STAT542

May 11, 2021

[1]:

```
import os
import warnings
%matplotlib inline
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('seaborn')
sns.set_style('whitegrid')
plt.rcParams['axes.labelsize']=12
import matplotlib.gridspec as gridspec
from wordcloud import WordCloud, STOPWORDS
pd.options.mode.chained_assignment = None # Warning for chained copies disabled
```

[2]:

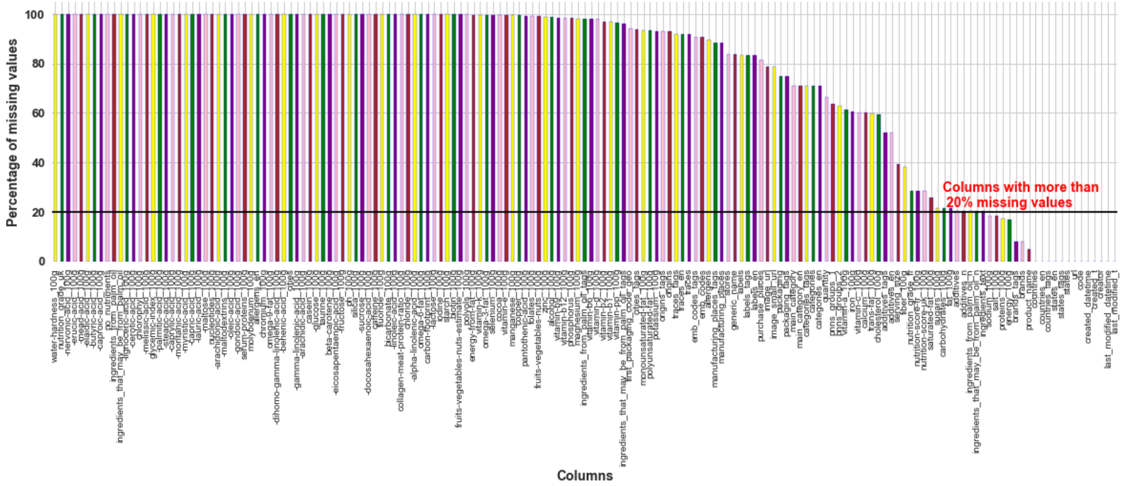
```
off = pd.read_csv("en.openfoodfacts.org.products.tsv", delimiter='\t',␣
 ↪encoding='utf-8')

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

```
C:\Users\arkam\anaconda3\lib\site-
packages\IPython\core\interactiveshell.py:3166: DtypeWarning: Columns
(0,3,5,19,20,24,25,26,27,28,36,37,38,39,48) have mixed types.Specify dtype
option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

[3]:

```
plt.figure(figsize=(20, 5)); thresh_pc = 20

percentage=(off.isnull().mean()) * 100
percentage.sort_values(ascending=False).plot.bar(color=('yellow', 'green',␣
 ↪'purple', 'pink', 'brown'),
                                                 edgecolor='b')
plt.axhline(y = thresh_pc, color='k', linestyle='-')
# plt.title('Missing values percentage per column', fontsize=20, weight='bold' )
plt.text(len(off.isnull().sum()/len(off))/1.2, thresh_pc + 12.5,
         'Columns with more than \n %s%s missing values' %(thresh_pc, '%'),␣
 ↪fontsize=14, weight='bold', color='r',
          ha='left', va='top')
```

```
# plt.text(len(off.isnull().sum()/len(off))/1.2, thresh_pc - 5,
#          'Columns with less than \n %s%s missing values' %(thresh_pc, '%'),␣
↪fontsize=14, weight='bold', color='k',
#          ha='left', va='top')
plt.xlabel('Columns', size=15, weight='bold', fontsize=14)
plt.ylabel('Percentage of missing values', weight='bold', fontsize=14)
plt.yticks(weight ='bold', fontsize = 13)

plt.savefig('initial_barplot.png', dpi=300)
plt.show()
```



