Projet 3

Mission :

Concevoir une application au service de la santé publique, à partir des données OpenFoodFacts.

Plan

1. Proposition dune idée d ’application
2. Information sur les données
3. Choix d’un pays
4. Nettoyage des données
5. Analyse Exploratoire des données
6. Analyse de la variance: ANOVA
7. Analyse de la composante principale: ACP

1. Idée d’application

Notre application scan un produit alimentaire, vérifie sa compatibilité pour un malade d’ostéoporose puis recommande une liste de produits riches en protéines végétales et en fibres avec moins d'additifs, sel, sucre, et gras. Cette recommandation sert à renforcer la santé des os afin de prévenir l'ostéoporose chez les adultes.

Vérification de la compatibilité:
SI fiber_100g + Proteins_100g + salt_100g + energy_100g + additives_n_100g = nutrition_score_fr == ‘A’. ➔ Compatible

Recommandation:
Si compatible => 3 produits similaires avec grade A.

2. Information sur la base des données

Nb de lignes : **320772**
Nb de colonnes: **162**

56 variables catégorielles
106 variables quantitatives

Plus de la moitié des données sont manquantes.

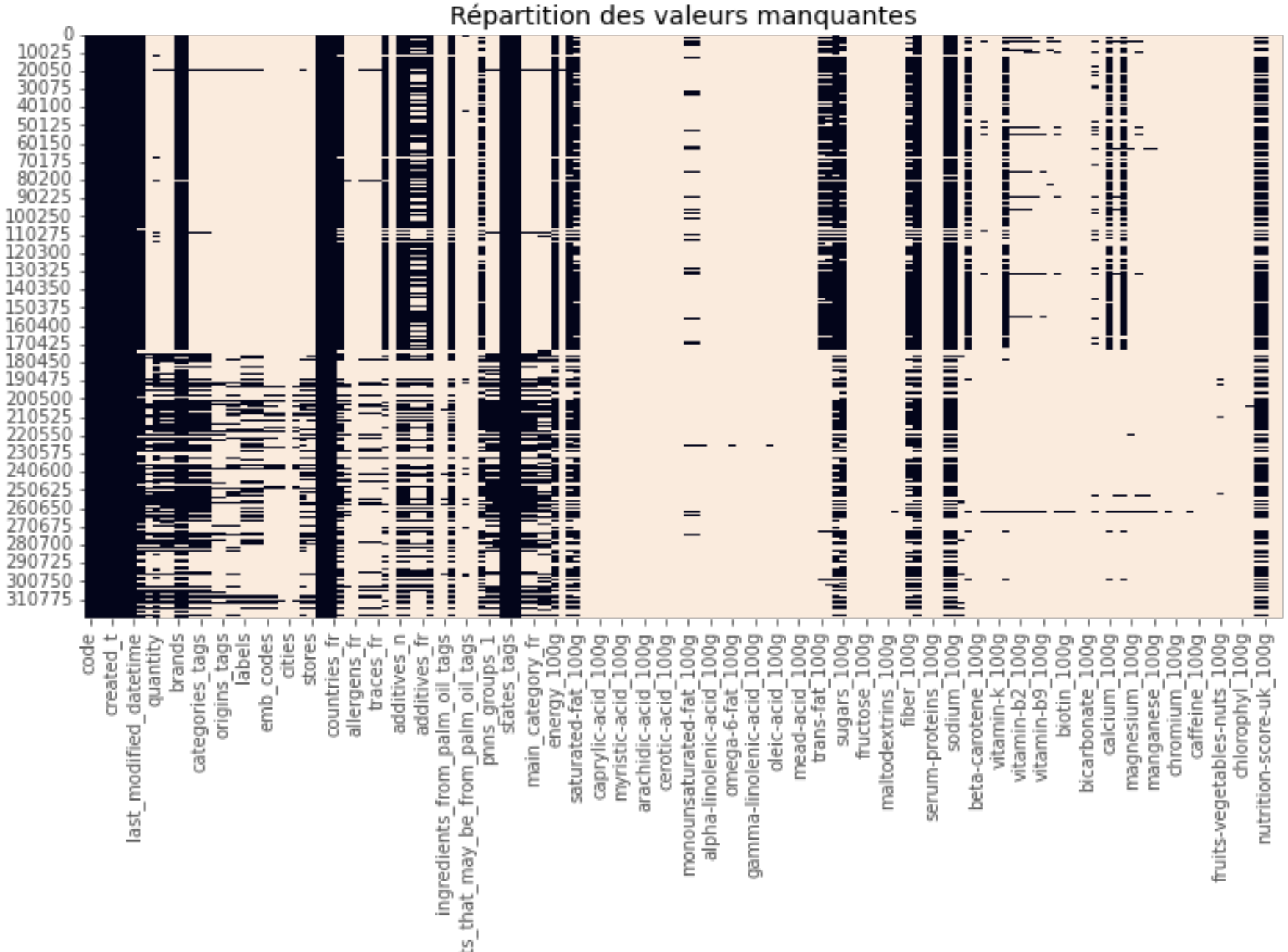
Information sur :

- Nom de produit
- Marques
- Pays d'origine
- Information nutritionnelles

Plus de 76% de données manquantes en moyenne

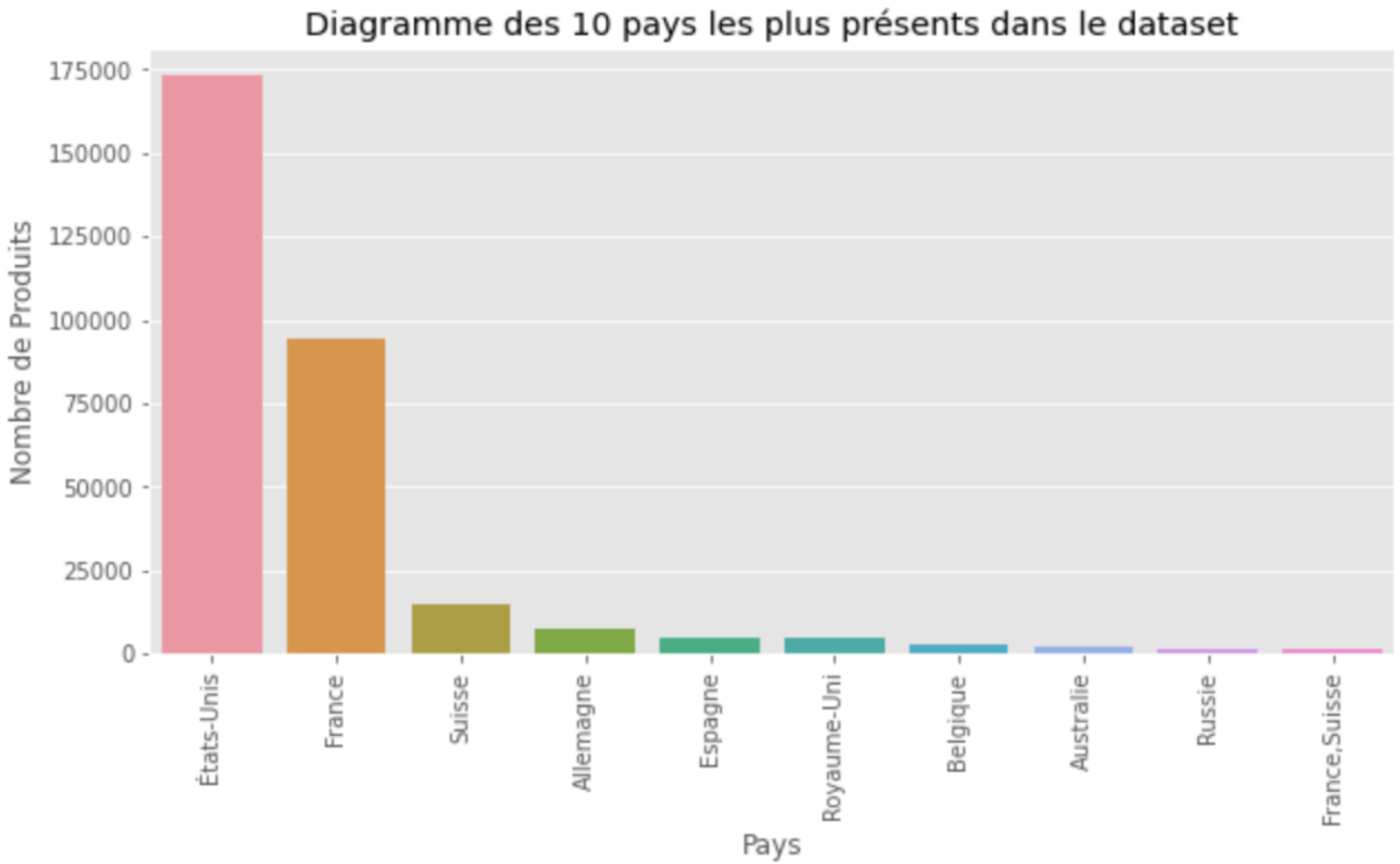
```
1  # percentage de données manquantes
2  NaN_col = data.isnull().mean() * 100
3  NaN_col.describe()
```

```
count    162.000000
mean      76.221573
```



