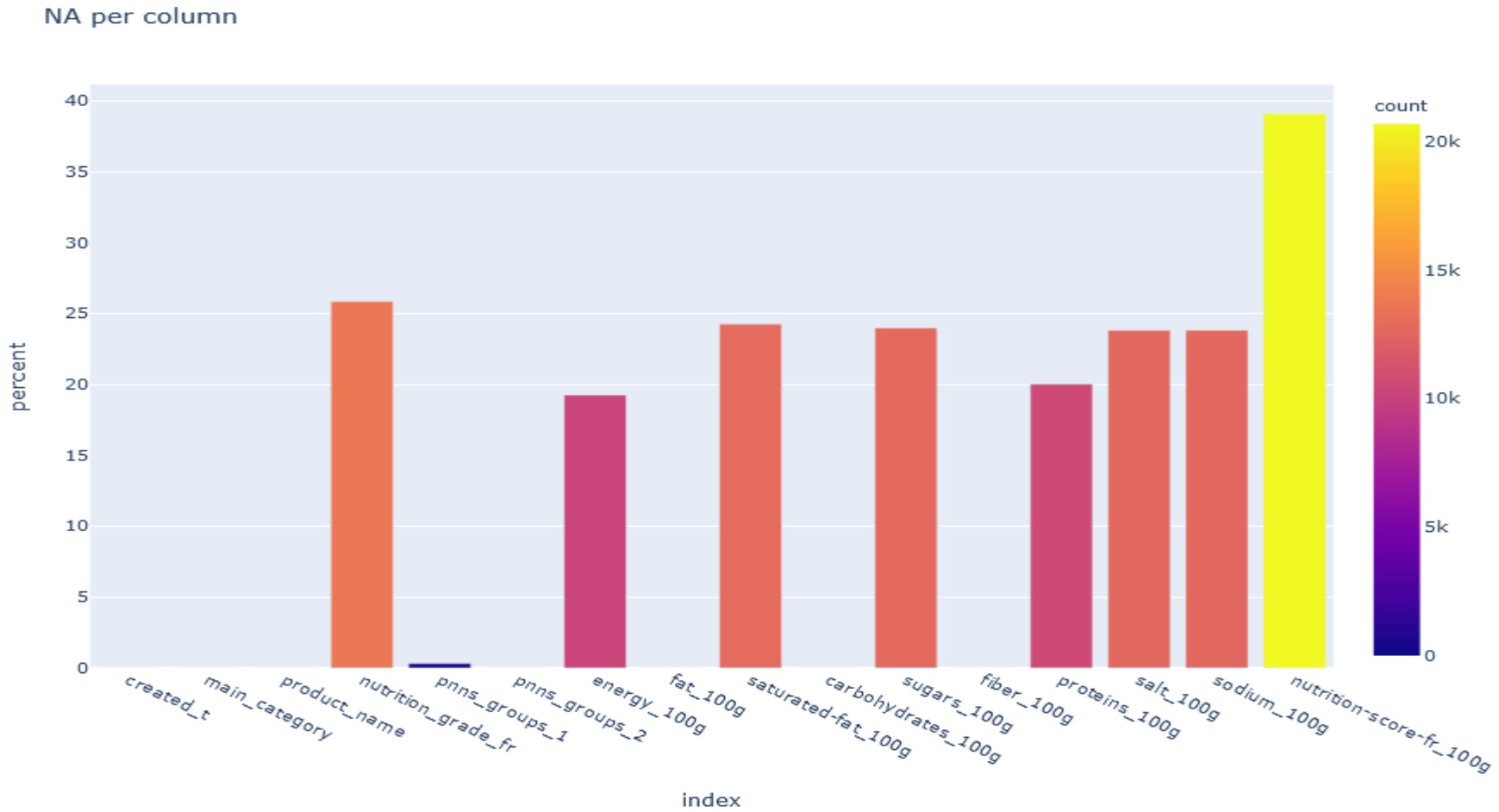
II. NETTOYAGE DES DONNÉES: D/ TRAITEMENT DES NAN:

- Imputation par la médiane: après



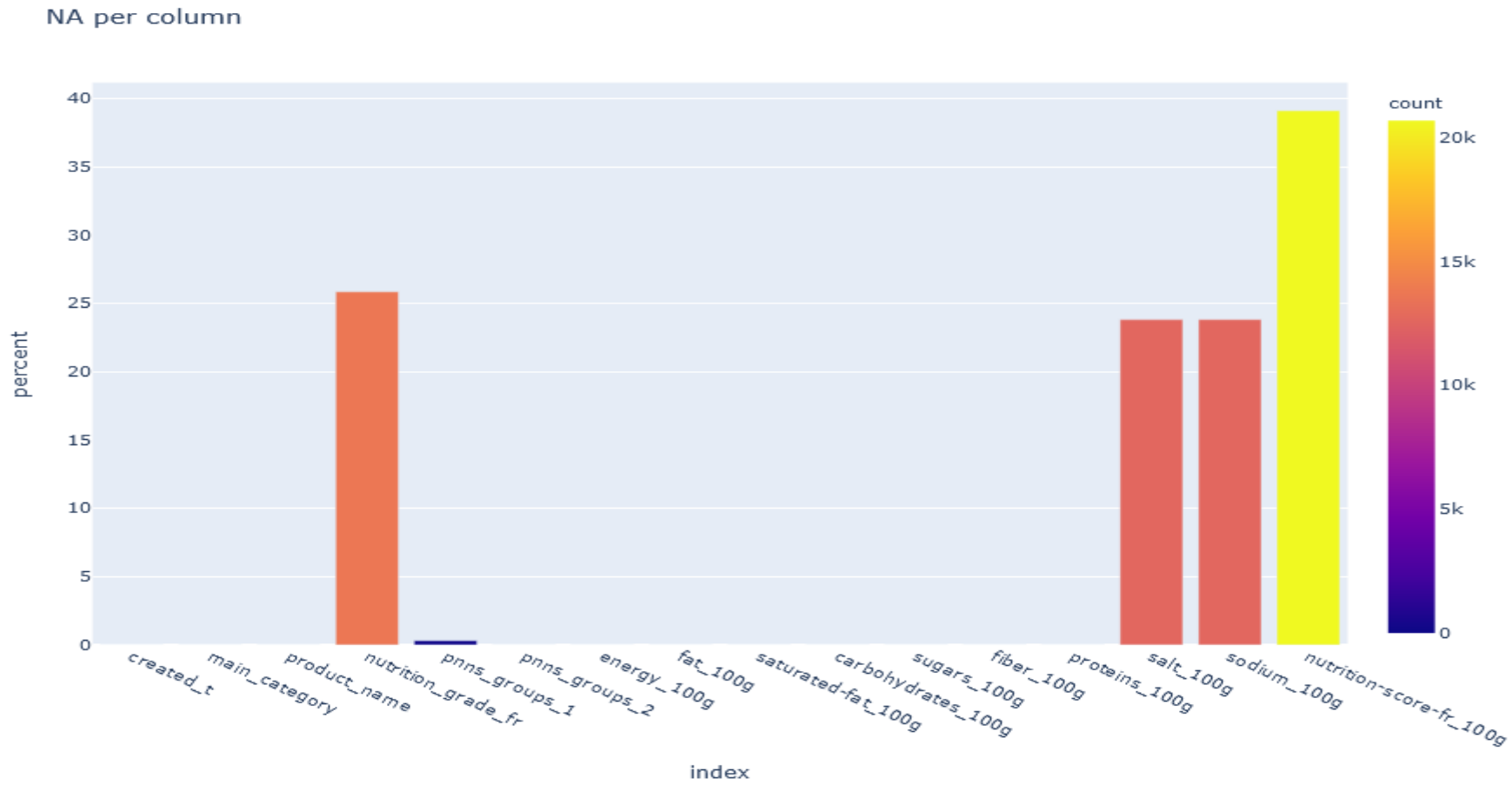
II. NETTOYAGE DES DONNÉES: D/ TRAITEMENT DES NAN:

