#3 CONCEVEZ UNE APPLICATION AU SERVICE DE LA SANTE PUBLIQUE

SOUTENANCE EMILIE GROSCHÊNE LE 04/03/2022
EVALUATEUR: RICK BOURGET
MENTOR: LEA NACCACHE

Sommaire

I. IDÉE D'APPLICATION

I. Idée d'application

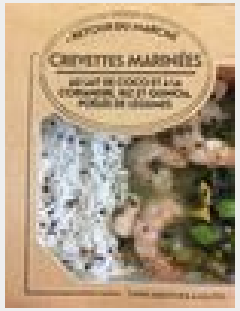
- Suite à l'appel à projet lancé par Santé publique France visant à trouver des **idées innovantes d'application en lien avec l'alimentation** et à partir de la base de données mise à disposition par **Open Food Facts**,
- **classement automatique** des produits en fonction de leur teneur en graisses ou sucres (lipides vs glucides)
- avec ajout d'un **logo** représentant cette caractéristique sur les fiches produits concernées
- et proposition **d'alternatives faibles en glucides et lipides** quand c'est possible

I. Idée d'application (exemples)

Pure via stevia - 250 g

NUTRI-SCORE D
NOVA 4

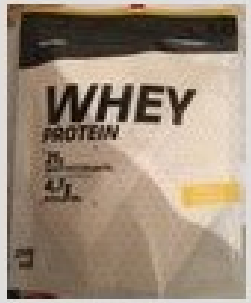
Riche en glucides



Crevettes marinees - Picard - 350 g e

NUTRI-SCORE B
NOVA 3

Faible en glucides et lipides



Whey Protein

NUTRI-SCORE C
NOVA ?

Faible en glucides et lipides

Huile d'olive vierge extra Bio Classico 75 CL - Carapelli

NUTRI-SCORE C
NOVA 2

Riche en lipides

Nouveau logo facile à interpréter

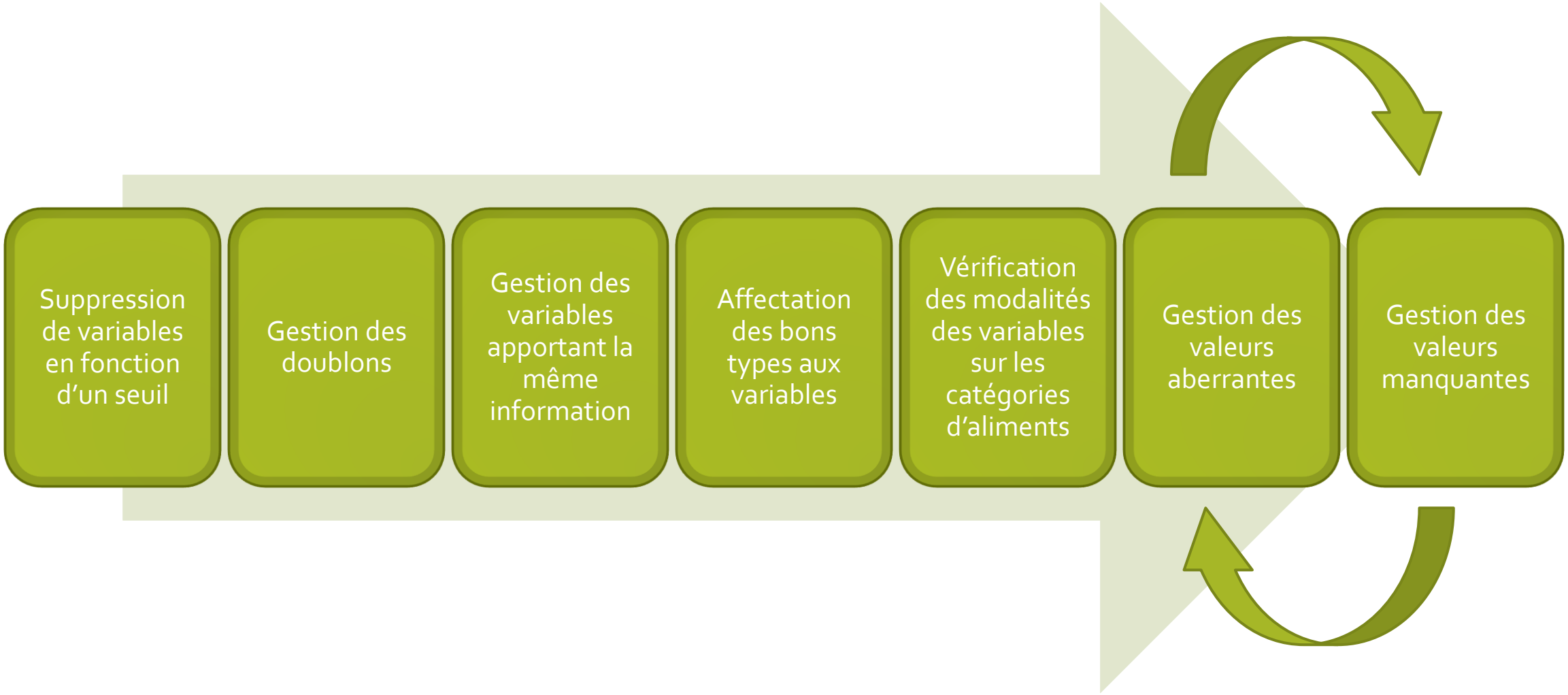
I. Idée d'application (exemples)

Suggestion de produits de même catégorie faibles en glucides et lipides

product_name	main_category_en	fat_100g	carbohydrates_100g	Clusters_str
health valley organic, soup, tomato	tomato soups	1.040000	7.920000	Faible en glucides et lipides
roasted cherry tomato and mascarpone soup	tomato soups	1.700000	4.700000	Faible en glucides et lipides
tomato basil soup	tomato soups	1.230000	9.430000	Faible en glucides et lipides
tomato basil soup	tomato soups	2.074700	7.053900	Faible en glucides et lipides
soup, tomato	tomato soups	0.000000	13.700000	Faible en glucides et lipides
campbell's soup tomato	tomato soups	1.250000	14.170000	Faible en glucides et lipides
tomato soup condensed	tomato soups	1.700000	4.700000	Faible en glucides et lipides
campbell's condensed soup tomato	tomato soups	2.500000	18.330000	Faible en glucides et lipides
tomato	tomato soups	0.491803	5.901639	Faible en glucides et lipides
soupe aux tomates	tomato soups	1.200000	13.600000	Faible en glucides et lipides
campbell's soupe tomate	tomato soups	0.800000	16.000000	Faible en glucides et lipides

II. OPÉRATIONS DE NETTOYAGE

II. Opérations de nettoyage



II. Opérations de nettoyage (détail)

- *Suppression de variables en fonction d'un seuil (1% remplissage)*

```
def shape_total_nan(dataframe):
    '''Fonction qui retourne le nombre de lignes, de variables, le nombre total de valeurs manquantes et
    le pourcentage associé'''
    missing = dataframe.isna().sum().sum()
    missing_percent = round(missing / (dataframe.shape[0] * dataframe.shape[1]) * 100, 2)

    print(f"Nombre de lignes: {dataframe.shape[0]}")
    print(f"Nombre de colonnes: {dataframe.shape[1]}")
    print(f"Nombre total de NaN du dataset: {missing}")
    print(f"% total de NaN du dataset: {missing_percent}%")
```

```
Nombre de lignes: 2053679
Nombre de colonnes: 187
Nombre total de NaN du dataset: 307558533
% total de NaN du dataset: 80.09%
```

```
def min_fill_threshold(dataframe, min_threshold_percent):
    '''Fonction qui renvoie le dataframe supprimé des variables qui sont sous le seuil de remplissage
    indiqué en entrée'''
    return dataframe.loc[:, dataframe.isna().mean() <= ((100-min_threshold_percent)/100)]
```

```
Nombre de lignes: 2053679
Nombre de colonnes: 90
Nombre total de NaN du dataset: 108566833
% total de NaN du dataset: 58.74%
```