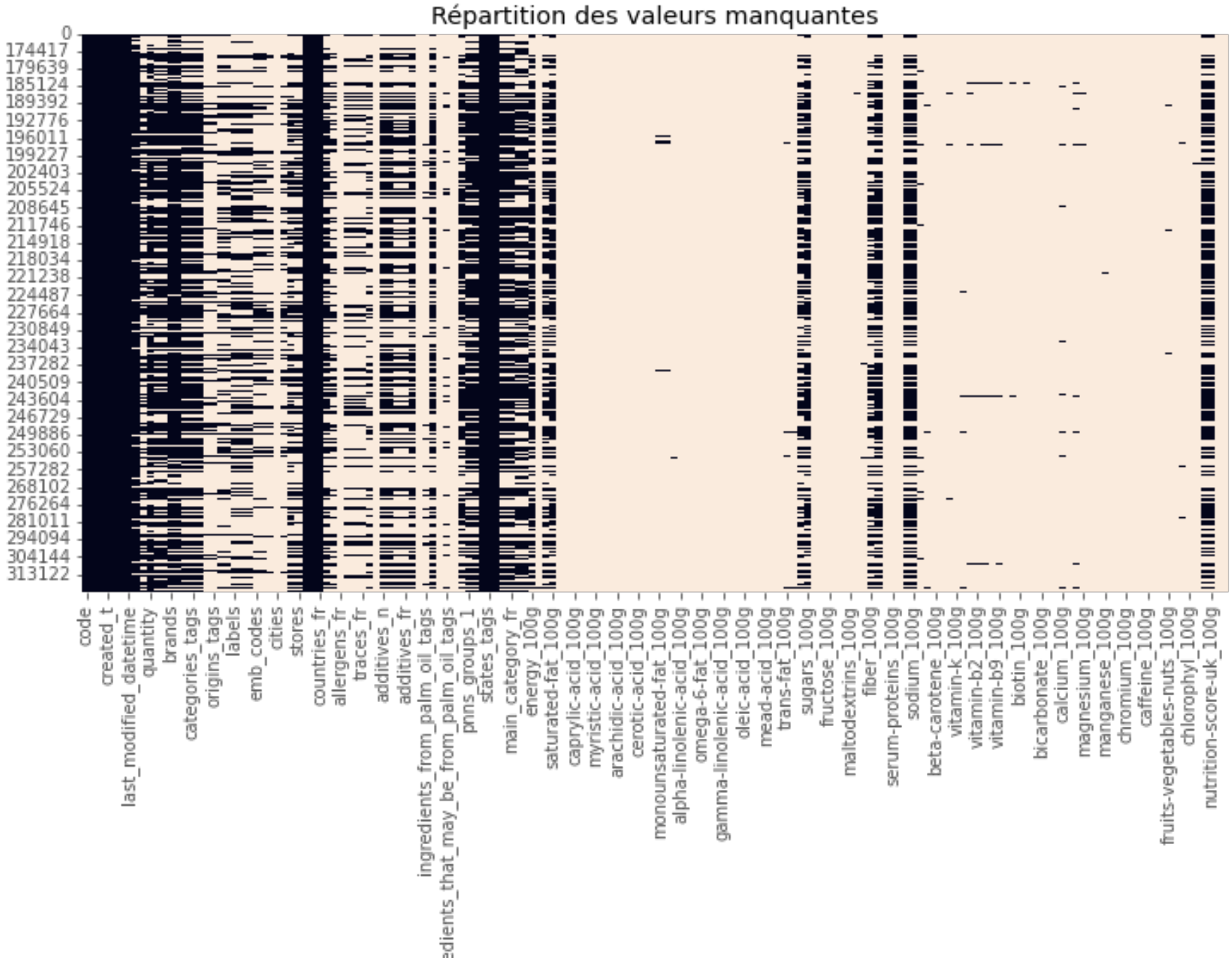
3. Choix d'un pays

- **France**



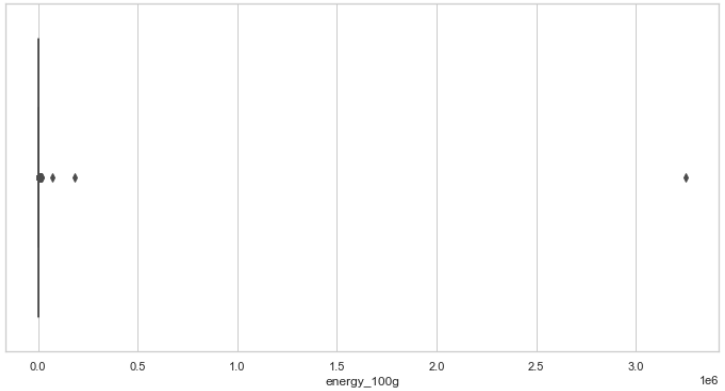
Nb de lignes : **98439**
Nb de colonnes: **162**

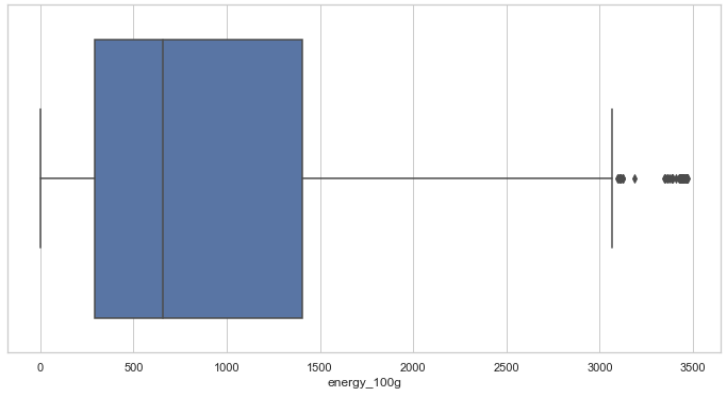
56 variables catégorielles
106 variables quantitatives



4. Nettoyage des données

1. Suppression des colonnes inutiles
2. Suppression des doublons dans la variables « code »
3. Suppression des variables qui ont plus de 75% de valeurs manquantes
4. Choix des variables pertinentes
5. Détection et suppression des outliers (Méthode IQR)
6. Suppression des noms insignifiants pour des produits





4. Nettoyage des données (suite 1)

Détection et suppression des valeurs aberrantes

	additives_n	ingredients_from_palm_oil_n	energy_100g	fat_100g	saturated-fat_100g	carbohydrates_100g	sugars_100g	fiber_100g	proteins_100g
count	38290.000000	38290.0	43737.000000	32239.000000	41889.000000	31907.000000	41981.000000	31451.000000	43531.000000
mean	1.412431	0.0	859.283497	8.567031	3.150888	23.357236	7.936407	1.769366	6.781799
std	1.841232	0.0	662.467776	10.102643	4.357026	25.009043	9.938815	1.912738	6.070050
min	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.0	294.000000	0.800000	0.200000	4.400000	1.000000	0.000000	1.600000
50%	1.000000	0.0	659.000000	4.000000	1.200000	12.000000	3.500000	1.300000	5.600000
75%	2.000000	0.0	1406.000000	13.600000	4.000000	40.000000	11.000000	2.800000	10.000000
max	7.000000	0.0	3473.000000	50.500000	18.000000	100.000000	43.000000	8.000000	24.700000