[3]:
```
off2=off.dropna(thresh=int(0.2*off.shape[0]), axis=1)
print("Shape before cleaning = ", off.shape)
print("Shape after cleaning  = ", off2.shape)
print("We dropped ", off.shape[1]- off2.shape[1]," columns")
off2.columns.values
```

```
Shape before cleaning =  (356027, 163)
Shape after cleaning  =  (356027, 54)
We dropped  109  columns
```

[3]:
```
array(['code', 'url', 'creator', 'created_t', 'created_datetime',
       'last_modified_t', 'last_modified_datetime', 'product_name',
       'quantity', 'packaging', 'packaging_tags', 'brands', 'brands_tags',
       'categories', 'categories_tags', 'categories_en', 'countries',
       'countries_tags', 'countries_en', 'ingredients_text',
       'serving_size', 'additives_n', 'additives', 'additives_tags',
       'additives_en', 'ingredients_from_palm_oil_n',
       'ingredients_that_may_be_from_palm_oil_n', 'nutrition_grade_fr',
       'pnns_groups_1', 'pnns_groups_2', 'states', 'states_tags',
       'states_en', 'main_category', 'main_category_en', 'image_url',
       'image_small_url', 'energy_100g', 'fat_100g', 'saturated-fat_100g',
       'trans-fat_100g', 'cholesterol_100g', 'carbohydrates_100g',
       'sugars_100g', 'fiber_100g', 'proteins_100g', 'salt_100g',
       'sodium_100g', 'vitamin-a_100g', 'vitamin-c_100g', 'calcium_100g',
       'iron_100g', 'nutrition-score-fr_100g', 'nutrition-score-uk_100g'],
      dtype=object)
```

```
[5]: off2['pnns_groups_1'].value_counts().head(10).to_frame()
```

```
[5]:                          pnns_groups_1
     unknown                          43603
     Sugary snacks                    14750
     Beverages                        13476
     Milk and dairy products          10733
     Cereals and potatoes             10078
     Fish Meat Eggs                    9473
     Composite foods                   7972
     Fat and sauces                    7122
     Fruits and vegetables             6763
     Salty snacks                      3299
```

```
[4]: ss = off2[np.logical_or(off2['pnns_groups_1']=='Sugary snacks',
                             off2['pnns_groups_1']=='Beverages')]
     print(ss.shape)
     ss.head(3)
```