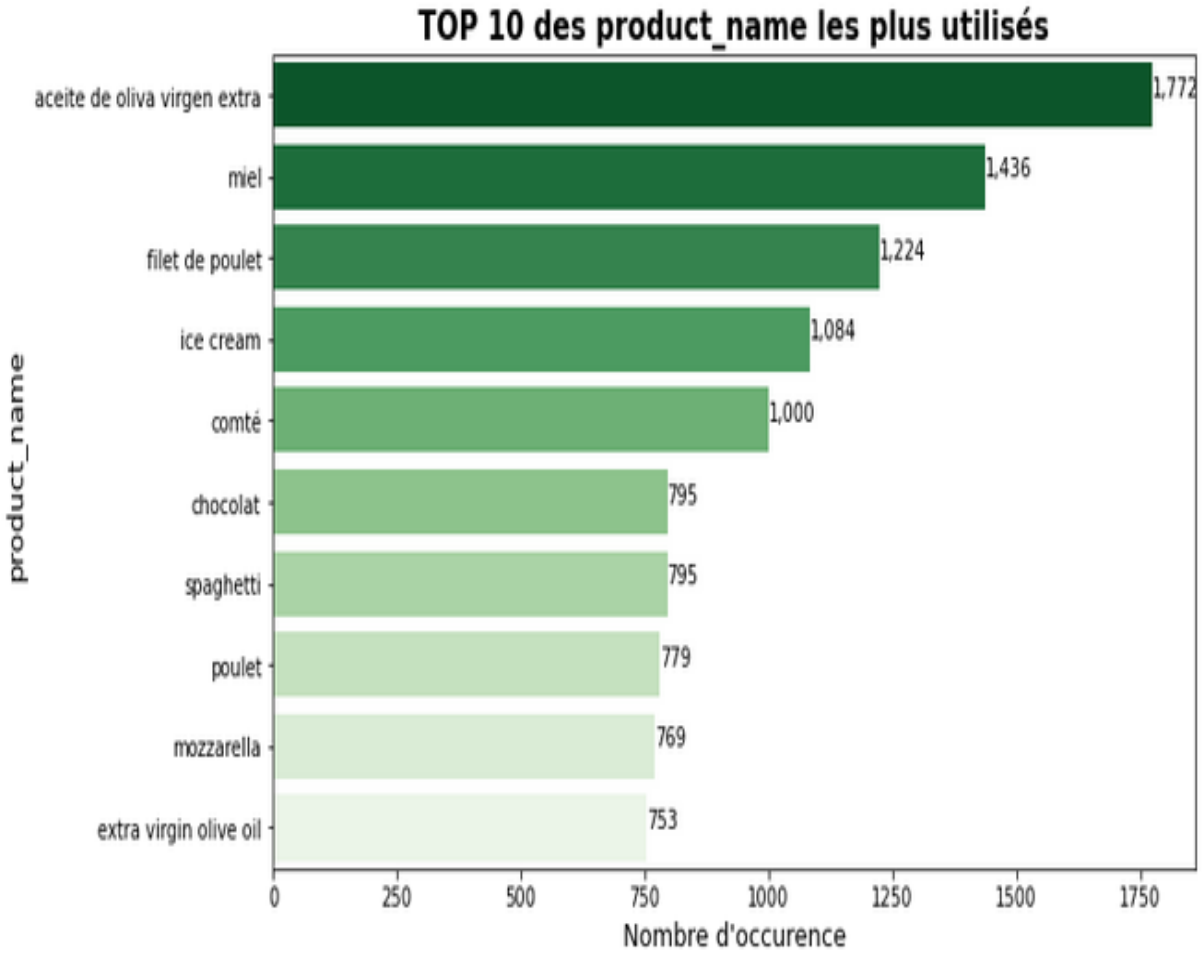
II. Opérations de nettoyage (détail)

- *Gestion des doublons*

  - **Intégraux**: suppression de 4 lignes totalement identiques

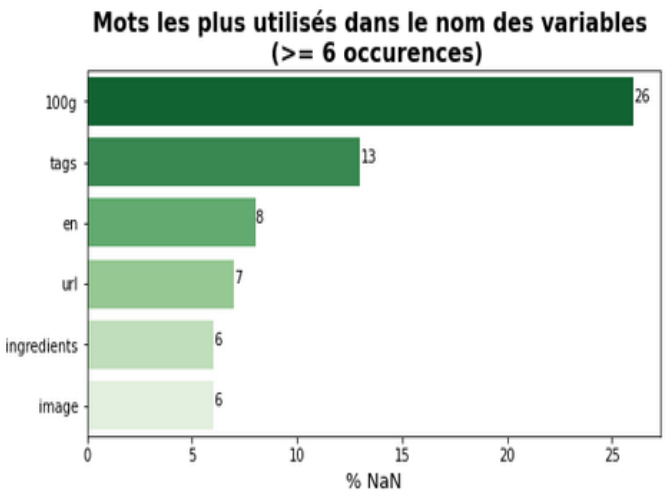
  - **Code**: suppression des 11 lignes dupliquées avec le plus grand nombre de valeurs manquantes par ligne

  - **Product_name**: après avoir transformé les modalités de cette colonne en minuscules, plus de 884 K produits ont le même nom. Il peut exister plusieurs produits identiques pour une même marque mais les codes barres étant différents, nous ne les considérons pas comme des doublons



II. Opérations de nettoyage (détail)

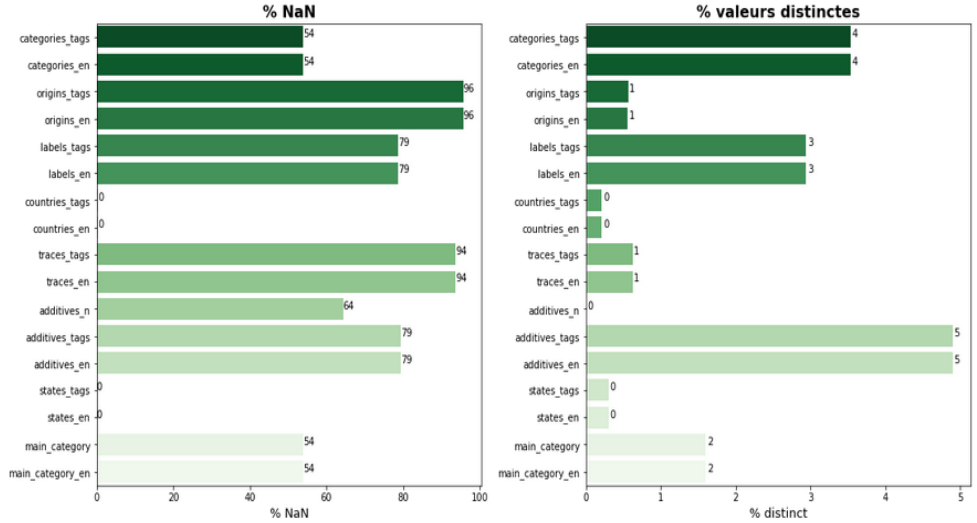
- *Gestion des variables apportant la même information*
  - Création d'une fonction à appliquer sur les mots qui reviennent dans le nom des variables permettant de récupérer toutes les variables ayant le même préfixe




```
liste_col_en = liste_col_prefixe(openfoodfacts, 'en')
```

```
describe_variables_light(openfoodfacts.iloc[:,openfoodfacts.colu
```

Variable name	Variable type	Example
categories_tags	object	en:groceries,en:condiments,en:sauces,en:mustards
categories_en	object	groceries,condiments,sauces,mustards
origins_tags	object	en:spain
origins_en	object	spain
labels_tags	object	fr:delois-france
labels_en	object	fr:delois-france
countries_tags	object	en:france
countries_en	object	france



- Décision de conserver une variable plutôt qu'une autre basée sur le **% de valeurs manquantes** et de **valeurs distinctes** puis choix validé sur quelques **exemples** pris au hasard

II. Opérations de nettoyage (détail)

- *Affectation des bons types aux variables*
  - **float32** pour les variables qui se terminent par _100g
  - **object** pour les variables se terminant par _tags et _en
  - **datetime** pour la variable last_modified_t
  - les autres variables sont au bon format
- *Vérification et modification des modalités des variables sur les catégories d'aliments*

```
openfoodfacts.loc[openfoodfacts['pnns_groups_1'] == 'sugary-snacks', 'pnns_groups_1'] = 'sugary snacks'
```

```
openfoodfacts.loc[openfoodfacts['pnns_groups_1']== 'unknown', 'pnns_groups_1'] = np.nan
```

```
openfoodfacts.loc[openfoodfacts['pnns_groups_2'] == 'pizza pies and quiche', 'pnns_groups_2'] = 'pizza pies
openfoodfacts.loc[openfoodfacts['pnns_groups_2'] == 'legumes', 'pnns_groups_2'] = 'vegetables'
```

```
openfoodfacts.loc[openfoodfacts['pnns_groups_2'] == 'unknown', 'pnns_groups_2'] = np.nan
```

II. Opérations de nettoyage (détail)

- *Remplissage des variables liées aux catégories d'aliments*

```
def mapping(dataframe, var_to_map, var_mapping):
    table_mapping = dataframe.groupby([var_mapping, var_to_map, 'last_modified_date'])['code'].count().reset
    table_mapping = table_mapping.sort_values('last_modified_date', ascending = True)
    table_mapping = table_mapping.drop_duplicates(subset = var_mapping, keep = 'last')

    # Merge du dataframe avec la table de mapping
    df = pd.merge(left = dataframe, right = table_mapping[[var_mapping, var_to_map]],
                  how = "left", on = var_mapping)

    # Lorsque la var_to_map est NaN, je complète avec la valeur du mapping
    df[f'{var_to_map}_x'] = np.where(df[f'{var_to_map}_x'].isnull(),
                                     df[f'{var_to_map}_y'],
                                     df[f'{var_to_map}_x'])

    # On supprime la variable issue de la jointure et on enlève le suffixe _x
    df.rename(columns={f'{var_to_map}_x': var_to_map}, inplace = True)
    df.drop(f'{var_to_map}_y', axis = 'columns', inplace = True)

    return df
```

Dataset après passage en revue des différents groupes de variables:

- Nombre de lignes: 2 053 664
- Nombre de variables: **41**
- **58.95%** de valeurs manquantes dans le dataset

- Mapping de **main_category_en** à l'aide du **product_name** (taux de remplissage passe de 54% à 63%)
- Mapping des **pnns_groups_1** à l'aide du **product_name**, des **main_category_en** puis des **pnns_groups_2** (taux de remplissage passe de 37% à près de 100%)
- Mapping des **pnns_groups_2** à l'aide du **product_name** et des **main_category_en** (taux de remplissage passe de 37% à 62%)

II. Opérations de nettoyage (détail)

- *Gestion des valeurs aberrantes*

  - La **distribution** de nos variables **n'étant pas Normale**, nous ne pouvons utiliser la méthode interquartile pour définir et supprimer les outliers du dataset.