4. Nettoyage des données (suite 2)

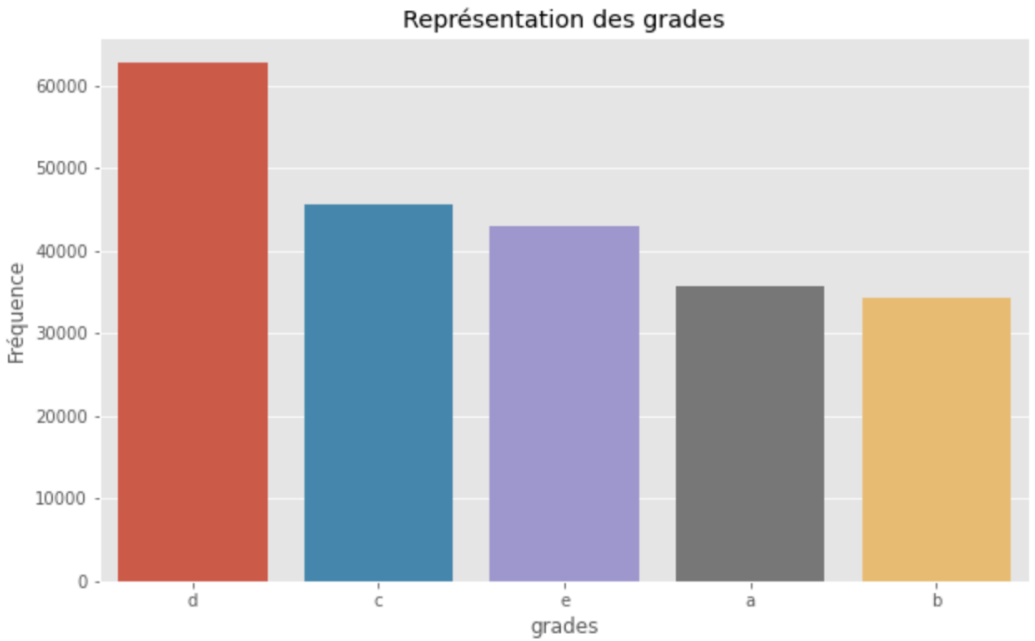
Gestion des valeurs manquantes

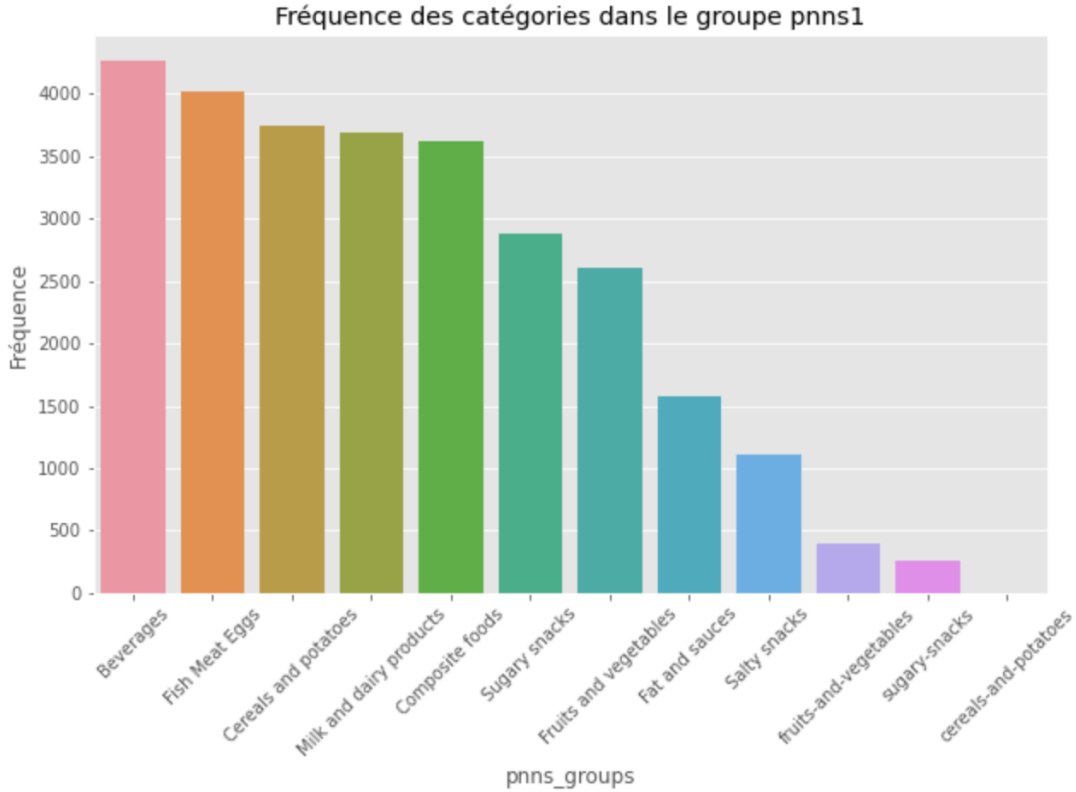
1. Remplacement des valeurs numériques manquantes par la médiane en fonction des groupes « pnns1 »
2. Suppression des valeurs manquantes pour « product_name », « image_url », et « brands »

```
def imputationForNutriScoreAndGrade(data, quanti, nutri_grade):
    # Avant d'imputer les valeurs manquantes de nutrition-grade (qui sont des valeurs qualitatives),
    #on encode les valeurs pour qu'elles soient numériques
    le = preprocessing.LabelEncoder()
    le.fit(data['nutrition_grade_fr'])
    data['nutrition_grade_fr'] = np.where(le.transform(data['nutrition_grade_fr']) == 5,
                                          np.nan, le.transform(data['nutrition_grade_fr']))

    data_quanti = data[quanti + nutri_grade].copy()

    # Ici on impute les valeurs manquantes pour nutrition-grade en utilisant le knn
    imputer = KNNImputer(n_neighbors=5)
    data_quanti[quanti + nutri_grade] = imputer.fit_transform(data_quanti[quanti + nutri_grade])

    data_quanti['nutrition_grade_fr'] = data_quanti['nutrition_grade_fr'].apply(math.floor)

    # On décode les valeurs de nutrition-grade pour retrouver les valeurs sous formes qualitatives (de A à E)
    data_quanti['nutrition_grade_fr'] = le.inverse_transform(data_quanti['nutrition_grade_fr'])

    data[quanti + nutri_grade] = data_quanti.copy()

    nutrition_grade_map = {}
    # On ajoute dans un dictionnaire chaque valeur du nutri-grade avec la moyenne des nutri-score associés
    for value in data['nutrition_grade_fr'].unique():
        nutrition_grade_map[value] = data['nutrition-score-fr_100g'][data['nutrition_grade_fr'] == value].mean()

    # On impute les valeurs manquantes de nutrition-score avec la moyenne coresspondant au bon nutrition-grade
    for index in data.index:
        if str(data['nutrition-score-fr_100g'][index]) == 'nan':
            data['nutrition-score-fr_100g'][index] = nutrition_grade_map[data['nutrition_grade_fr'][index]]

    return data.copy()
```