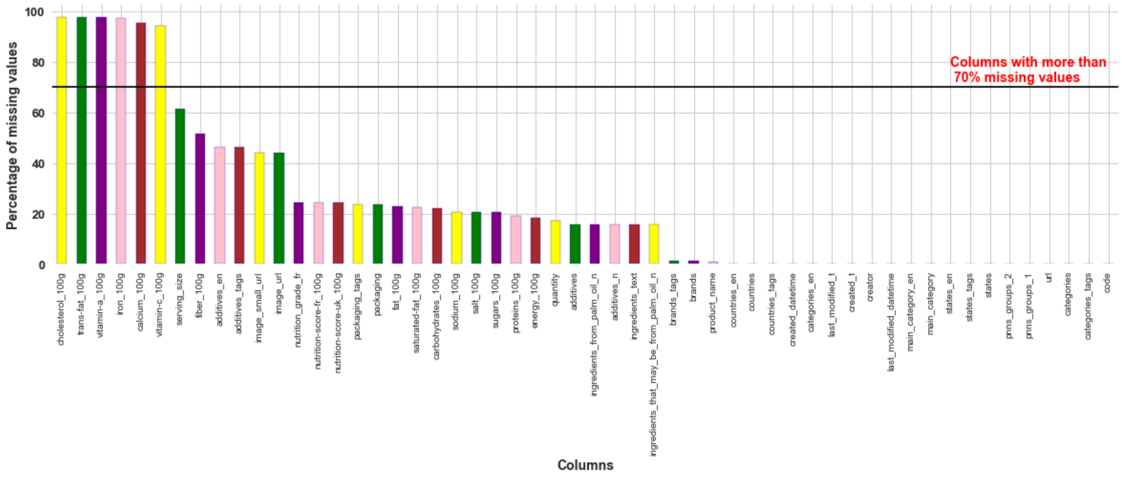
```
     (28226, 54)
```

```
[4]:         code                                                url   creator  \
     177   394710  http://world-en.openfoodfacts.org/product/0000...        b7
     182  1938067  http://world-en.openfoodfacts.org/product/0000...        b7
     185  7020254  http://world-en.openfoodfacts.org/product/0000...  teolemon

            created_t      created_datetime last_modified_t last_modified_datetime  \
     177  1484497370  2017-01-15T16:22:50Z      1484501040   2017-01-15T17:24:00Z
     182  1484501528  2017-01-15T17:32:08Z      1484504972   2017-01-15T18:29:32Z
     185  1420150193  2015-01-01T22:09:53Z      1504376301   2017-09-02T18:18:21Z

                           product_name  quantity      packaging packaging_tags  \
     177  Danoises à la cannelle roulées  1.150 kg          Frais          frais
     182    Chaussons tressés aux pommes  1.200 kg          Frais          frais
     185                       Root Beer   33 cl e  Canette,Métal  canette,metal

                      brands         brands_tags  \
     177  Kirkland Signature  kirkland-signature
     182  Kirkland Signature  kirkland-signature
     185                 A&W                 a-w
```

```
                                            categories  \
177      Snacks sucrés,Biscuits et gâteaux,Pâtisseries
182      Snacks sucrés,Biscuits et gâteaux,Pâtisseries
185  Boissons,Boissons gazeuses,Sodas,Boissons sucr…

                                       categories_tags  \
177  en:sugary-snacks,en:biscuits-and-cakes,en:past…
182  en:sugary-snacks,en:biscuits-and-cakes,en:past…
185  en:beverages,en:carbonated-drinks,en:sodas,en:…

                                         categories_en countries  \
177          Sugary snacks,Biscuits and cakes,Pastries    Canada
182          Sugary snacks,Biscuits and cakes,Pastries    Canada
185  Beverages,Carbonated drinks,Sodas,Sugared beve…    France

    countries_tags countries_en  \
177      en:canada       Canada
182      en:canada       Canada
185      en:france       France

                                      ingredients_text        serving_size  \
177  Ingrédients: Pâte (farine, eau, beurre, sucre,…   146 g / 1 danoise
182  Ingrédients : Pâte (farine, margarines d'huile…  150 g / 1 chausson
185  Eau gazéifiée, sirop de maïs à haute teneur en…               33 cl

     additives_n                                         additives  \
177         10.0   [ ingredients -> fr:ingredients  ]  [ pate ->…
182          5.0   [ ingredients -> fr:ingredients  ]  [ pate ->…
185          3.0   [ eau-gazeifiee -> fr:eau-gazeifiee  ]  [ eau…

                                        additives_tags  \
177  en:e1100,en:e170,en:e202,en:e203,en:e300,en:e3…
182            en:e202,en:e211,en:e330,en:e509,en:e920
185                            en:e150,en:e211,en:e999

                                          additives_en  \
177  E1100 - Alpha-Amylase,E170 - Calcium carbonate…
182  E202 - Potassium sorbate,E211 - Sodium benzoat…
185  E150 - Caramel,E211 - Sodium benzoate,E999 - Q…

     ingredients_from_palm_oil_n  ingredients_that_may_be_from_palm_oil_n  \
177                          0.0                                      1.0
182                          0.0                                      0.0
185                          0.0                                      0.0

    nutrition_grade_fr  pnns_groups_1        pnns_groups_2  \
177                NaN  Sugary snacks   Biscuits and cakes
182                  c  Sugary snacks   Biscuits and cakes
185                  e      Beverages  Sweetened beverages

                                                states  \
177  en:to-be-checked, en:complete, en:nutrition-fa…
182  en:to-be-checked, en:complete, en:nutrition-fa…
185  en:to-be-checked, en:complete, en:nutrition-fa…

                                           states_tags  \
177  en:to-be-checked,en:complete,en:nutrition-fact…
182  en:to-be-checked,en:complete,en:nutrition-fact…
185  en:to-be-checked,en:complete,en:nutrition-fact…

                                             states_en     main_category  \
177  To be checked,Complete,Nutrition facts complet…  en:sugary-snacks
182  To be checked,Complete,Nutrition facts complet…  en:sugary-snacks
185  To be checked,Complete,Nutrition facts complet…      en:beverages

    main_category_en                                          image_url  \
177    Sugary snacks                                                NaN
182    Sugary snacks                                                NaN
185        Beverages  http://en.openfoodfacts.org/images/products/00…

                                       image_small_url  energy_100g  fat_100g  \
177                                                NaN       1520.0      14.4
182                                                NaN       1090.0      10.7
185  http://en.openfoodfacts.org/images/products/00…        215.0       0.0

     saturated-fat_100g  trans-fat_100g  cholesterol_100g  carbohydrates_100g  \
177                 NaN             NaN           0.04110                54.1
182                 2.0           0.667           0.00533                38.7
185                 0.0             NaN               NaN                14.2

     sugars_100g  fiber_100g  proteins_100g  salt_100g  sodium_100g  \
177         28.1        2.05           4.79     0.9220       0.3630
182         24.7        2.00           3.33     0.6470       0.2550
185         13.6        0.00           0.00     0.0616       0.0242

     vitamin-a_100g  vitamin-c_100g  calcium_100g  iron_100g  \
177        0.000205         0.00616        0.0548    0.00247
182        0.000000         0.00160        0.0133    0.00048
185             NaN             NaN           NaN        NaN

     nutrition-score-fr_100g  nutrition-score-uk_100g
177                      NaN                      NaN
182                      9.0                      9.0
```

```
185                    18.0                    3.0
```