  - Notre approche pour repérer ces outliers sera une **approche logique**:

Nutriments sont compris entre 0 et 100 (hors énergie)

Sous catégorie est <= à la variable « parent »

Somme des nutriments est <= 100 (hors énergie et variables secondaires)

Energie maximum = 3 766 KJ

II. Opérations de nettoyage (détail)

- *Gestion des valeurs manquantes*

  - lignes avec **aucun nutriment / product_name / pnns_groups_2** renseignés => **suppression**

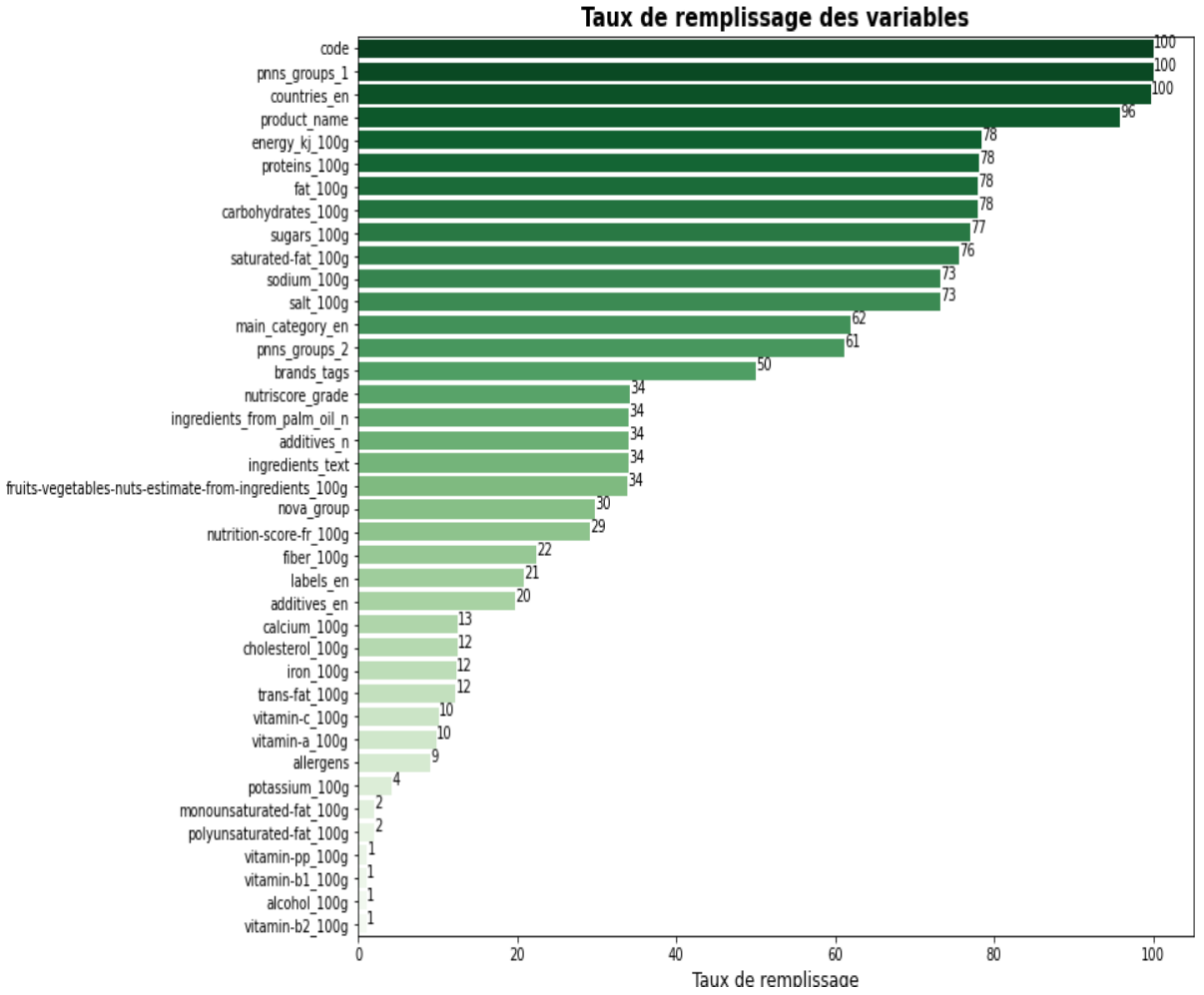
  - variables permettant de **caractériser les aliments:**
    - ✓ constante **'unknown'**
    - ✓ constante **'None'** pour les variables **additives_en, labels_en et allergens**

  - variables de **compte** => 0

  - **pourcentage de NaN faible** (<=30%)
    - ✓ **médiane** pour les **variables quantitatives**
    - ✓ **mode pour les variables qualitatives**

  - **variables corrélées entre elles** => **iterative imputer**

  - pour les **autres variables quantitatives** ainsi que les variables **nova_group** et **nutriscore_grade** => **Knn** (K Nearest Neighboors)
    - ✓ le KNN étant adapté aux petits jeux de données car très consommateur en ressources, les variables quantitatives ont été finalement complétées par la médiane des pnns_groups_2 et unknown pour le nova_group et nutriscore_grade

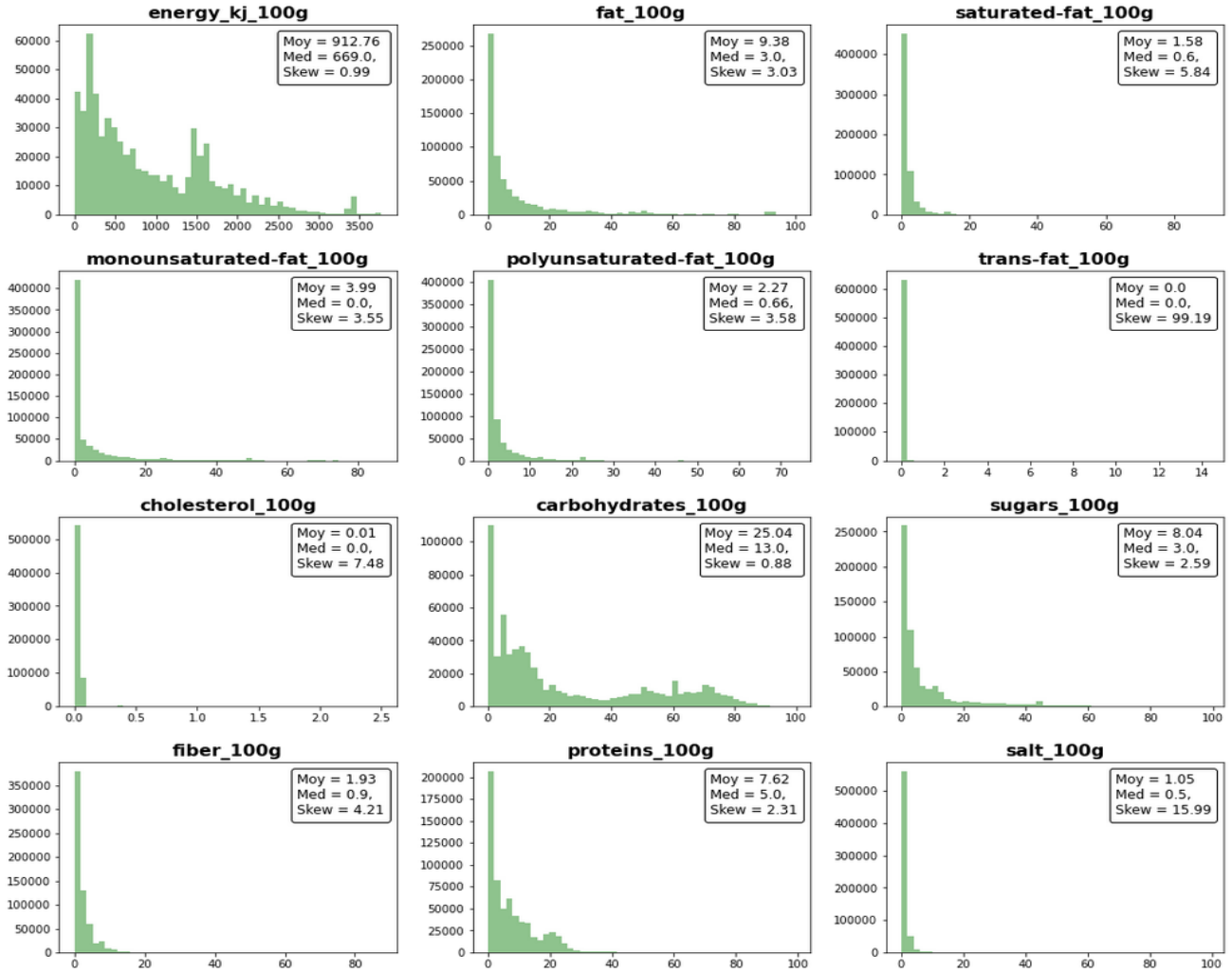



```
Nombre de lignes: 632776
Nombre de colonnes: 39
Nombre total de NaN du dataset: 0
% total de NaN du dataset: 0.0%
```