5. Analyse Exploratoire

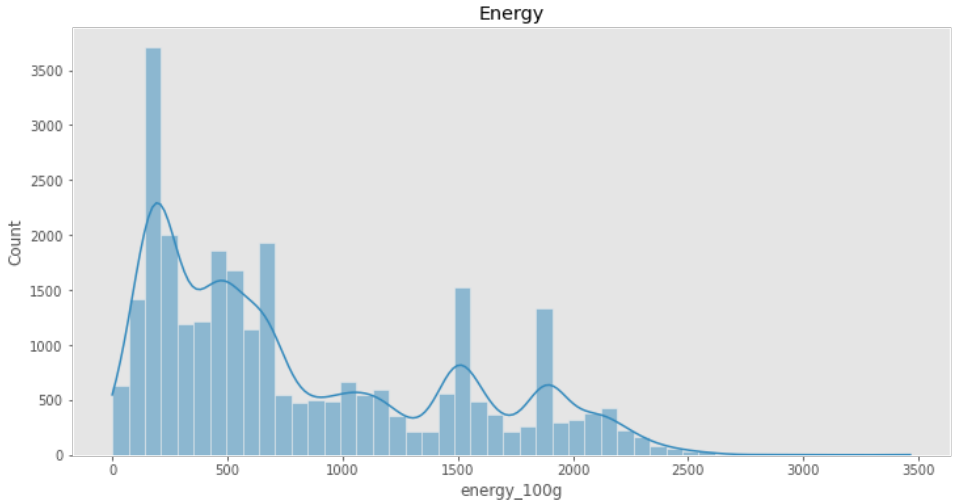
5.1 – Analyse Univariée

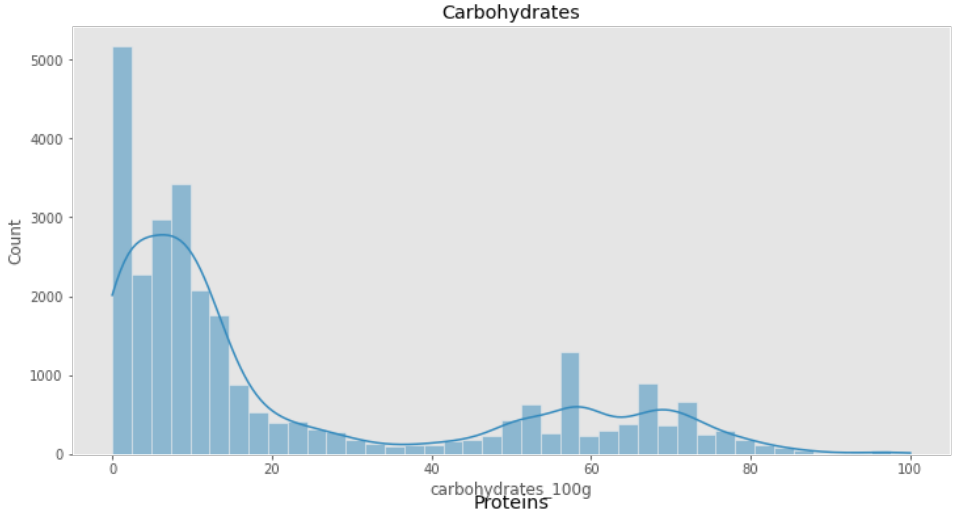


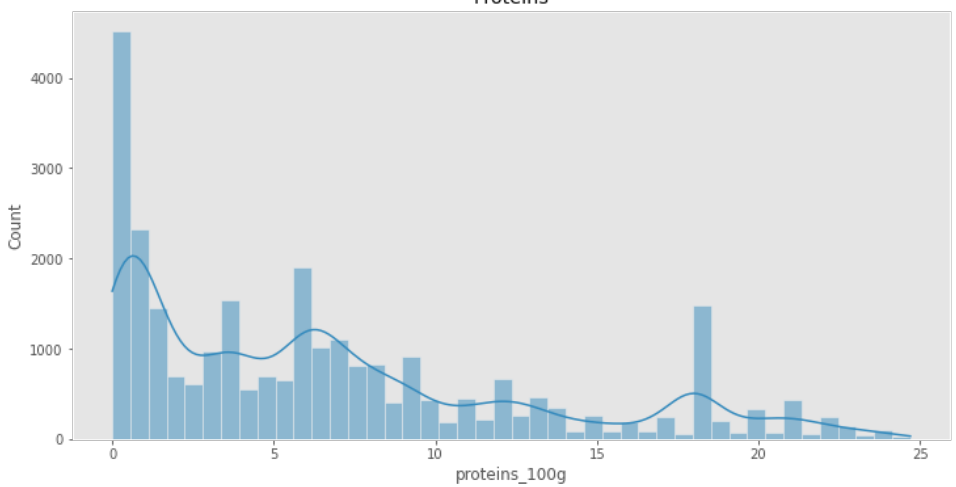


5. Analyse Exploratoire

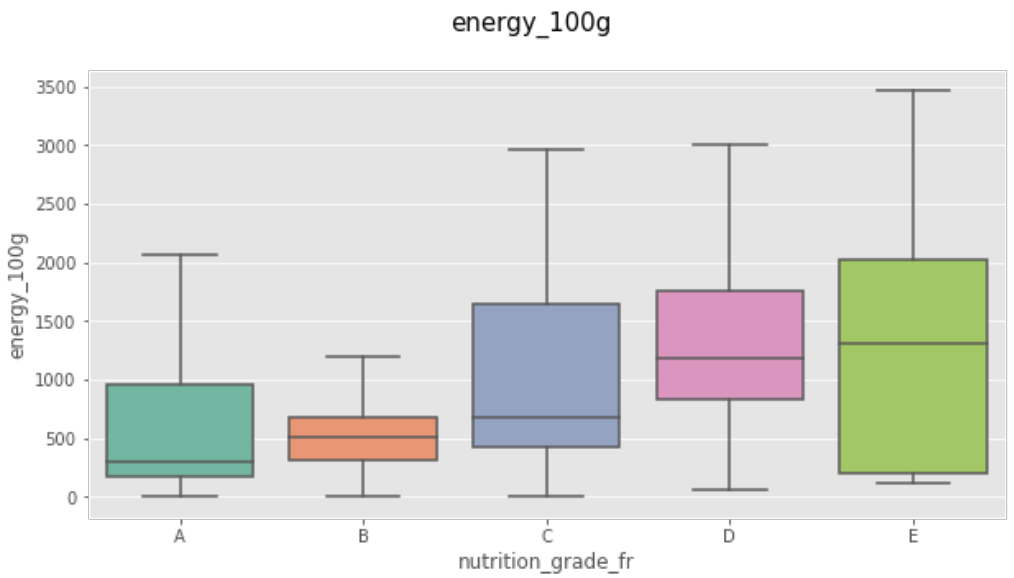
5.1 – Analyse Univariée (suite 1)