[5]:
```
plt.figure(figsize=(20, 5)); thresh_pc = 70

percentage=(ss.isnull().mean()) * 100
percentage.sort_values(ascending=False).plot.bar(color=('yellow', 'green',␣
↪'purple', 'pink', 'brown'),
                                                 edgecolor='b')
plt.axhline(y = thresh_pc, color='k', linestyle='-')
# plt.title('Missing values percentage per column', fontsize=20, weight='bold' )
plt.text(len(ss.isnull().sum()/len(off))/1.2, thresh_pc + 12.5,
         'Columns with more than \n %s%s missing values' %(thresh_pc, '%'),␣
↪fontsize=14, weight='bold', color='r',
          ha='left', va='top')
# plt.text(len(ss.isnull().sum()/len(off))/1.2, thresh_pc - 5,
#           'Columns with less than \n %s%s missing values' %(thresh_pc, '%'),␣
↪fontsize=14, weight='bold', color='g',
#           ha='left', va='top')
plt.xlabel('Columns', size=15, weight='bold', fontsize=14)
plt.ylabel('Percentage of missing values', weight='bold', fontsize=14)
plt.yticks(weight ='bold', fontsize = 13)

plt.show()
```



[5]:
```
ss2=ss.dropna(thresh=int(0.7*ss.shape[0]), axis=1)
print("Shape before cleaning = ", ss.shape)
print("Shape after cleaning  = ", ss2.shape)
print("We dropped ", ss.shape[1]- ss2.shape[1]," columns")
ss2.columns.values
```

```
Shape before cleaning =  (28226, 54)
Shape after cleaning  =  (28226, 42)
We dropped  12  columns
```

[5]:

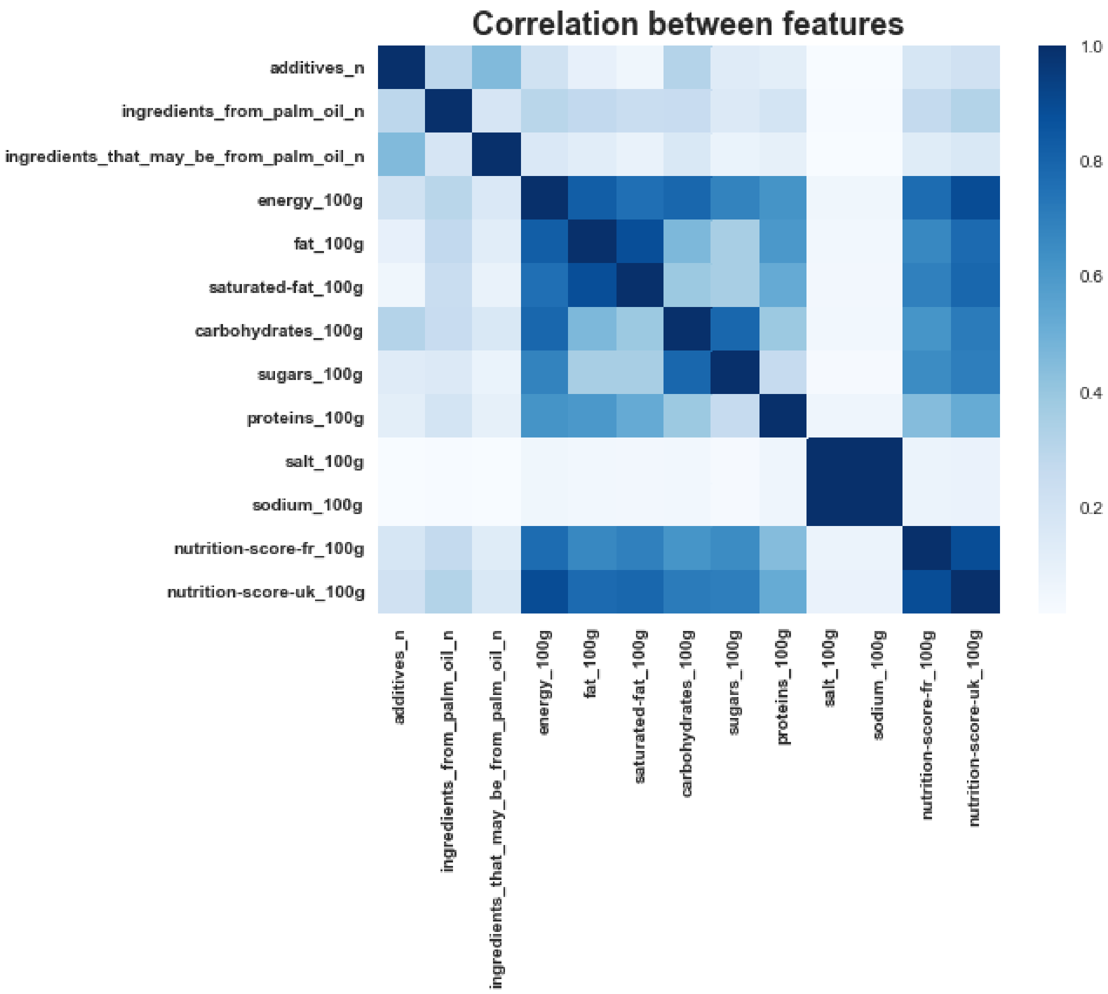
```
array(['code', 'url', 'creator', 'created_t', 'created_datetime',
       'last_modified_t', 'last_modified_datetime', 'product_name',
       'quantity', 'packaging', 'packaging_tags', 'brands', 'brands_tags',
       'categories', 'categories_tags', 'categories_en', 'countries',
       'countries_tags', 'countries_en', 'ingredients_text',
       'additives_n', 'additives', 'ingredients_from_palm_oil_n',
       'ingredients_that_may_be_from_palm_oil_n', 'nutrition_grade_fr',
       'pnns_groups_1', 'pnns_groups_2', 'states', 'states_tags',
       'states_en', 'main_category', 'main_category_en', 'energy_100g',
       'fat_100g', 'saturated-fat_100g', 'carbohydrates_100g',
       'sugars_100g', 'proteins_100g', 'salt_100g', 'sodium_100g',
       'nutrition-score-fr_100g', 'nutrition-score-uk_100g'], dtype=object)
```