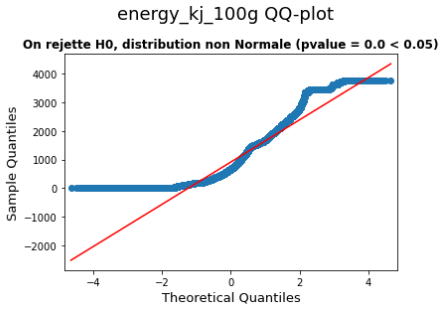
III. ANALYSE EXPLORATOIRE

III. Analyse exploratoire (analyse univariée)

- **Variables quantitatives continues: mesures de tendance centrale**

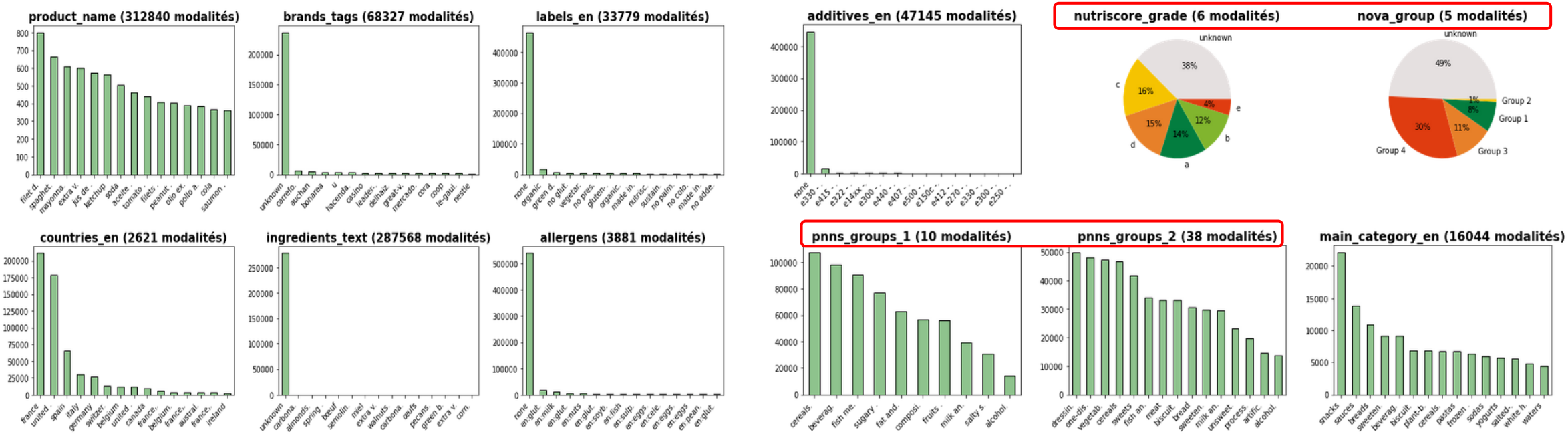


- La grande majorité des distributions montre un **pic aux alentours de la valeur 0**. Il ne s'agit probablement pas de valeurs erronées car certains produits peuvent ne pas contenir tel ou tel nutriment (exemple de l'huile qui ne contient pas de sucres etc).
- **Certaines distributions sont bimodales** comme l'énergie ou les fruits/légumes/oléagineux
- La plupart des distributions ne sont pas symétriques mais **étalées vers la droite**
- Les distributions ne suivent pas **une distribution Normale** (test de normalité de Shapiro Wilk)