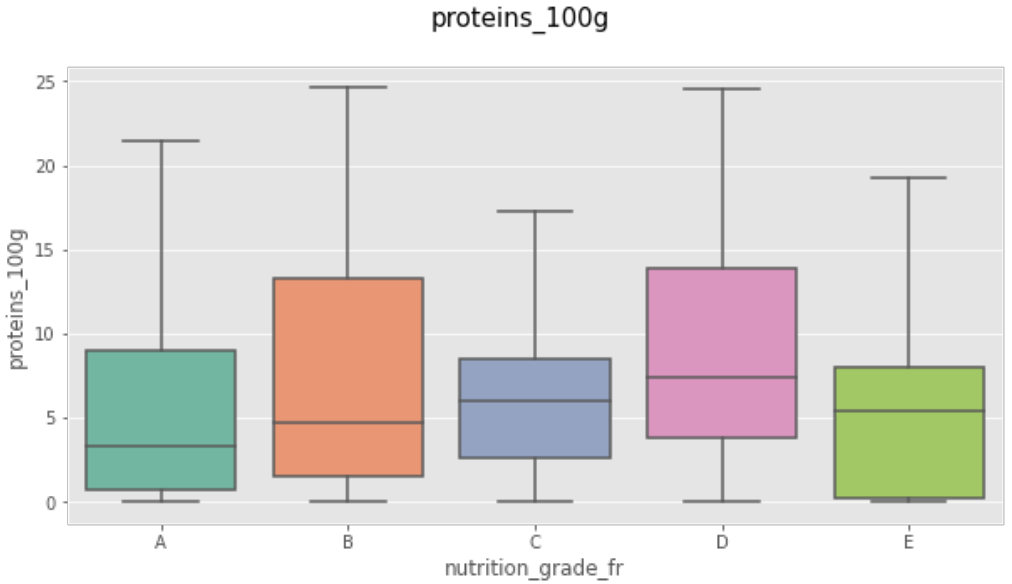


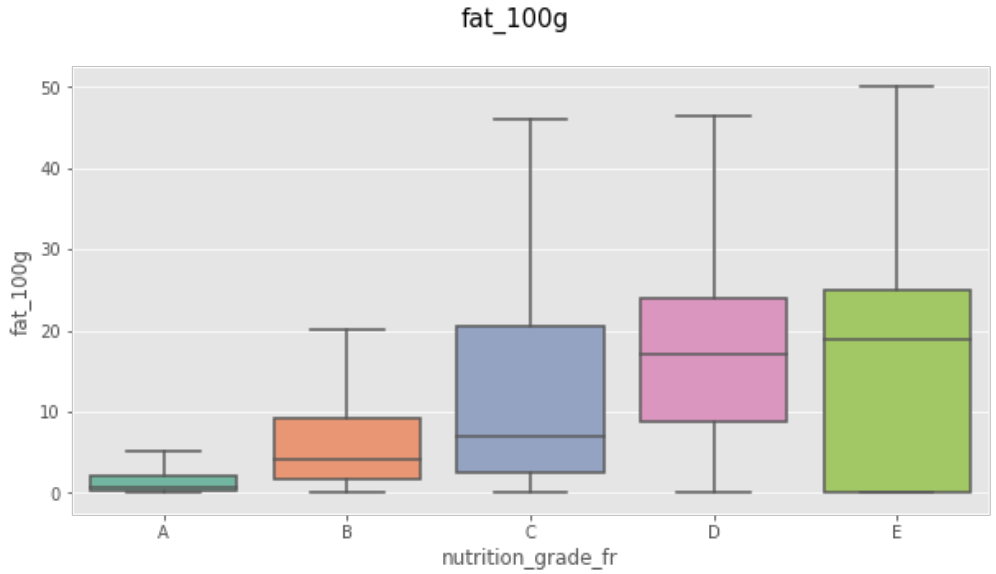


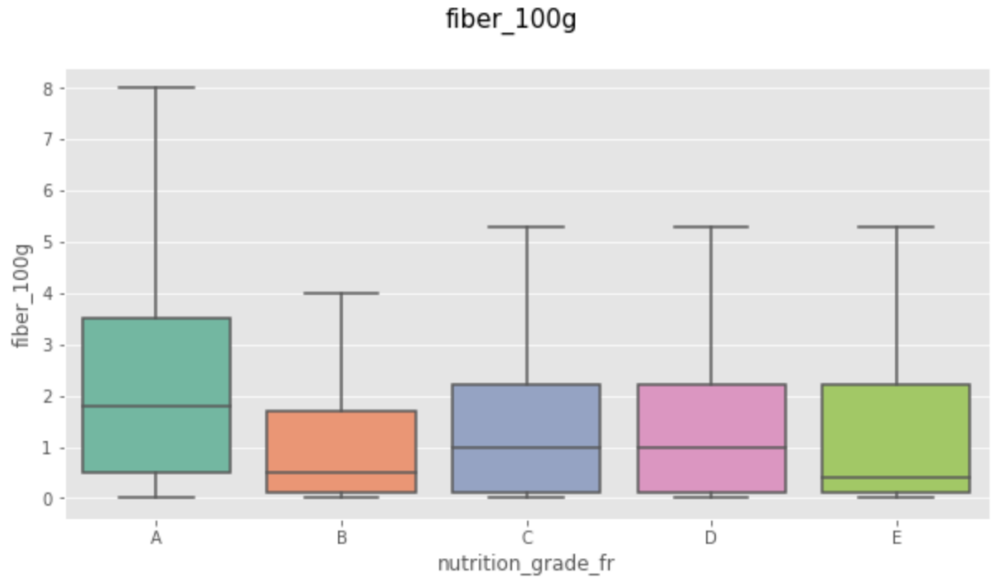


5.2 Analyse Bivariée

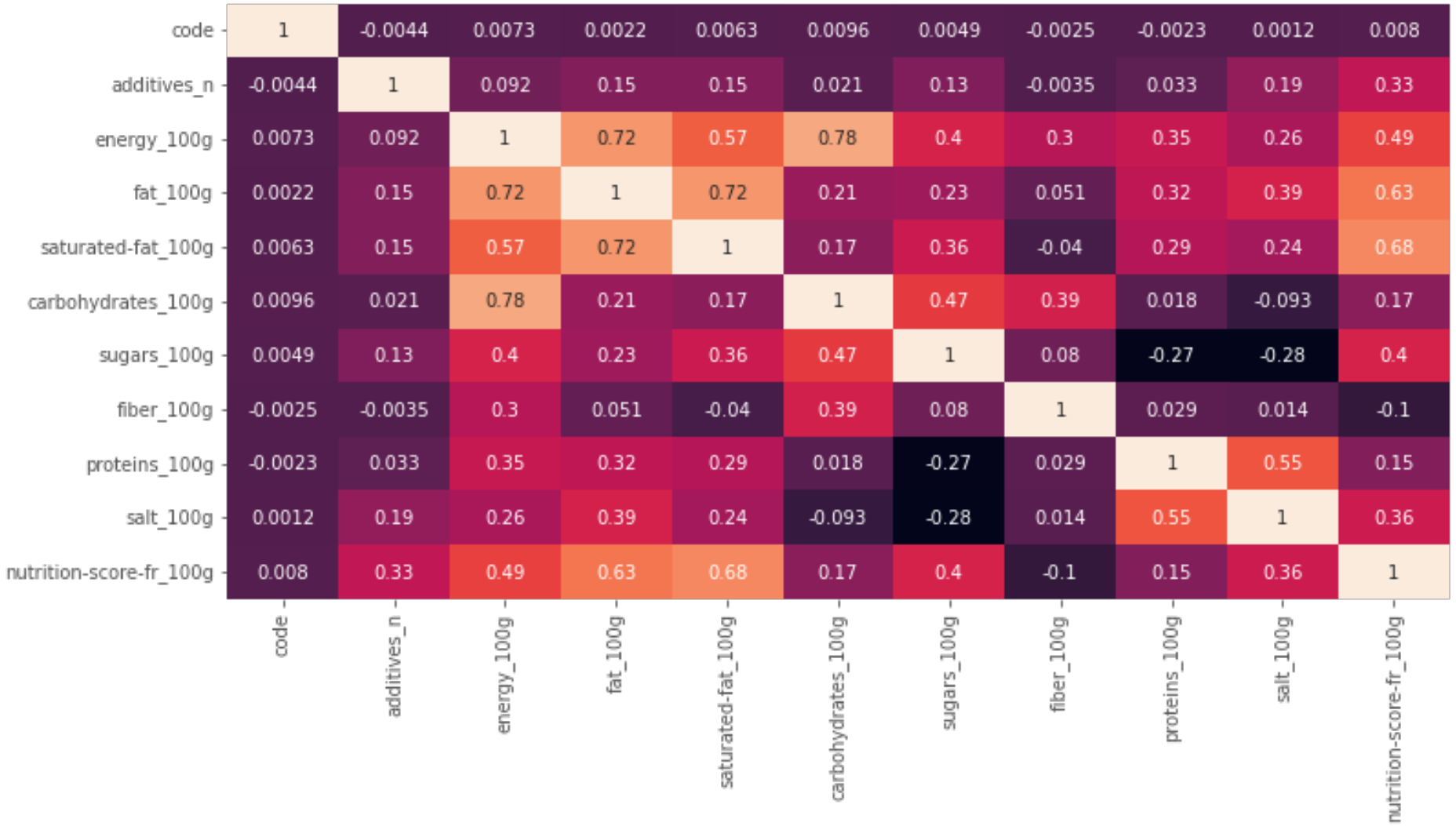








5.3 Analyse Multivariée



5. Analyse de la variance : ANOVA

ANOVA est une analyse de la variance qui permet de conclure s'il y a une différence ou non entre les variances à travers des groupes. Dans notre cas il s'agit de voir si la variance des nutriments est la même pour tous les grades nutritionnels.