- Imputation par knn: avant



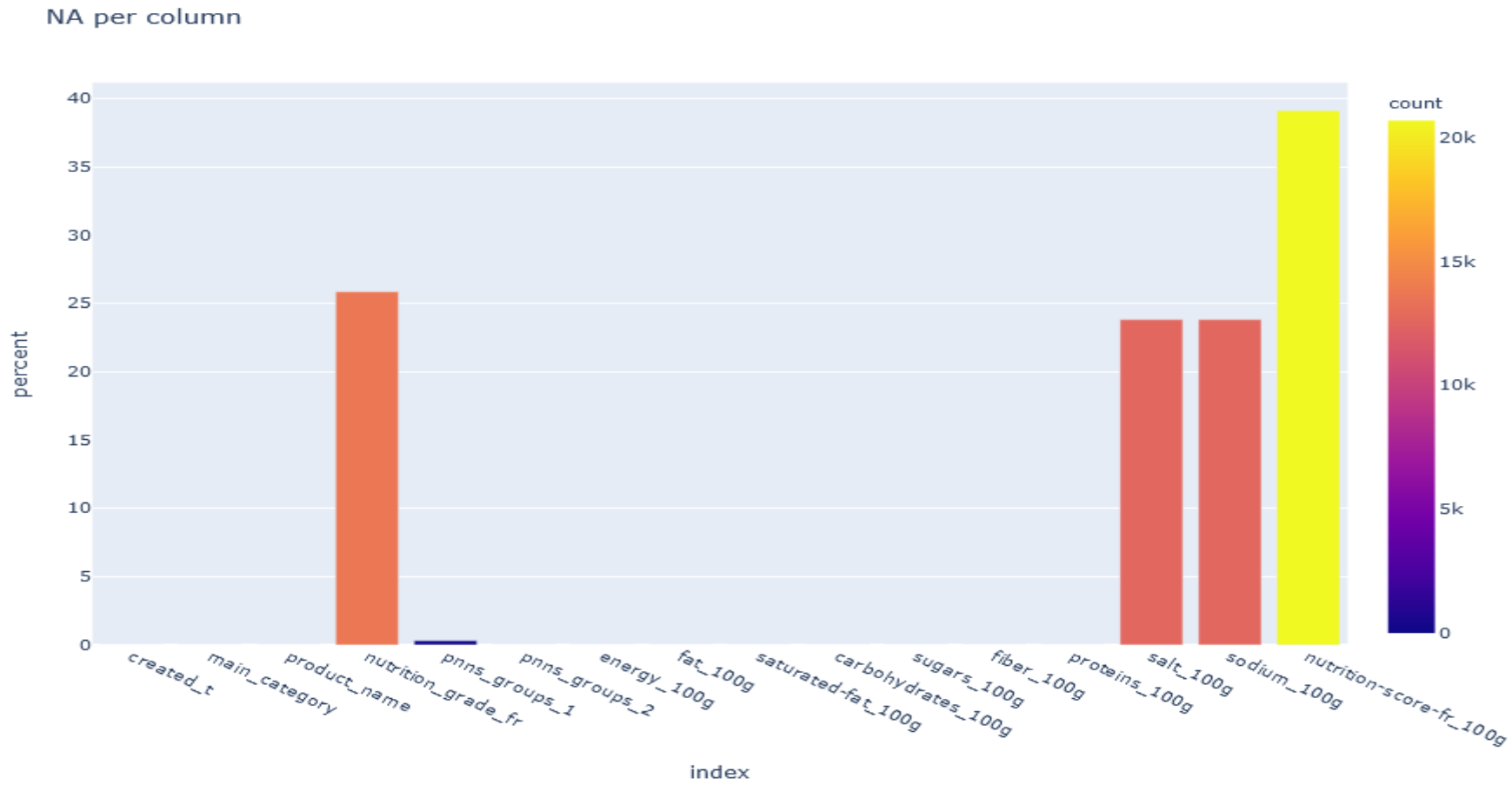
II. NETTOYAGE DES DONNÉES: D/ TRAITEMENT DES NAN:

- Imputation par knn: après



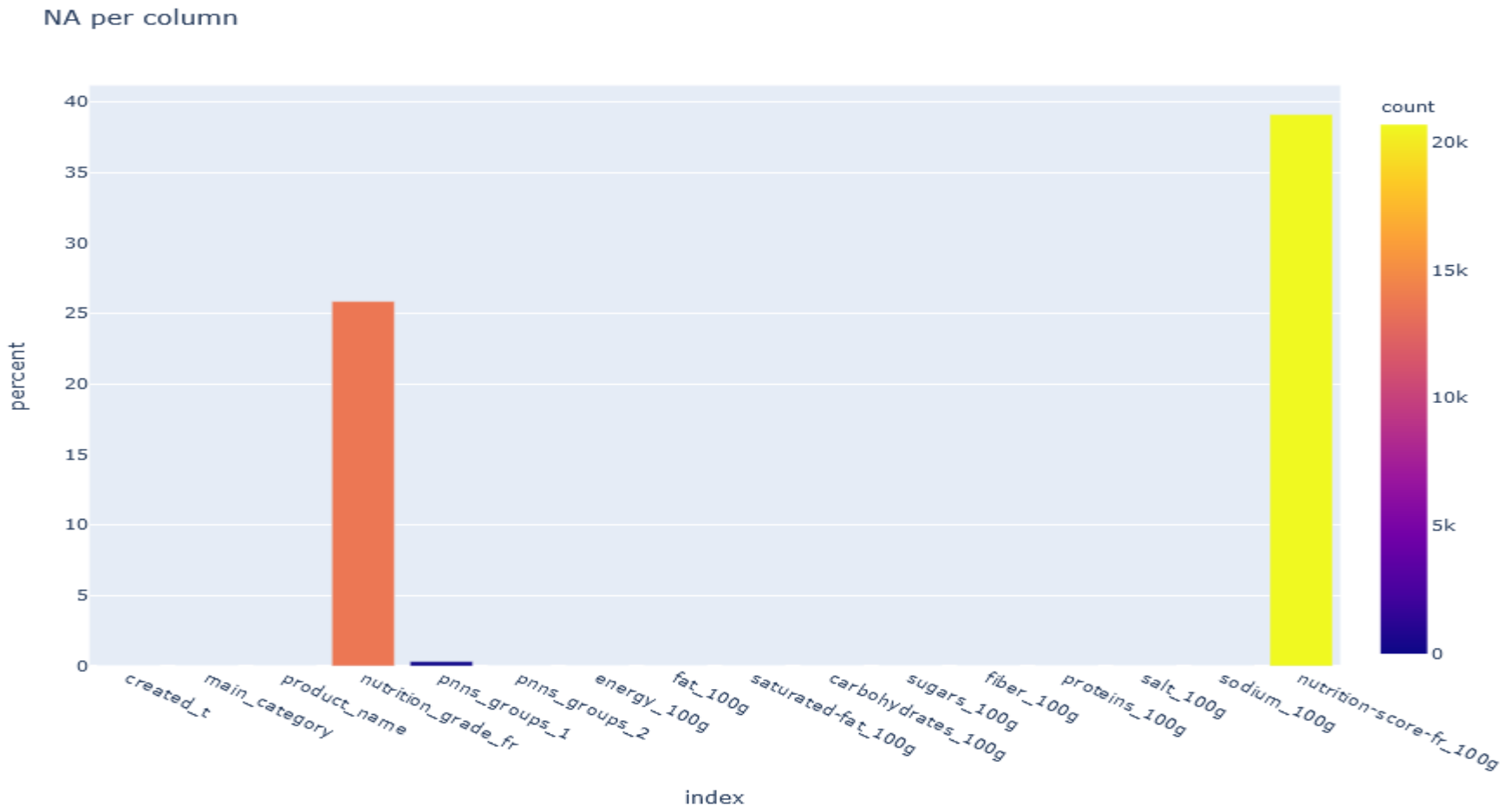
II. NETTOYAGE DES DONNÉES: D/ TRAITEMENT DES NAN:

- Imputation par interactiveImputer: avant

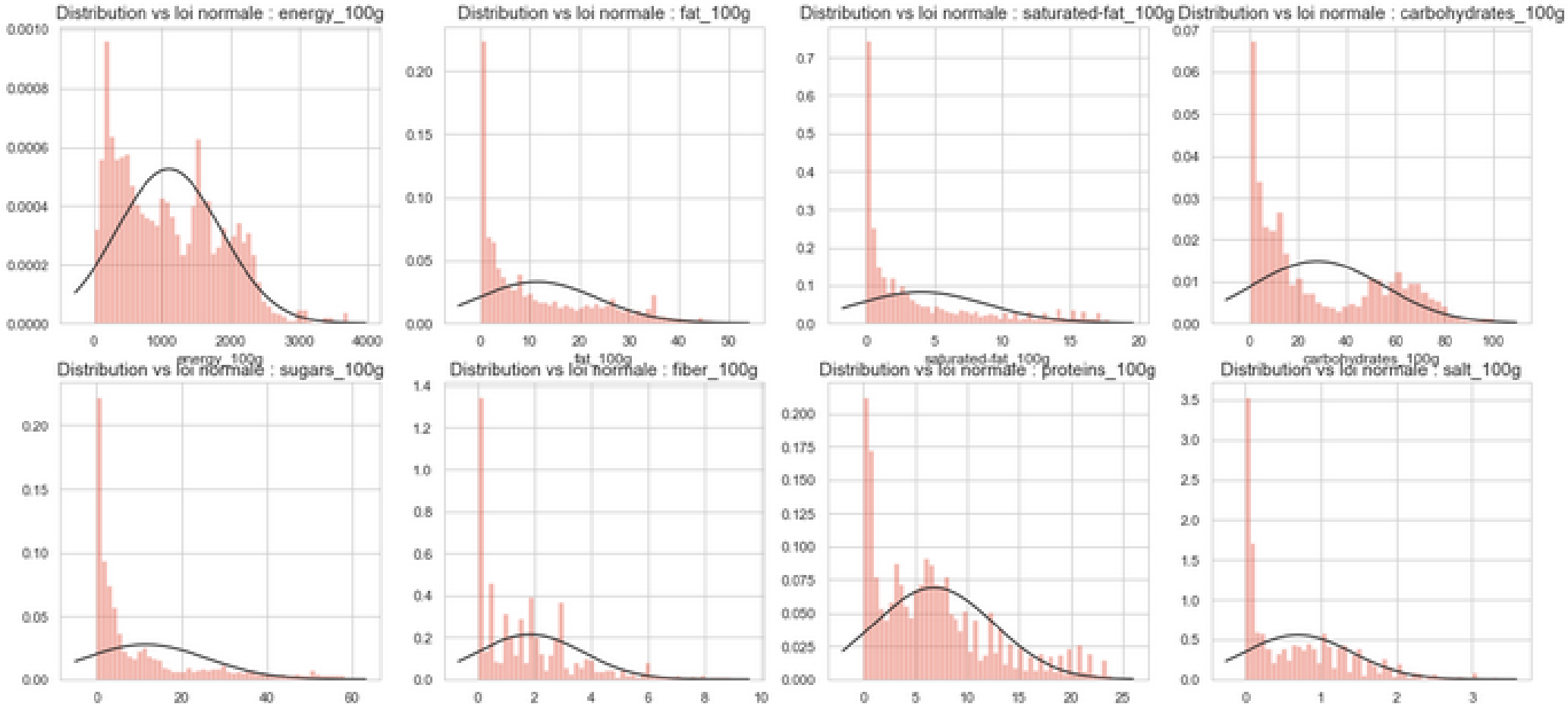


II. NETTOYAGE DES DONNÉES: D/ TRAITEMENT DES NAN:

- Imputation par interactiveImputer: après



II. NETTOYAGE DES DONNÉES: VÉRIFICATION DE LA DISTRIBUTION DES DONNÉES



II. NETTOYAGE DES DONNÉES: QUE RESTE-IL ?

- **Nutrigrade**
- **Nutrition-score:** Une des variables primordiales de ce jeu de données

  - Plusieurs autres variables pour calculer le nutriscore et le nutrigrade pour 100 g:
  - En nutriments et aliments à favoriser (fibres, protéines, fruits et légumes),
  - En nutriments à limiter (énergie, acide gras saturés, sucres, sel).

✓ Création de deux fonctions pour le nutriscore et le nutrigrade

```
# Fonction pour le calcul du du Nutriscore global

def calc_globalscore(row):
    #Energy
    if row["energy_100g"] <= 335:
        a = 0
    elif ((row["energy_100g"] > 335) & (row["energy_100g"] <= 1675)):
        a = 5
```

```
#Fonction de calcl du Nutriscore grade
def calc_nutriscore(row):
    if row["calc_global_score"] < 0 :
        nutriscore = "a"
    elif ((row["calc_global_score"] >= 0) & (row["calc_global_score"] < 5)) :
        nutriscore = "b"
```

II. NETTOYAGE DES DONNÉES: QUE RESTE-IL ?

- **Nutrigrade**
- **Nutrition-score:** Une des variables primordiale de ce jeu de données

- Nous devons à présent vérifier les erreurs d'imputation sur les scores déjà connus, ce qui nous permettra de voir si les calculs sont satisfaisants et utilisables : Utilisation de l'accuracy score

[350]:
```
df_scores = datas_NaN[['nutrition_grade_fr', 'nutrition-score-fr_100g', 'calc_nutriscore', 'calc_global_

from sklearn.metrics import accuracy_score
#
accuracy_nutrigrade = accuracy_score(df_scores['nutrition_grade_fr'].values, df_scores['calc_nutriscore'
print("L'accuracy_score sur les Nutrigrades calculés est de : {:.2f} %.".format(accuracy_nutrigrade*100)
```