III. Analyse exploratoire (analyse univariée)

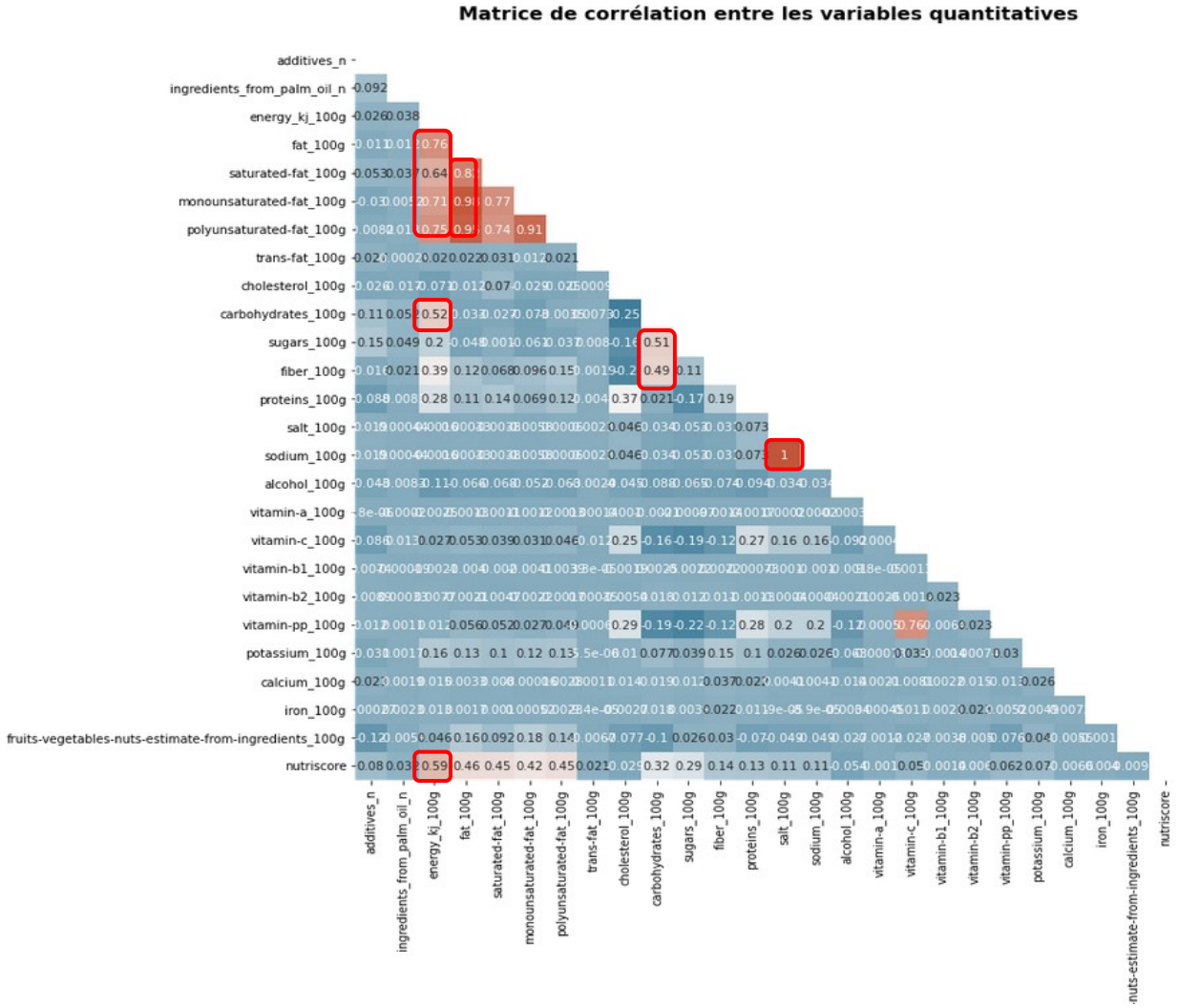
- **Variables qualitatives**



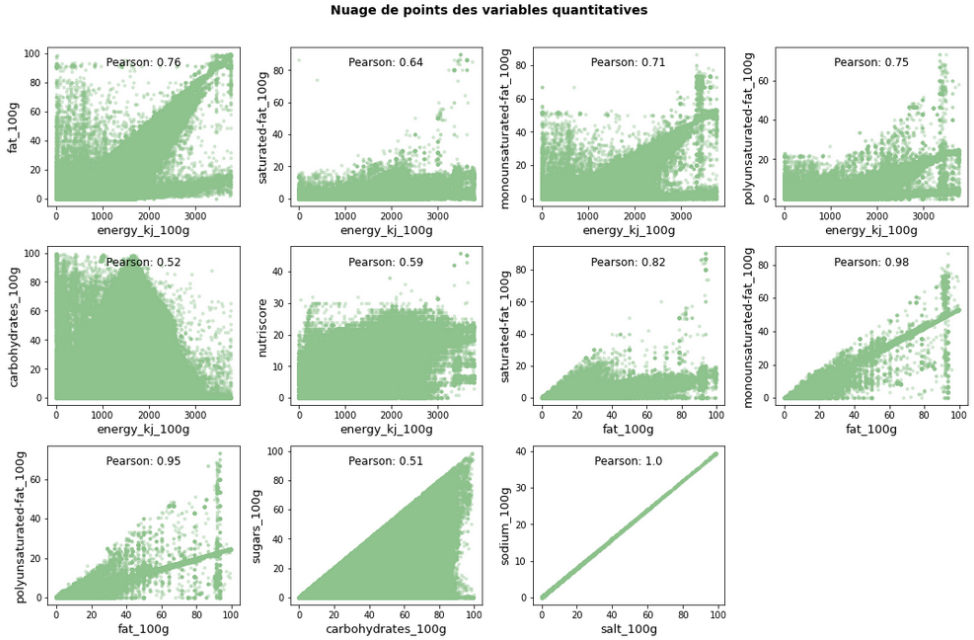
Certaines variables telles que le nom du produit, la marque, les labels etc ont de très nombreuses modalités. Seules les variables **pnns_groups_1, pnns_groups_2, nutriscore_grade** ou encore le **nova_group** ont un nombre limité de modalités et pourraient être utilisées lors de **l'analyse bivariée**.

III. Analyse exploratoire (analyse bivariée)

- **2 variables quantitatives**

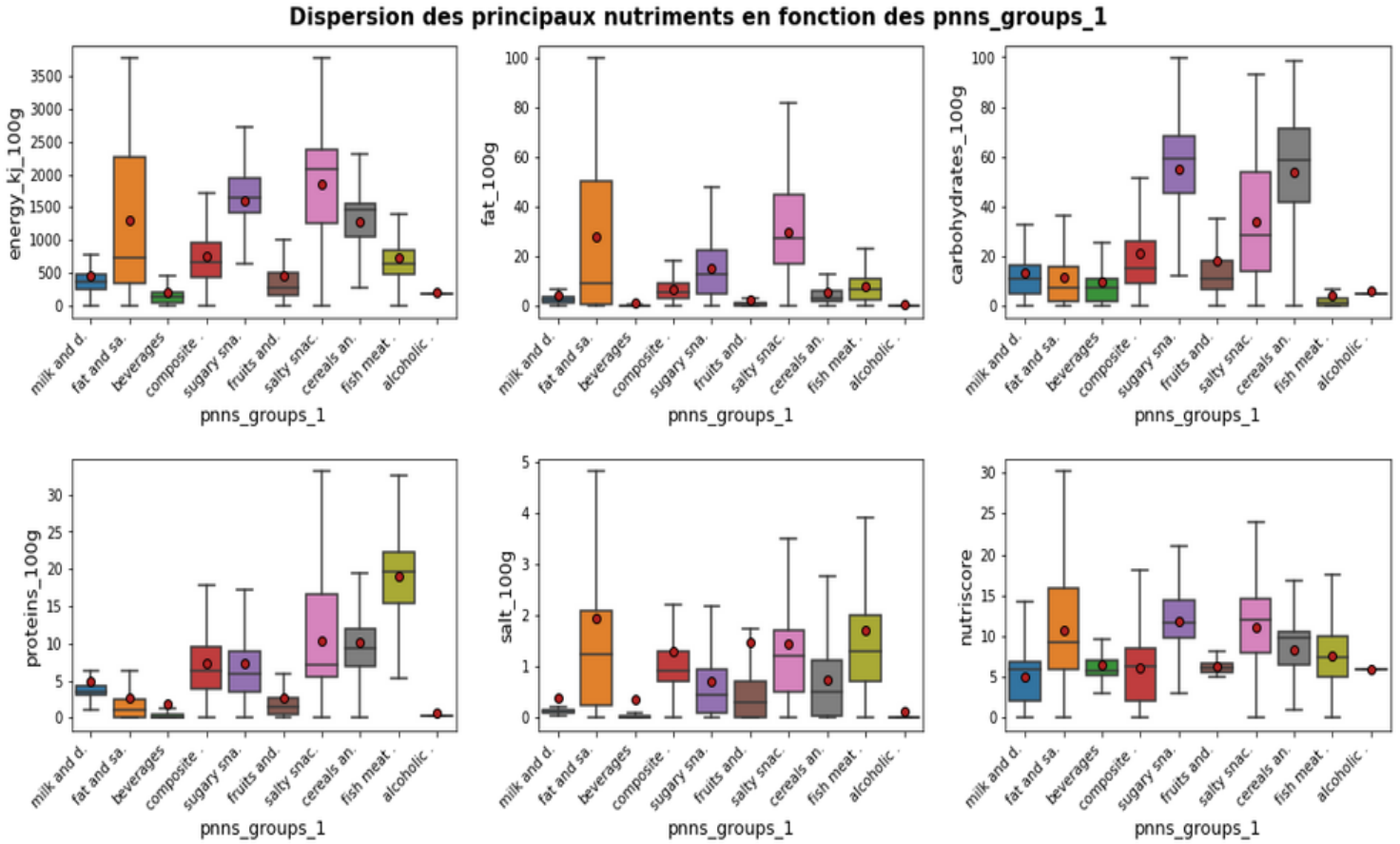


Les variables les plus corrélées entre elles sont (coefficient de Pearson > 0.5):



III. Analyse exploratoire (analyse bivariée)

- **1 variable quantitative et 1 variable qualitative**



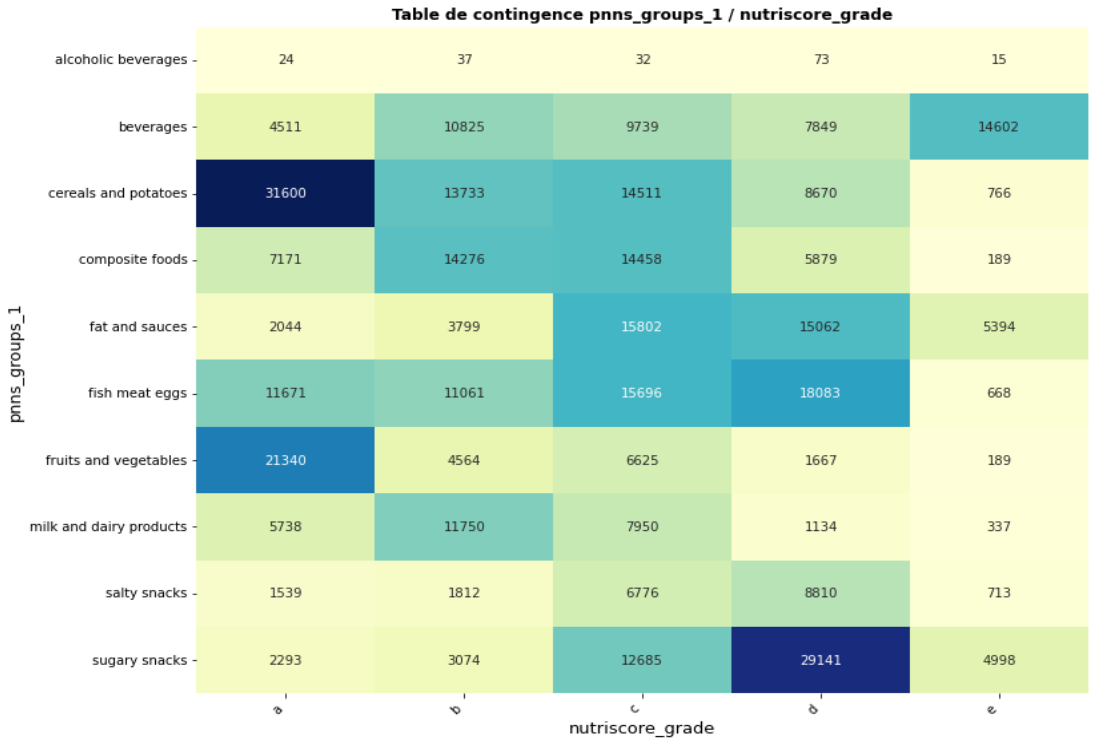

```
Test non paramétrique de Kruskal Wallis:

Variable energy_kj_100g par pnns_groups_1
pvaleur: 0.0 < 0.05 => on rejette H0, les médianes entre les pnns_groups_1 sont différentes
--------------------------------------------------------------------------------
Variable fat_100g par pnns_groups_1
pvaleur: 0.0 < 0.05 => on rejette H0, les médianes entre les pnns_groups_1 sont différentes
--------------------------------------------------------------------------------
Variable carbohydrates_100g par pnns_groups_1
pvaleur: 0.0 < 0.05 => on rejette H0, les médianes entre les pnns_groups_1 sont différentes
--------------------------------------------------------------------------------
Variable proteins_100g par pnns_groups_1
pvaleur: 0.0 < 0.05 => on rejette H0, les médianes entre les pnns_groups_1 sont différentes
--------------------------------------------------------------------------------
Variable salt_100g par pnns_groups_1
pvaleur: 0.0 < 0.05 => on rejette H0, les médianes entre les pnns_groups_1 sont différentes
--------------------------------------------------------------------------------
Variable nutriscore par pnns_groups_1
pvaleur: 0.0 < 0.05 => on rejette H0, les médianes entre les pnns_groups_1 sont différentes
--------------------------------------------------------------------------------
```