Nous avons donc les deux hypothèses suivantes:

Hypothèse nulle : La variance des nutriments est la même pour touts les grades nutritionnels
Hypothèse alternative: La variance des nutriments est différente à travers les grades nutritionnels

Protéines

	sum_sq	df	F	PR(>F)
nutrition_grade_fr	4.809346e+04	4.0	322.928	3.486512e-272

Energie

	sum_sq	df	F	PR(>F)
nutrition_grade_fr	2.063273e+09	4.0	1499.613206	0.0

Fat

	sum_sq	df	F	PR(>F)
nutrition_grade_fr	8.364569e+05	4.0	3301.596223	0.0

Carbohydrates

	sum_sq	df	F	PR(>F)
nutrition_grade_fr	9.419815e+05	4.0	400.27314	0.0

Sugars

	sum_sq	df	F	PR(>F)
nutrition_grade_fr	4.445740e+05	4.0	1517.554591	0.0

Fibres

	sum_sq	df	F	PR(>F)
nutrition_grade_fr	3845.001006	4.0	376.488743	1.926913e-316

Salt

	sum_sq	df	F	PR(>F)
nutrition_grade_fr	2149.537871	4.0	1455.488889	0.0

Nutrition score

	sum_sq	df	F	PR(>F)
nutrition_grade_fr	1.428242e+06	4.0	72457.689084	0.0

Nombre Additives

	sum_sq	df	F	PR(>F)
nutrition_grade_fr	9938.901092	4.0	818.581393	0.0

p-values < 0.005, dans tous les cas de figures

- Rejet hypothèse nulle.
- Différence statistiquement significative entre la variance des nutriments à travers les grades nutritionnels.

6. Analyse Multivariée: ACP

- L'analyse en composantes principales (ACP) vise à réduire la dimensionnalité de l'ensemble des données
- Redimensionner les données de sorte à capturer la plus grande variance des données.

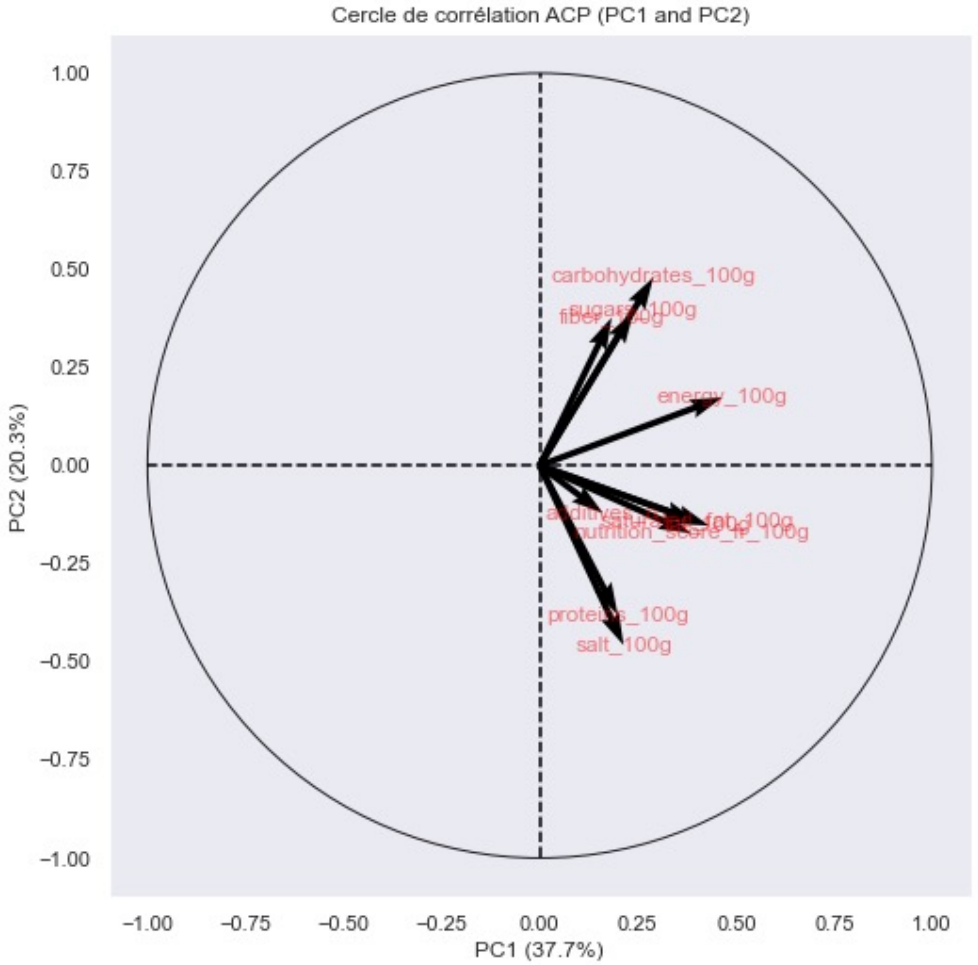
1. Sélection des colonnes pour l'ACP (nombre de composantes - features à redimensionner)
2. Normalisation - Mise à l'échelle
3. Calcul des composantes principales (avec PCA de sklearn)
4. Transformer la matrice obtenue en DataFrame
5. Graphique pour afficher les cercles d’association entre les variables et les composantes principales