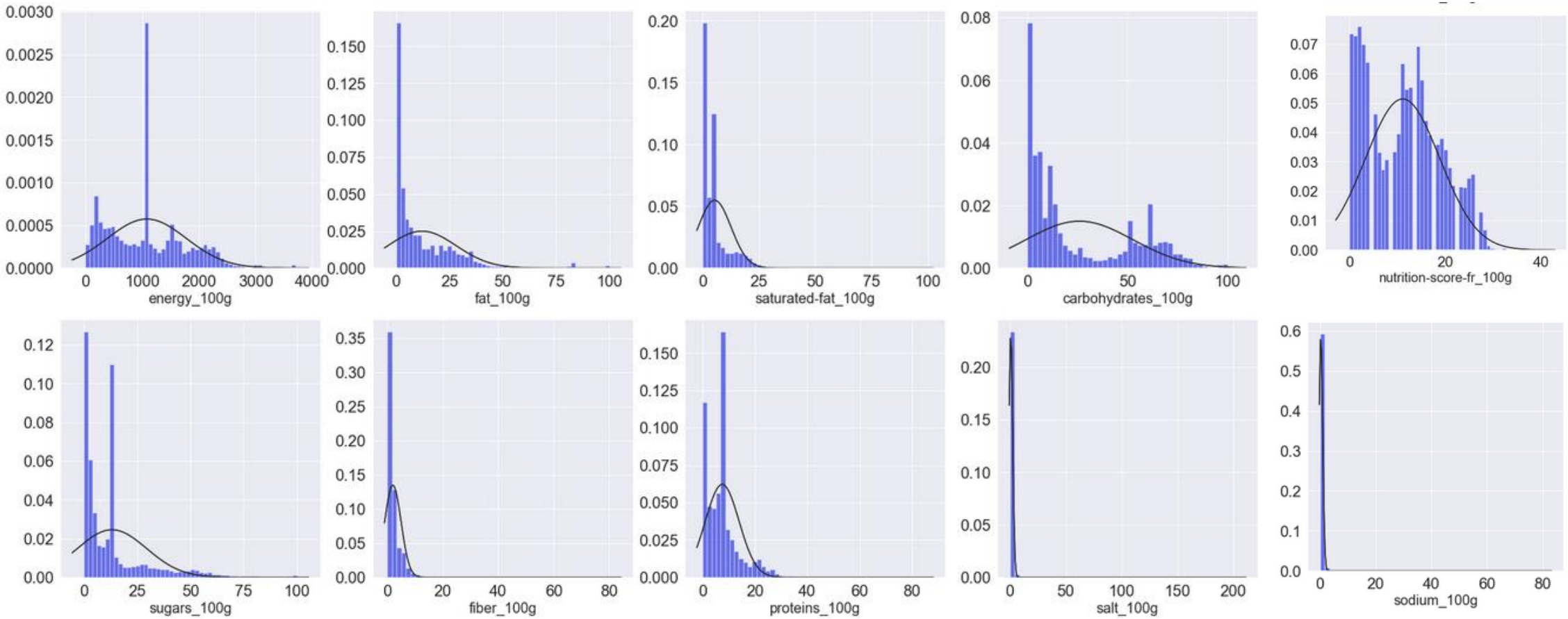
```
L'accuracy_score sur les Nutrigrades calculés est de : 49.70 %.
```

La précision du calcul est donc inférieure à 50%. Ces calculs ne peuvent donc pas être utilisés **pour compléter nos données**.

- Notre application aura donc pour but d'estimer le nutrigrade d'un produit en fonction de ses caractéristiques connues, comme la catégorie, sa teneur en nutriments ...

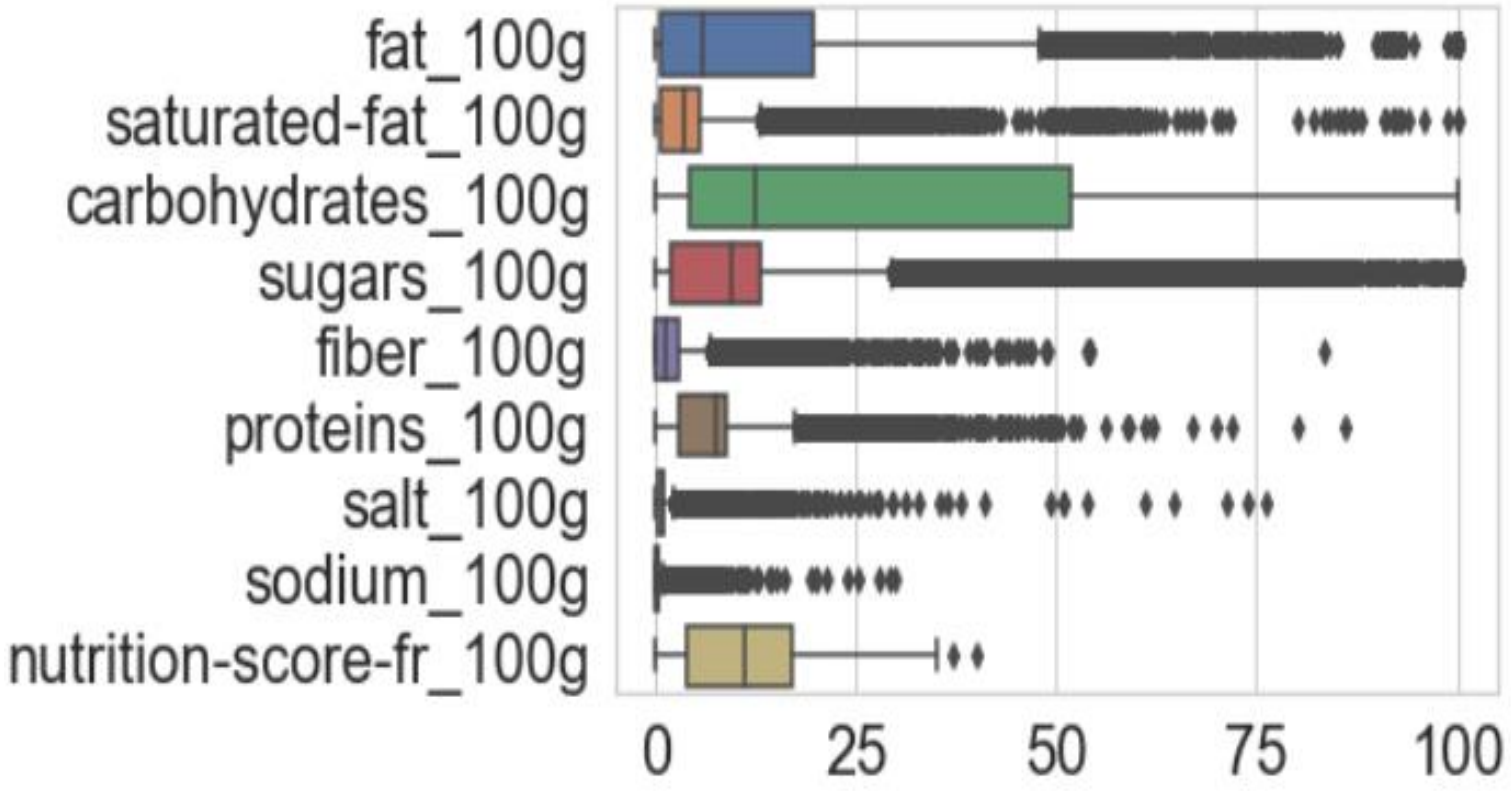
III. ANALYSE EXPLORATOIRE
A/ ANALYSE UNIVARIÉE

- Création d'une fonction permettant de voir en histogramme les différentes variables des données nettoyées