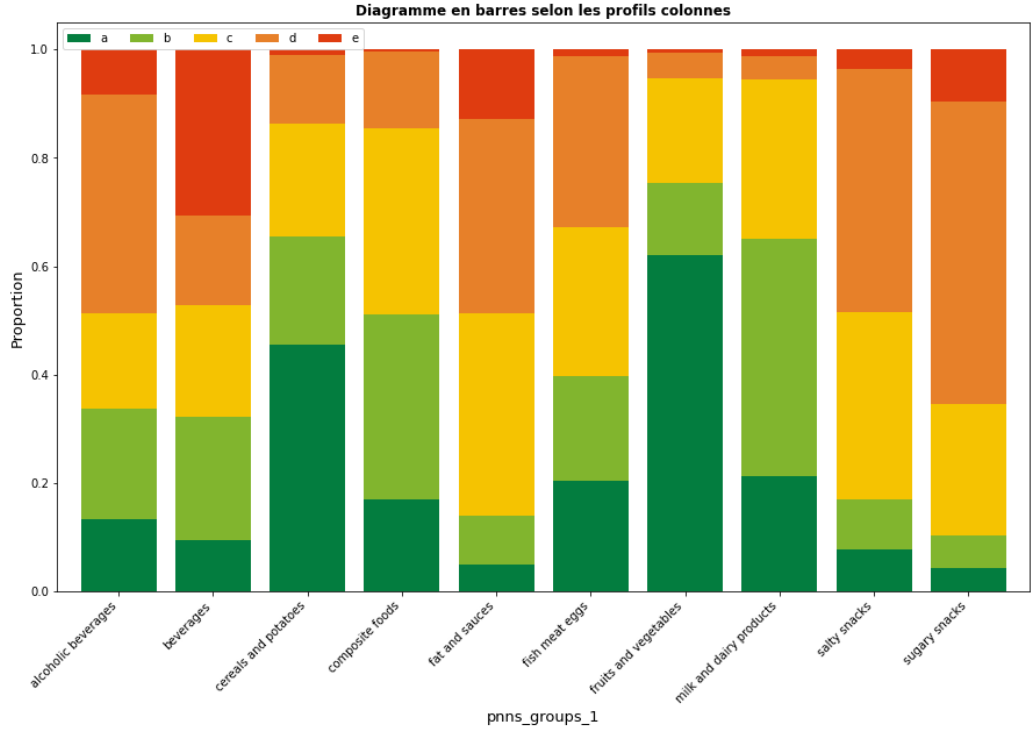
Test paramétrique: ANOVA suivi de TUKEY
Test non paramétrique: Kruskall-Wallis suivi de Mann-Whitney

Les condition d'application de l'ANOVA n'étant pas respectées (distribution non normale), **le test non paramétrique de Kruskall-Wallis qui est un test sur la moyenne des rangs confirme que les médianes entre pnns_groups_1 diffèrent.**

III. Analyse exploratoire (analyse bivariée)

- **2 variables qualitatives**






```
Chaque effectif de la table de contingence >= 5 => Test du Chi2 applicable
--------------------------------------------------------------------------
Chi2: 184822.16613753958
Degrees of freedom: 36
--------------------------------------------------------------------------
pvalue: 0.0 < 0.05 => on rejette H0, les variables sont dépendantes
```

Le test applicable est le test du **Chi2**. Les conditions d'utilisation de ce test sont:
- chaque effectif du tableau doit être supérieur ou égal à 5

Les hypothèses sont:
- Ho: indépendance entre les 2 variables qualitatives
- HA: association entre les 2 variables qualitatives
- alpha = 0.05

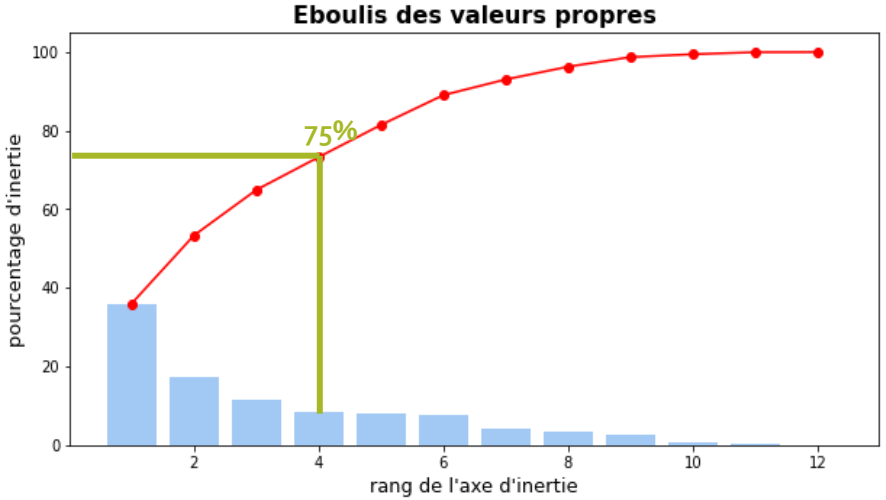
Le test du Chi2 confirme l'hypothèse **d'association entre les variables pnns_groups_1 et nutriscore_grades**.

Les meilleures notes semblent être attribuées aux fruits et légumes et céréales / pommes de terre alors que les moins bonnes sont données aux boissons, gras et sauces et snacks sucrés.

III. Analyse exploratoire (analyse multivariée)

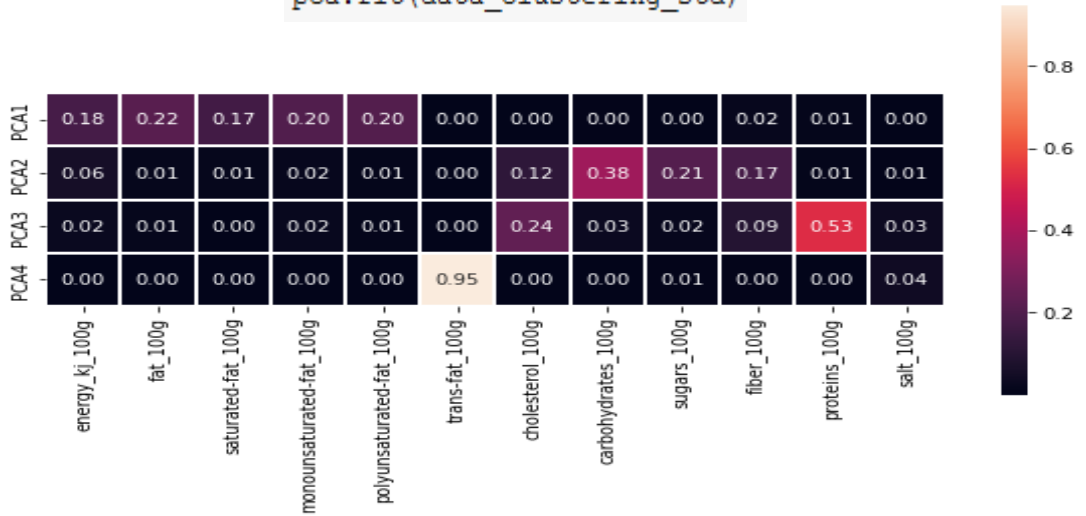
- **Réduction de dimension par Analyse en Composantes Principales**

```
# Standardisation
scaler = StandardScaler()
data_clustering_std = scaler.fit_transform(data_clustering)
```



Le graphique montre la **quantité de variance capturée** (sur l'axe des y) en fonction du **nombre de composantes que nous incluons** (sur l'axe des x). Une règle empirique consiste à préserver environ 80 % de la variance. Il faudrait donc garder 5 composantes. Comme l'on observe un pallier à partir de la 4ème composante, nous allons réaliser l'ACP sur 4 composantes, capturant ainsi un peu plus de 70% de la variance.