6. Analyse Multivariée: ACP

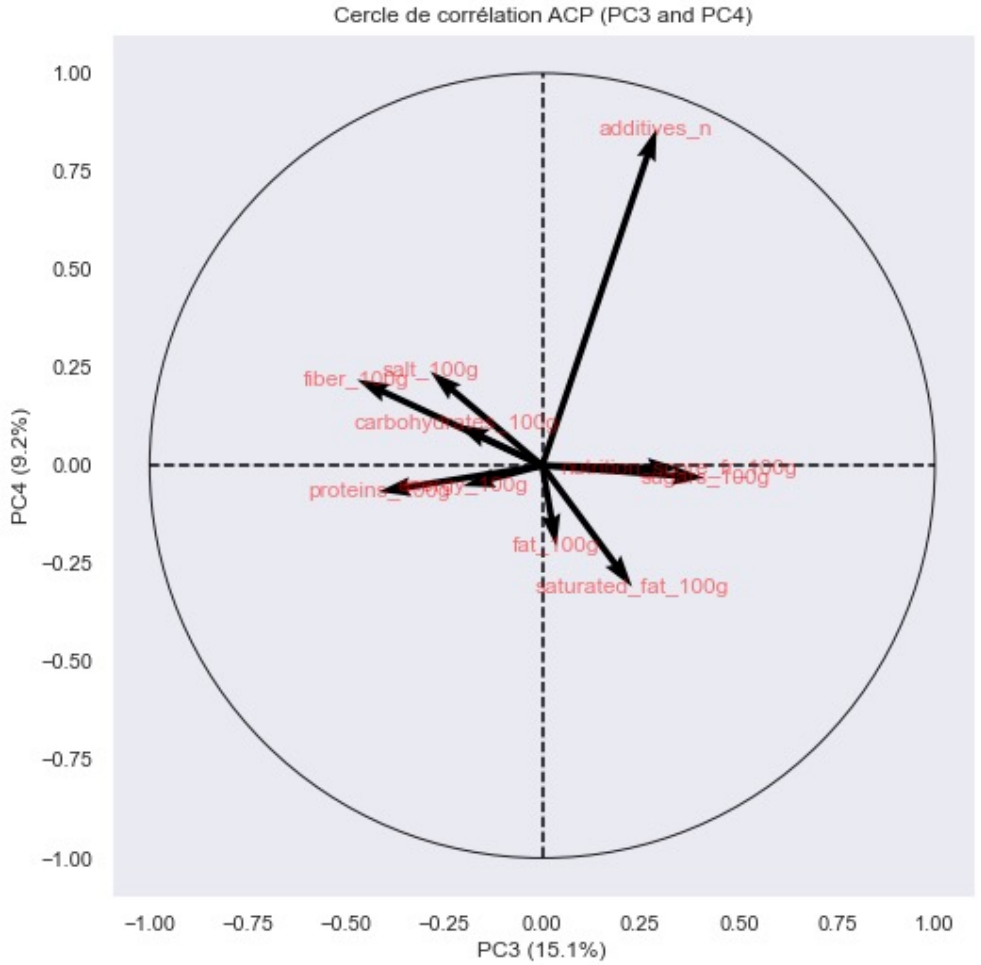
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
additives_n	0.145048	-0.087740	-0.280187	0.859923	-0.195051	-0.266538	-0.195145	-0.052803	-0.056772	0.010535
energy_100g	0.460877	0.167224	0.252389	-0.073110	-0.204592	0.064037	-0.257461	-0.048484	0.039040	0.757934
fat_100g	0.443342	-0.125166	-0.046490	-0.173849	0.252741	-0.035036	-0.488655	-0.479812	0.273358	-0.382836
saturated_fat_100g	0.419225	-0.081023	-0.222022	-0.241252	0.226447	-0.404182	0.039444	0.280408	-0.643937	-0.026602
carbohydrates_100g	0.271373	0.460129	0.312140	0.056392	-0.434198	0.214779	-0.117385	0.271502	-0.165185	-0.514384
sugars_100g	0.237519	0.472883	-0.316729	-0.012159	-0.073543	-0.011855	0.592735	-0.512426	0.003429	0.005074
fiber_100g	0.088457	0.266601	0.543367	0.342821	0.665315	-0.152780	0.181089	0.061005	0.076045	-0.006480
proteins_100g	0.208303	-0.422771	0.383859	-0.082439	-0.378908	-0.486233	0.388629	-0.012971	0.282145	-0.113710
salt_100g	0.222557	-0.498769	0.198890	0.192199	0.044624	0.607080	0.246568	-0.222620	-0.379170	-0.023177
utrition_score_fr_100g	0.404285	-0.090490	-0.355583	0.046590	0.150988	0.292721	0.213442	0.544306	0.499417	-0.005343

6. Analyse Multivariée: ACP

Energy, Fat, Saturated_fat ont des charges positives importantes sur la composante 1.
Carbohydrates, sugar ont une association plus importantes avec la composante 2.

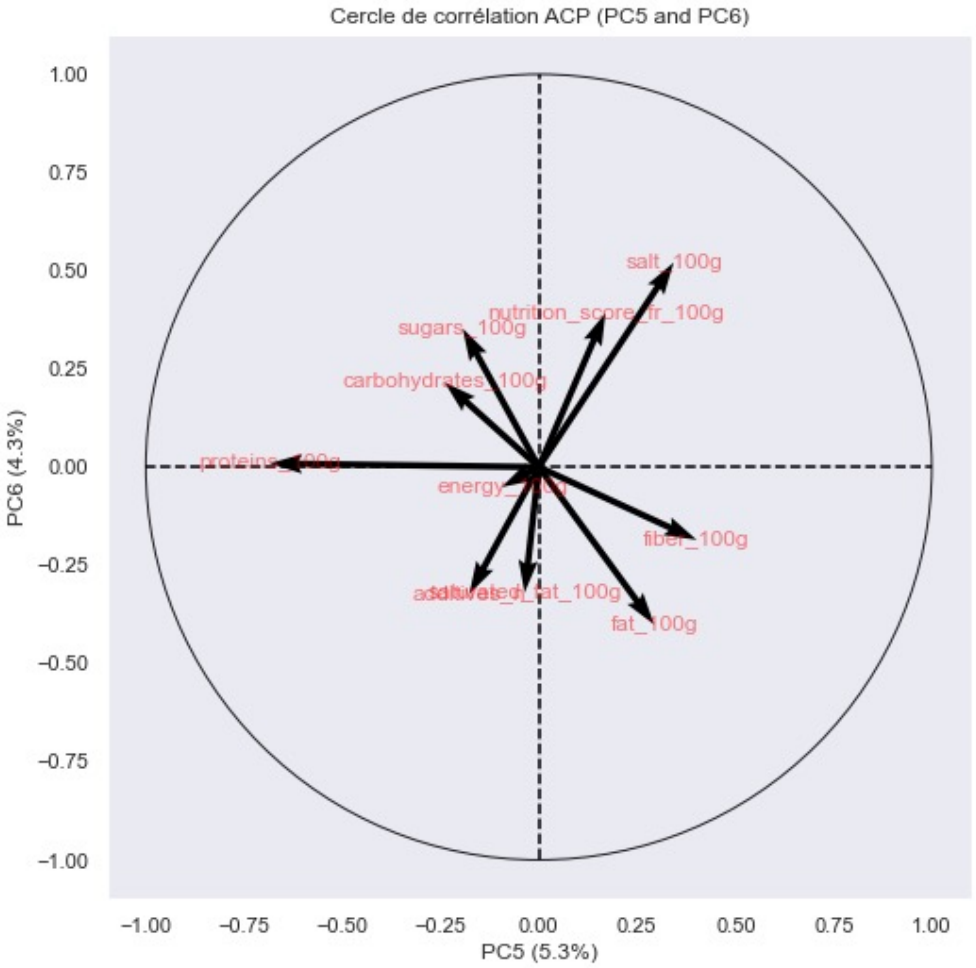


Protéines, Fibres ont une association plus importantes avec la composante 3.
Additives ont une corrélation plus importantes avec la composante 4.

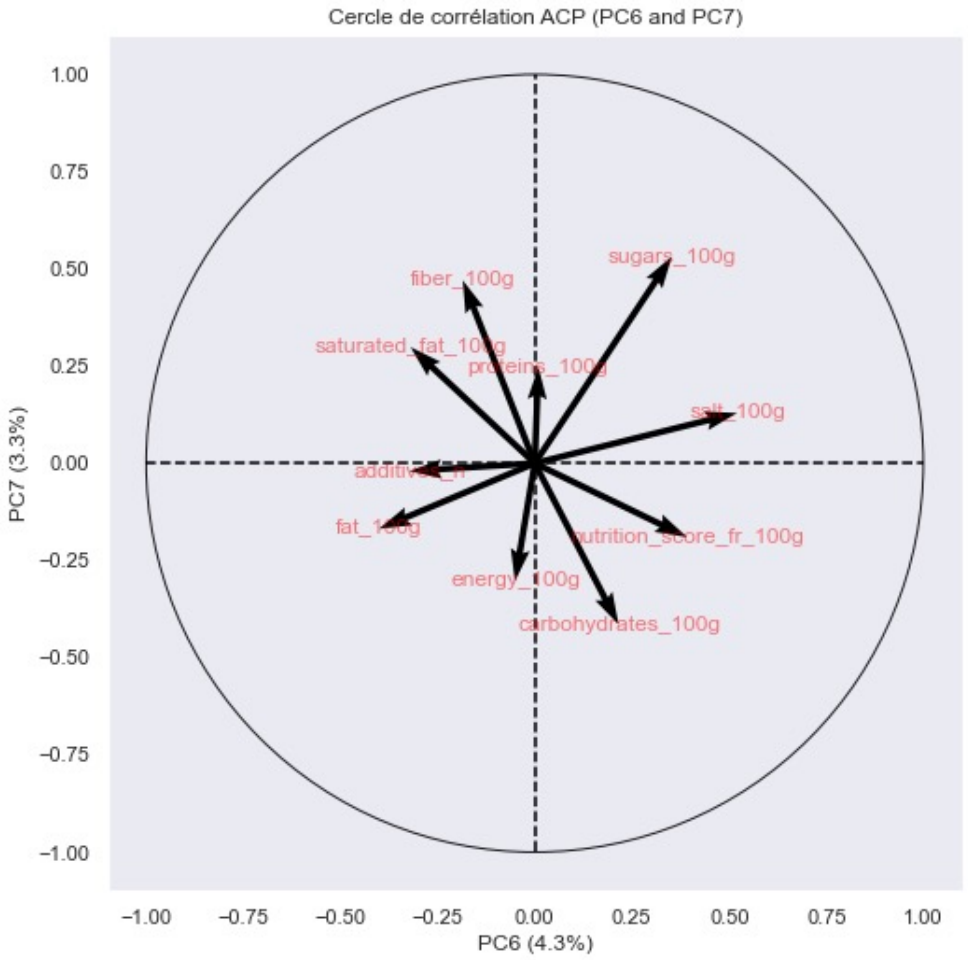


6. Analyse Multivariée: ACP

Fibres a une charge positive plus importante sur la composante 5.