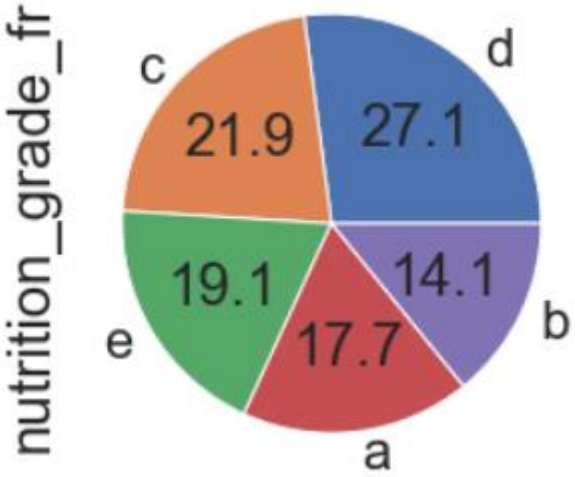


III. ANALYSE EXPLORATOIRE
A/ ANALYSE UNIVARIÉE

- Boxplot de l'ensemble des variables quantitatives

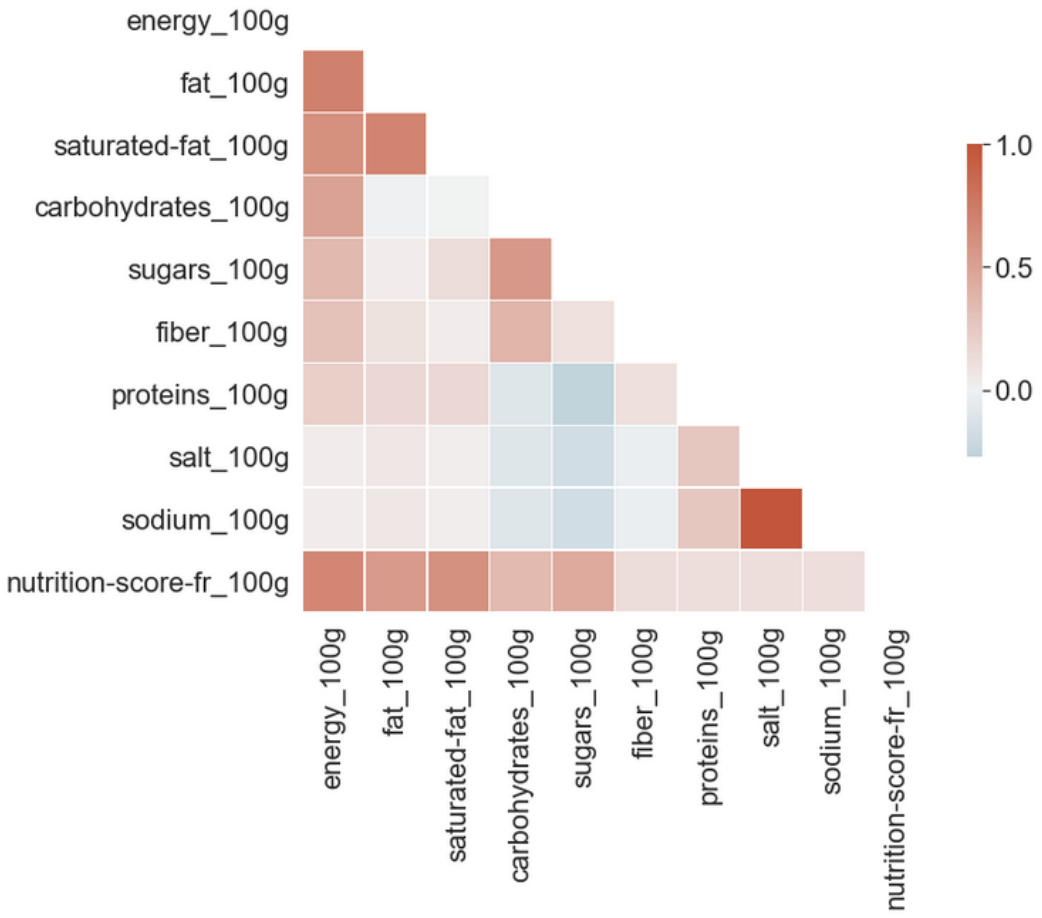


- Pie Plot de la variable qualitative



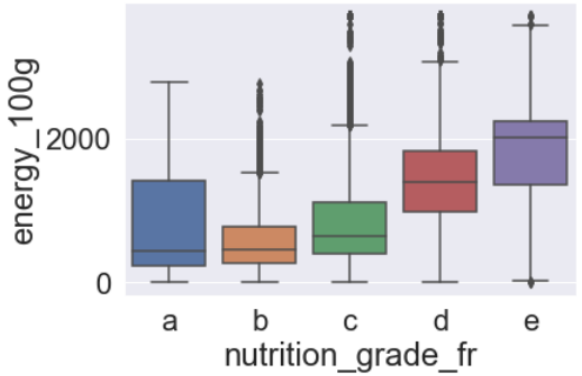
III. ANALYSE EXPLORATOIRE
B/ ANALYSE BIVARIÉE

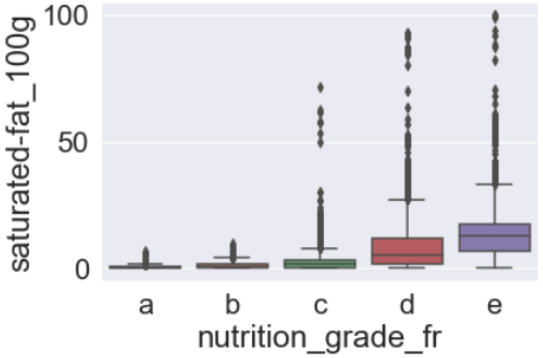
- Corrélation entre l'ensemble des variables quantitatives

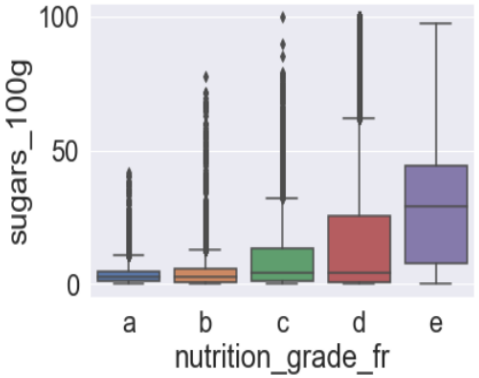


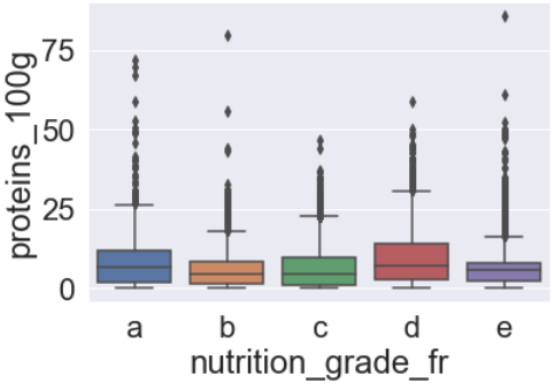
III. ANALYSE EXPLORATOIRE
B/ ANALYSE BIVARIÉE

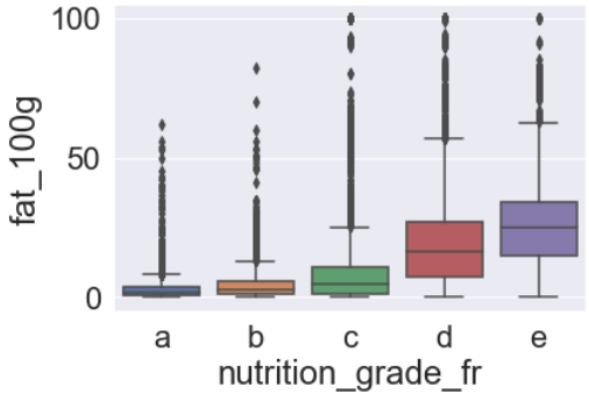
- Variation entre une variable catégorielle et quantitatives