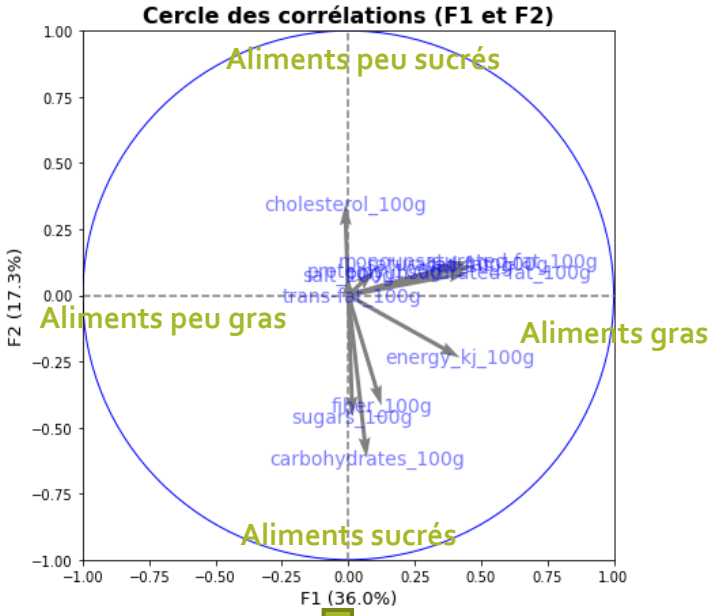
```
# ACP
pca = PCA(n_components = 4)
pca.fit(data_clustering_std)
```

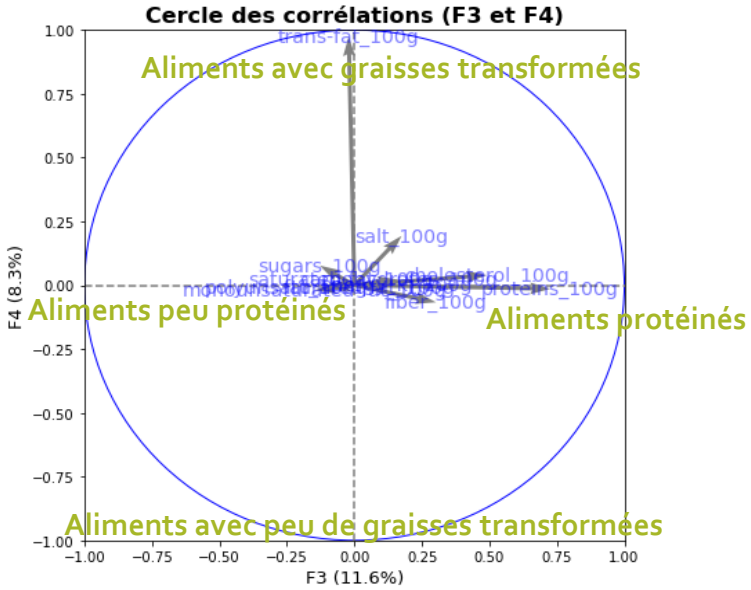


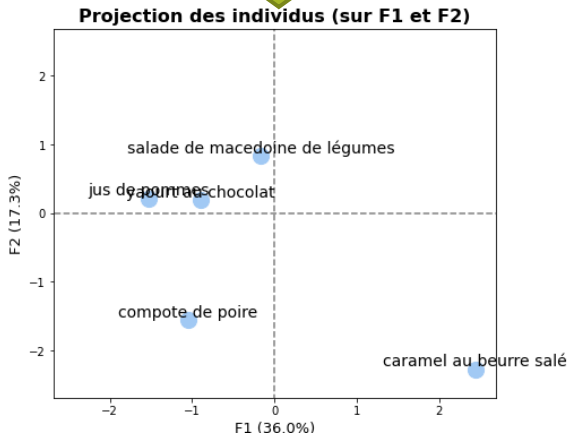
- La première composante principale est constituée de 22% de la variable **lipides,** 20% des variables **graisses monosaturées** et **graisses polysaturées,** 18% de la variable **énergie** et 17% de la variable **graisses saturées**.
- La deuxième composante principale est constituée de 38% de la variable **glucides,** 21% de la variable **sucres,** 17% de la variable **fibres** et 12% de la variable **cholestérol** et 6% de la variable **énergie**.
- La troisième composante principale est constituée de 53% de la variable **protéines,** 24% de la variable **cholestérol** et 9% de la variable **fibres**.
- La quatrième composante principale est constituée de 95% de la variable **graisses transformées** et 4% de la variable **sel**.

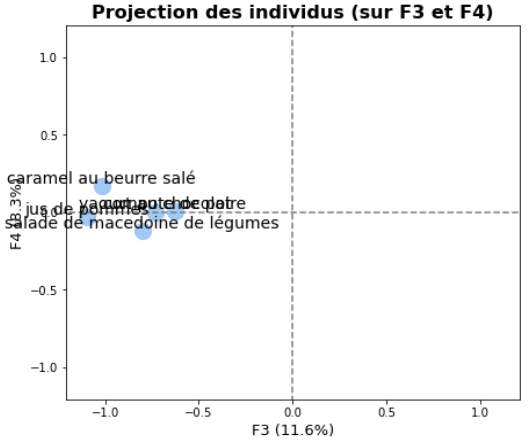
III. Analyse exploratoire (analyse multivariée)