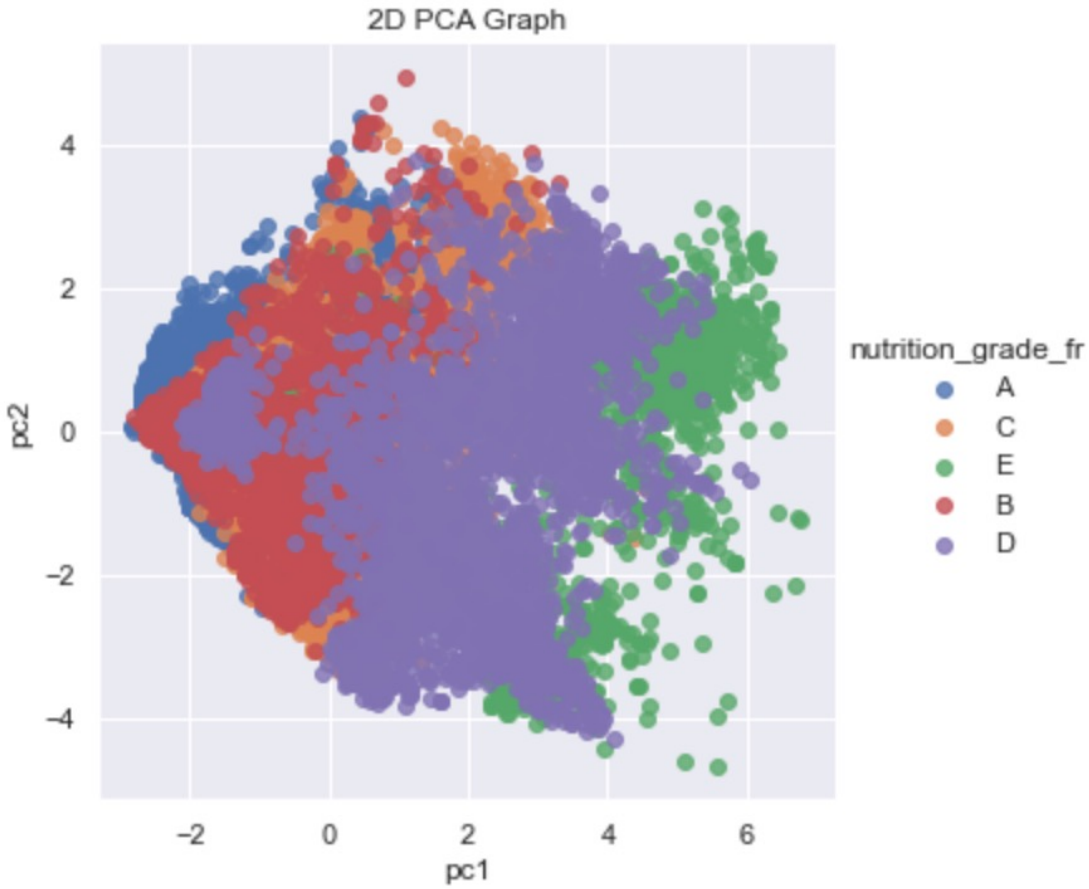


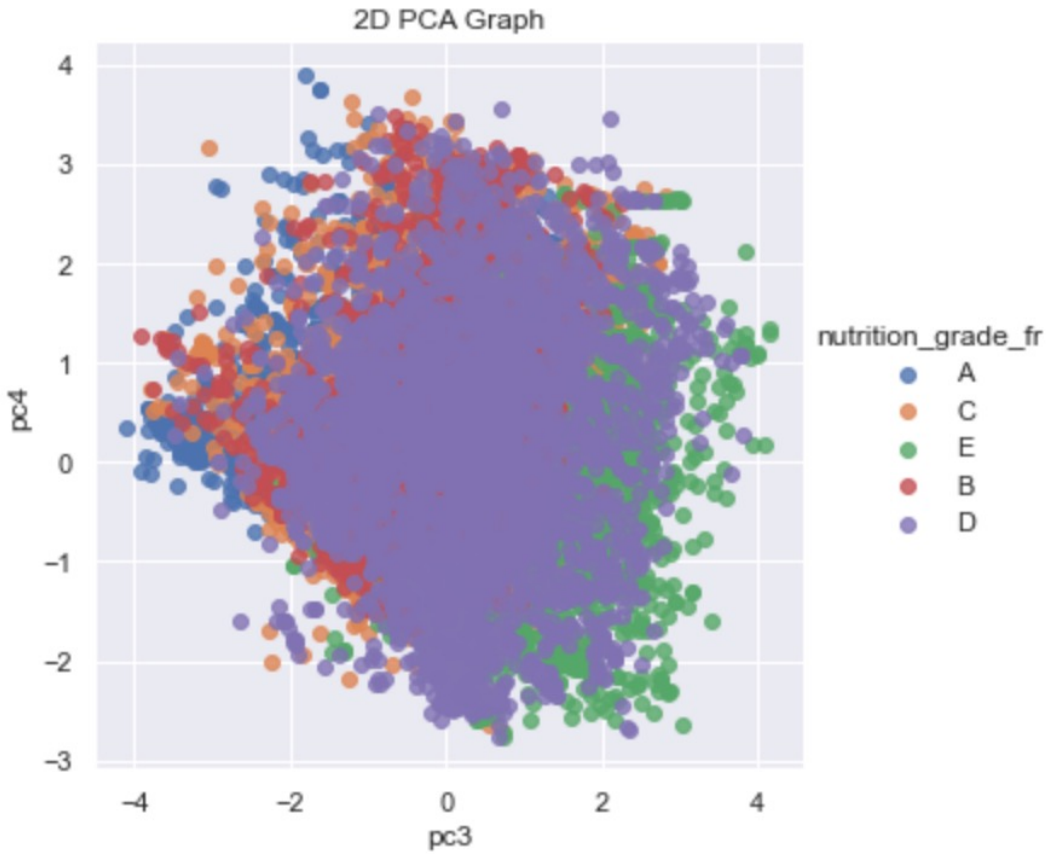
Sucres et fibres ont une association plus importante avec la composante 7.



6. Analyse Multivariée: ACP

Les grades E a une association plus importante et positive avec la composante 1 et 3.
Les grades A, B, et C ont une corrélation plus importantes et positives avec les composantes 2 et 4.





7. Conclusion: ACP

Notre jeu de données contient toutes les données nécessaires à notre idée d'application de recommandation pour les personnes atteintes de L'Ostéoporose.

- protéines (proteins_100g),
- fibres (fiber_100g)
- sucre (sugar_100g)
- sel (salt_100g)
- énergie (energy_100g)
- nutri-score (nutrition_grade_fr),

- En effet ces personnes doivent suivre un régime alimentaire en évitant certains aliments notamment le sel, le sucre etc. tout en privilégiant les fibres et les protéines issues des céréales et légumes.

- L'application sera un moteur de vérification de la compatibilité d'un produit pour personnes atteintes de l'Ostéoporose. Elle recommandera ensuite des produits riches en fibres et en protéines avec le grade le plus haut possible.