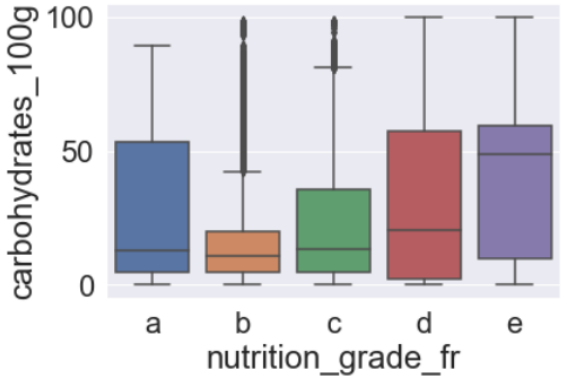


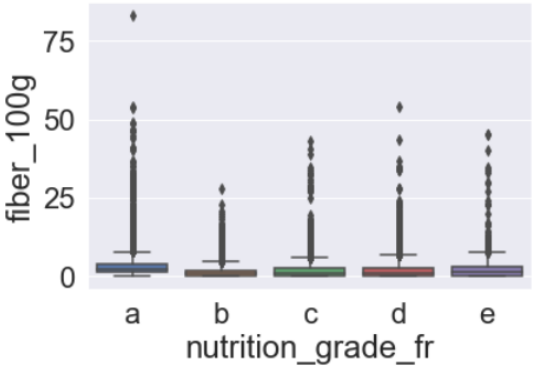


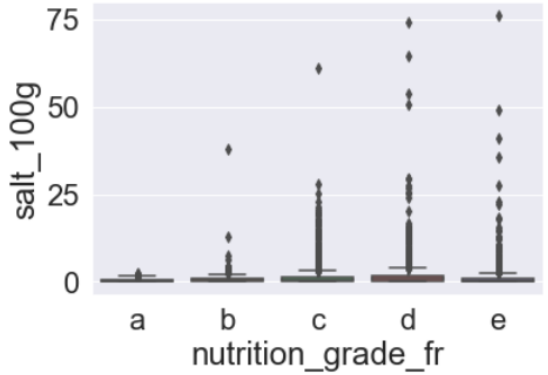












III. ANALYSE EXPLORATOIRE
B/ ANALYSE BIVARIÉE

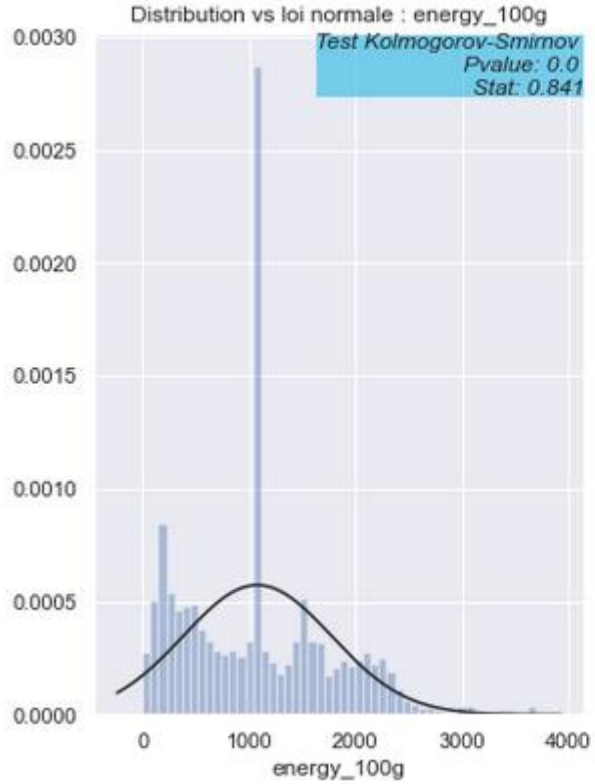
- Variation entre une variable catégorielle et quantitatives

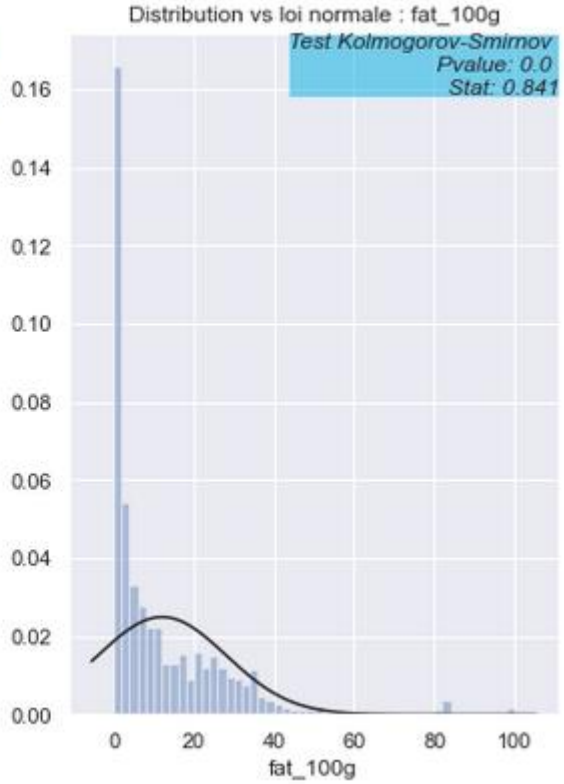
- on voit une différence entre les nutri grades pour toutes les variables.

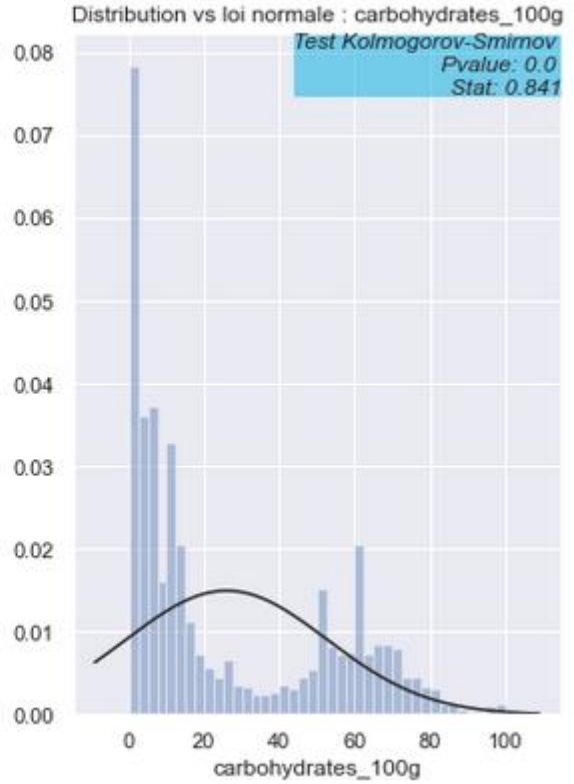
- Toutefois, il faut voir si ces différences sont significatives ou pas

- Pour vérifier si ces différence sont statistiquement significatives on fait l'ANOVA
  - en utilisant une comparaison multivariée sur l'ensemble des échantillons

III. ANALYSE EXPLORATOIRE
C/ ANALYSE MULTIVARIÉE: TEST DE NORMALITÉ







Ho est l'hypohèse de normalité des données

- Si la valeur p > alpha

L'échantillon suit une loi normale
(on ne rejette pas l'hypothèse H0)

- Si la valeur p < alpha

L'échantillon suit une loi normale
(on rejette l'hypothèse H0)