- **Réduction de dimension par Analyse en Composantes Principales**





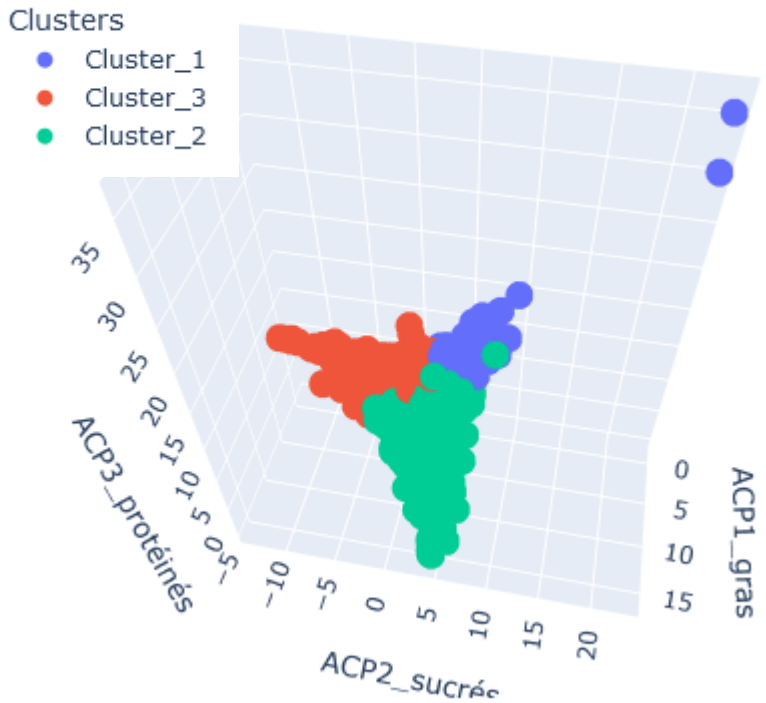




IV. FAISABILITÉ DE L'APPLICATION

IV. Faisabilité de l'application

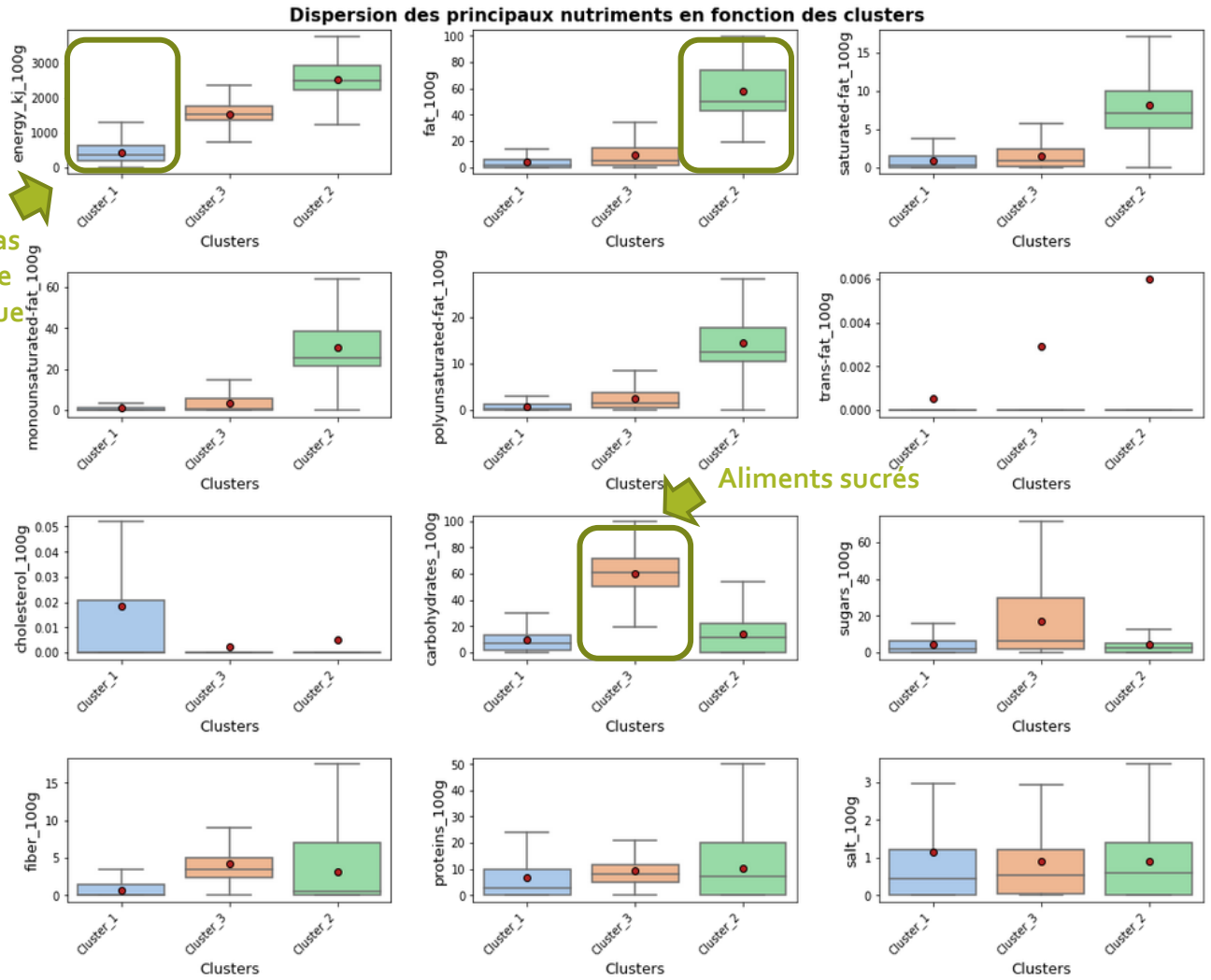
- **K-means avec 3 clusters sur les 4 composantes principales**



Aliments gras

Aliments ni gras ni sucrés, faible apport calorique

Aliments sucrés



Le **cluster 1** semble représenter des aliments plutôt **protéinés,** le **cluster 2** les aliments **gras** et le **cluster 3** les aliments **sucrés** (ACP2 sucrée négative cf cercle de corrélation)

IV. Faisabilité de l'application

- **K-means avec 3 clusters sur les 4 composantes principales**

```
Test non paramétrique de Kruskal Wallis:

Variable fat_100g par Clusters_str
pvaleur: 0.0 < 0.05 => on rejette H0, les médianes entre les Clusters_str sont différentes
-----------------------------------------------------------------------------
Variable carbohydrates_100g par Clusters_str
pvaleur: 0.0 < 0.05 => on rejette H0, les médianes entre les Clusters_str sont différentes
-----------------------------------------------------------------------------
```

Test post hoc non paramétrique de Mann Whitney Clusters / Fat_100g

	Faible en glucides et lipides	Riche en Glucides	Riche en Lipides
Faible en glucides et lipides	1.0	0.0	0.0
Riche en Glucides	0.0	1.0	0.0
Riche en Lipides	0.0	0.0	1.0

```
Clusters_str
Faible en glucides et lipides     1.49
Riche en Glucides                 5.09
Riche en Lipides                 50.00
Name: fat_100g, dtype: float64
```

Test post hoc non paramétrique de Mann Whitney Clusters / carbohydrates_100g

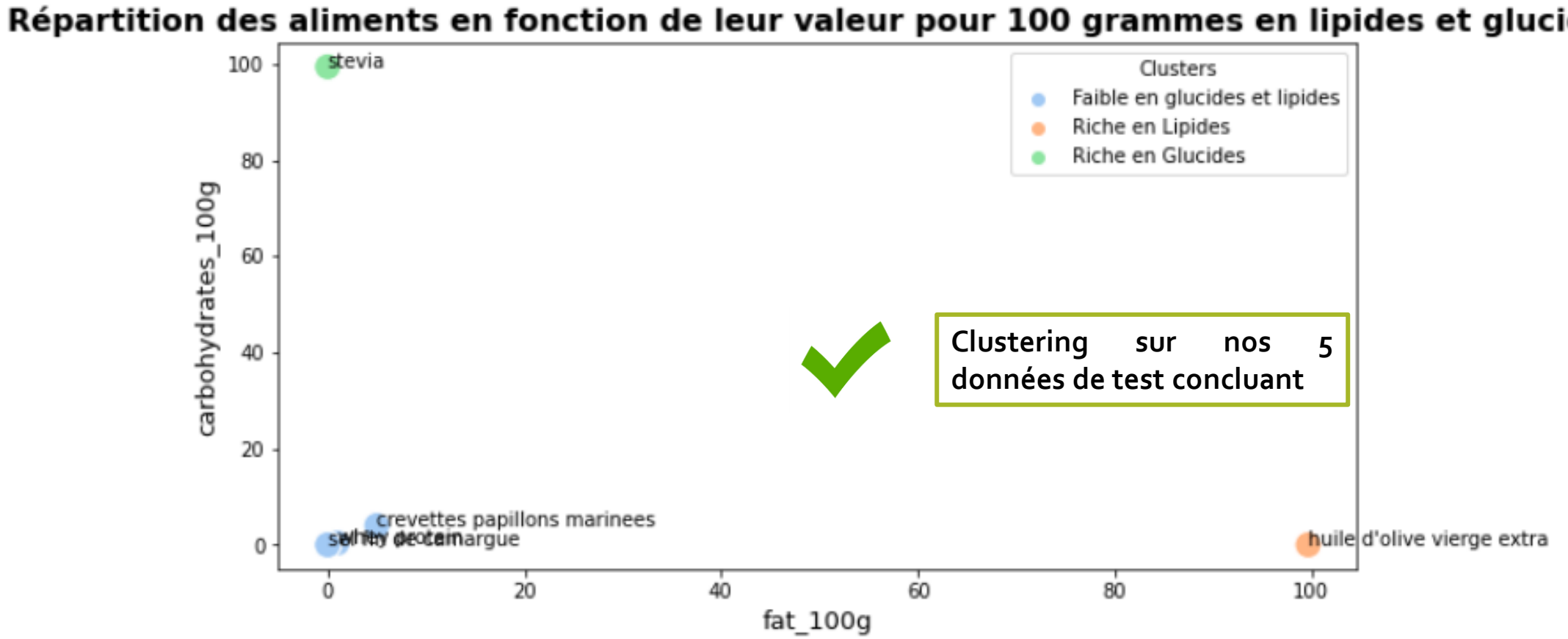
	Faible en glucides et lipides	Riche en Glucides	Riche en Lipides
Faible en glucides et lipides	1.0	0.0	0.0
Riche en Glucides	0.0	1.0	0.0
Riche en Lipides	0.0	0.0	1.0

```
Clusters_str
Faible en glucides et lipides     7.2
Riche en Glucides                61.0
Riche en Lipides                 11.4
Name: carbohydrates_100g, dtype: float64
```

IV. Faisabilité de l'application

- **Test du clustering sur 5 produits sortis initialement du dataset**

```
data_test_std = scaler.transform(data_test)
scores_pca_test = pca.transform(data_test_std)
kmeans_pca.predict(scores_pca_test)
```



V. CONCLUSIONS

V. Conclusion

- **Faisabilité du projet**

➢ **K-Means avec 3 clusters sur les 4 composantes principales concluant:**

- ✓ **Tests de Kruskal Wallis + Mann Whitney** confirment que la teneur en lipides et glucides diffère entre nos clusters
- ✓ Test sur 5 nouvelles données concluant, les **aliments sont affectés dans les bons clusters**

➢ **Axes d'amélioration à prévoir:**

- ✓ **Gestion des valeurs manquantes et aberrantes** devrait être soumise à validation par un **nutritionniste** (connaissance métier)
- ✓ Certaines **variables** très **peu remplies** devraient être **écartées** d'office
- ✓ Différents tests sur le nombre de composantes principales et de clusters pourraient être faits afin **d'optimiser les clustering** en proposant notamment une **classification plus détaillée** (nombre de clusters plus important)

MERCI