III. ANALYSE EXPLORATOIRE
C/ ANALYSE MULTIVARIÉE: TEST DE KURSKAL

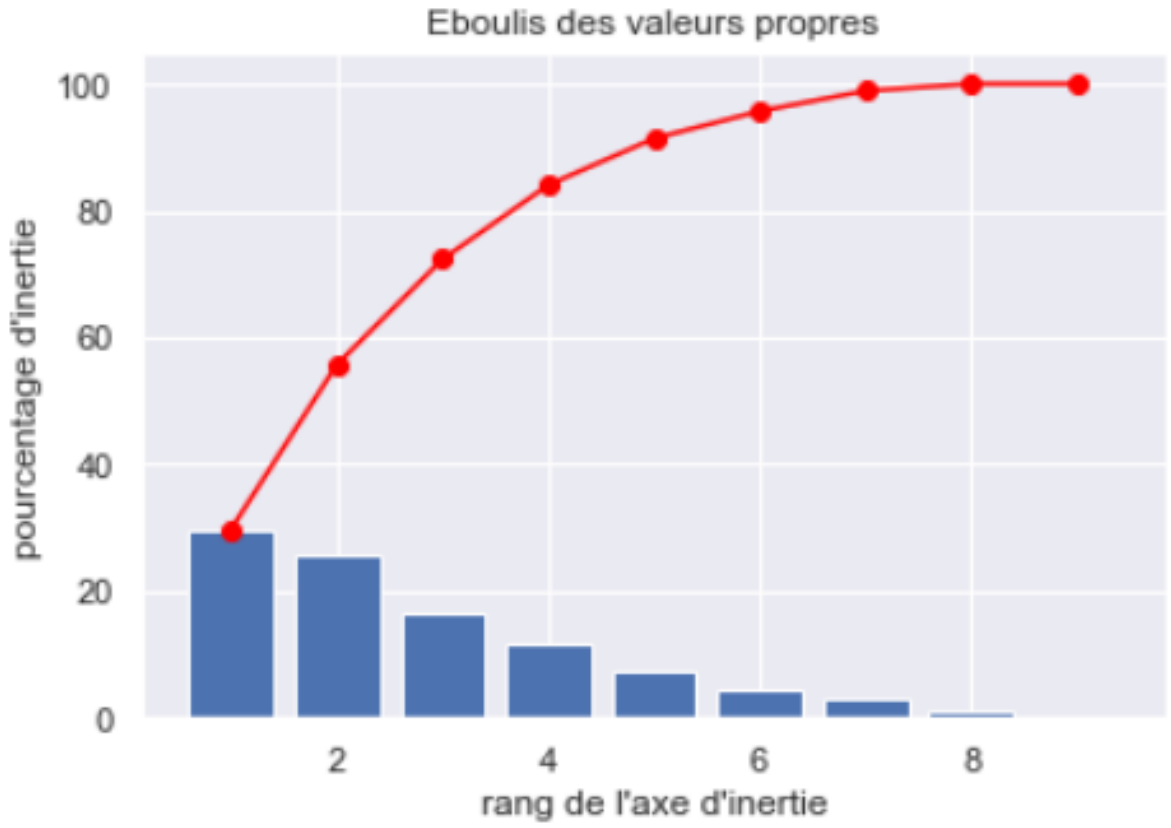
Variable catégorielles et quantitatives

	feature	statistic	pvalue
0	energy_100g	11330.76300080792	0.0
1	fat_100g	10539.816704157129	0.0
2	saturated-fat_100g	12308.884515091378	0.0
3	carbohydrates_100g	1425.8597744519577	1.7066307129807492e-307
4	sugars_100g	5826.521283047669	0.0
5	fiber_100g	2428.940478040603	0.0
6	proteins_100g	776.8166097616947	8.069036394339711e-167
7	salt_100g	2973.6984471559626	0.0
8	sodium_100g	2974.1836134939253	0.0

On conclut donc à une différence significative globale des moyennes des différents nutrigrades de nos features car, elles présentent toutes une valeurs de p< à notre seuil de décision.

III. ANALYSE EXPLORATOIRE
C/ ANALYSE MULTIVARIÉE: ACP

- Variables quantitatives: **éboulis des valeurs propres**

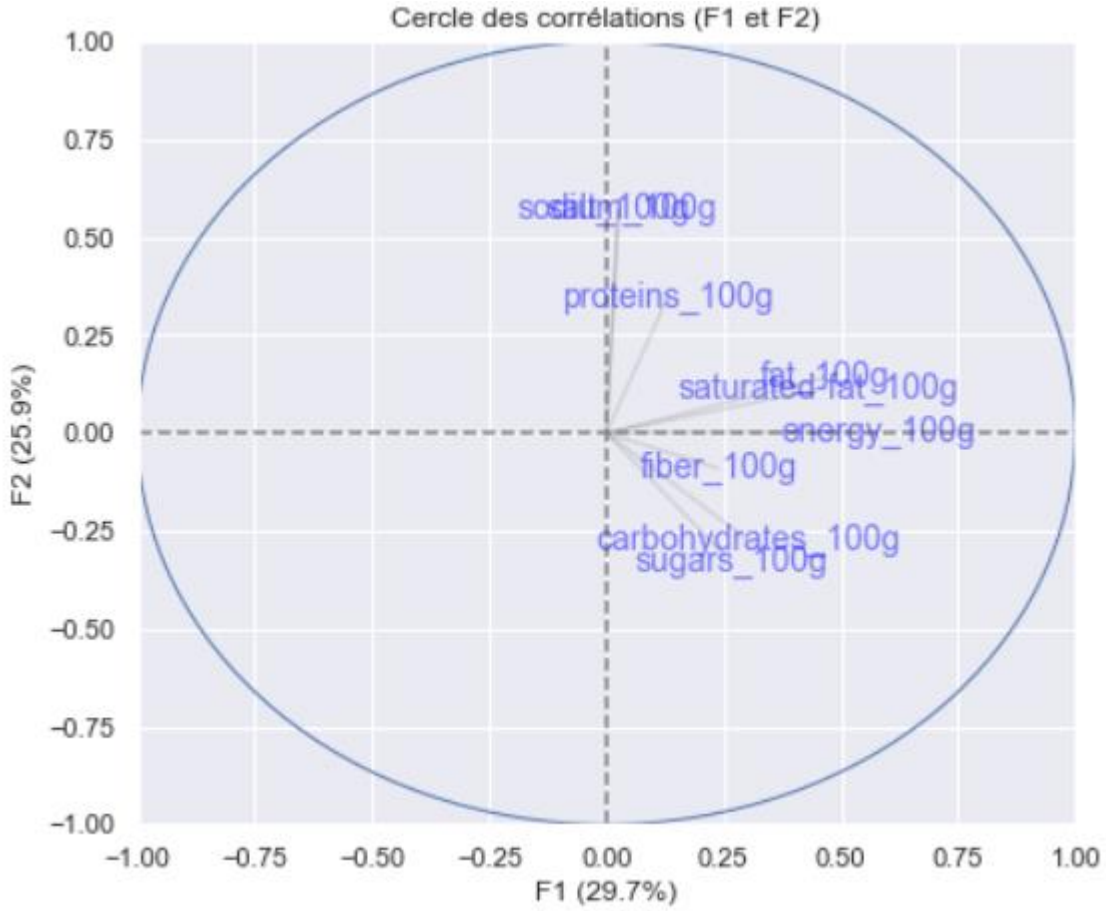


On peut observer que deux composantes permettent d'expliquer 58% de la variance de nos variables.

- Nous allons donc choisir pour nos interprétations 2 composantes principales

III. ANALYSE EXPLORATOIRE
C/ ANALYSE MULTIVARIÉE: ACP

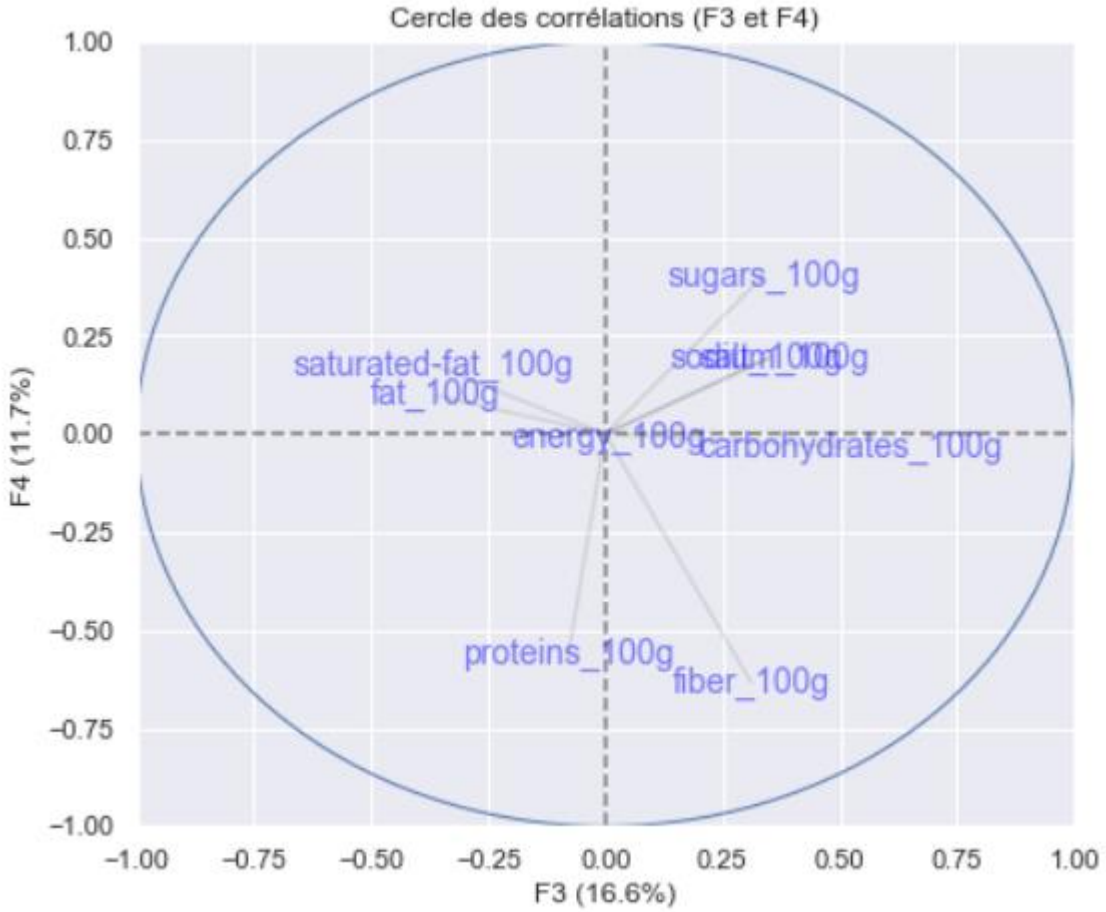
- Variables quantitatives: **Affichage des cercles de corrélation**



- **F1** est très corrélée avec le gras, fibre, les nutriments énergétiques
- **F2** corrélée avec le sodium, les protéines, et le sel et négativement corrélée avec le sucre, carbohydrate

III. ANALYSE EXPLORATOIRE
C/ ANALYSE MULTIVARIÉE: ACP

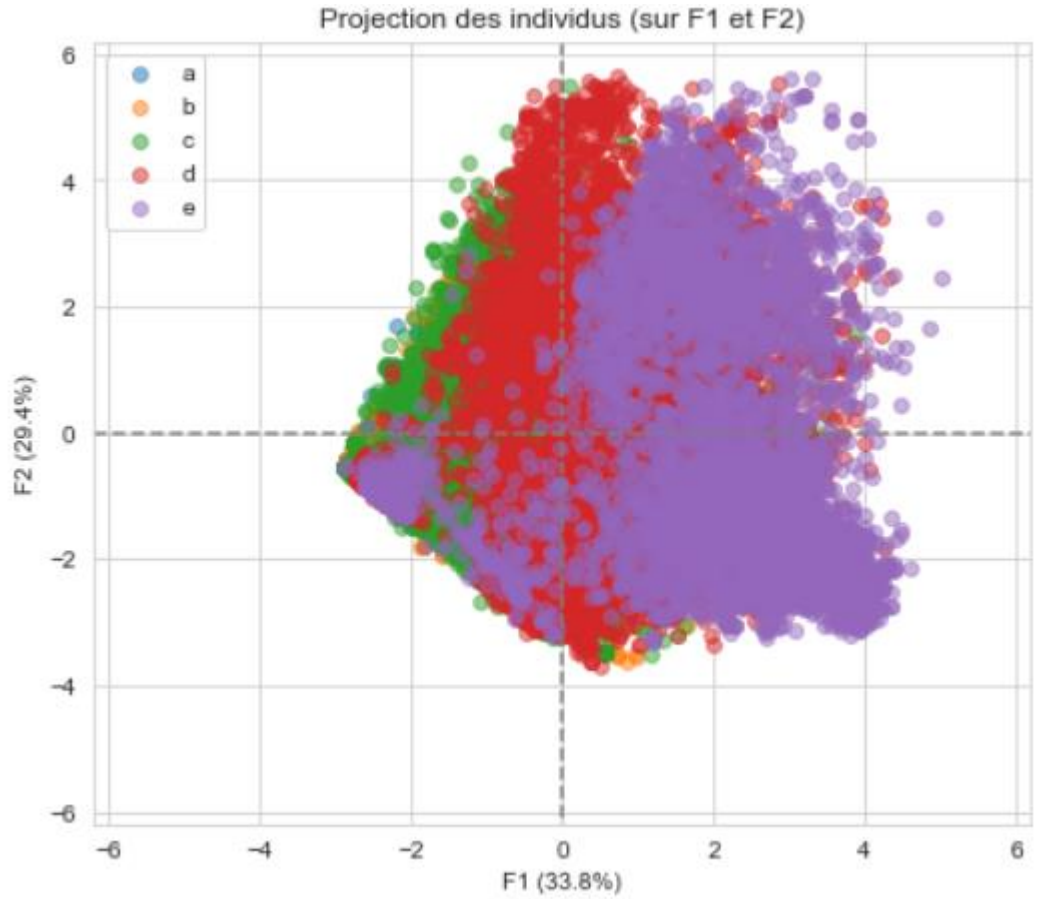
- Variables quantitatives: **Affichage des cercles de corrélation**



- **F3** est très corrélée avec le carbohydrate, le sucre, et négativement corrélée avec le gras
- **F4** corrélée avec le sucre et négativement corrélé avec les protéines et les fibres

III. ANALYSE EXPLORATOIRE
C/ ANALYSE MULTIVARIÉE: ACP

- Variables quantitatives: **Projection des individus sur F1 et F2**



- On peut voir que plus l'aliment est gras, et plus il est énergiques, et plus son score est faible (e), à l'inverse moins il est gras, et plus son score est meilleur (a)
- Les nutriments carbohydrate, sucre, et protéines possèdent quand à eux des score relativement élevé.

CONCLUSION

- Ainsi, pour l'application, on peut se baser
  - d'une part sur les conclusions de l'ACP: On peut recalculer donc le nutrigrade en regardant le taux de matière grasse, de sucre ou de protéines (Sur F1 et F2)
- On peut éventuellement recalculer le nutrigrade en se référant à notre fonction définie précédemment: mais cette fois, nous pouvons combiner certains nutriments en se basant sur le les composantes principales F1 et F2