[25]:

```python
ss2.to_csv('ss2_forLeon.csv')
```

[6]:

```python
ss2 = ss2.fillna(0, axis=1)

ss_corr=ss2.corr()
f,ax=plt.subplots(figsize=(8,6))
sns.heatmap(ss_corr, cmap='Blues')
plt.title("Correlation between features",
          weight='bold',
          fontsize=18)
plt.xticks(weight='bold')
plt.yticks(weight='bold')

plt.show()
```



[7]:

```
ss2 = ss.dropna(thresh=int(0.2*ss.shape[0]), axis=1)
print("Shape before cleaning = ", ss.shape)
print("Shape after cleaning  = ", ss2.shape)
print("We dropped ", ss.shape[1]- ss2.shape[1]," columns")
```

```
Shape before cleaning =  (28226, 54)
Shape after cleaning  =  (28226, 48)
We dropped  6  columns
```

[10]:

```
ss2['main_category'].value_counts().head(20).to_frame()
```

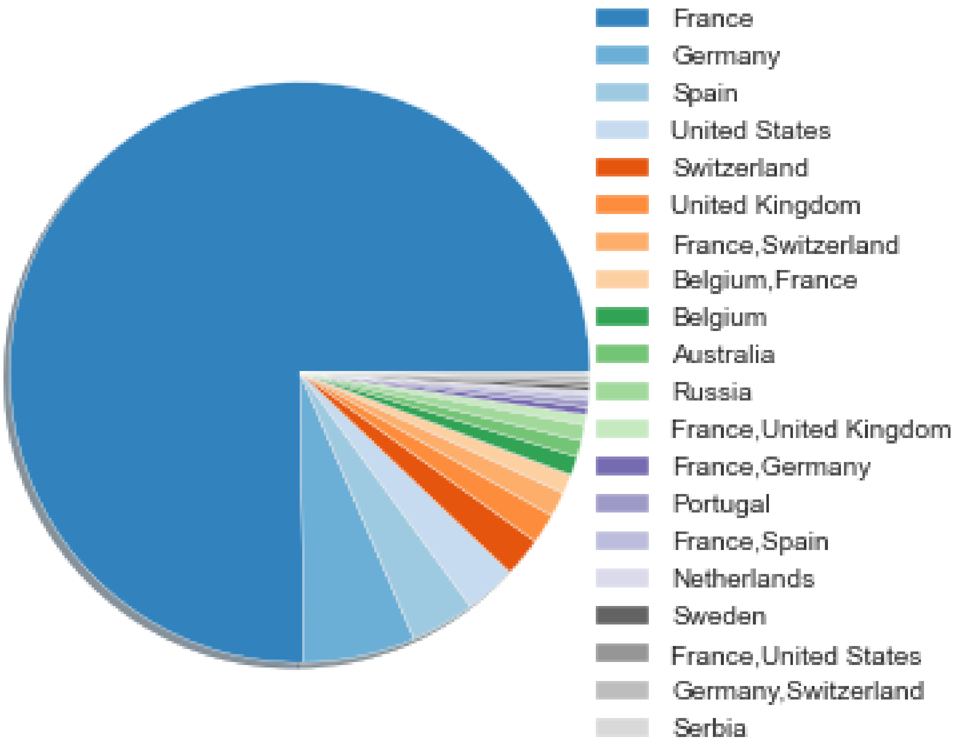
[10]:

```
                                     main_category
en:sugary-snacks                             11195
en:beverages                                 10587
en:fruit-juices                               2383
en:plant-based-foods-and-beverages            1548
en:spreads                                    1188
```

```
en:fruit-juices-and-nectars                   436
en:sweeteners                                 408
en:breakfasts                                 250
en:groceries                                   98
en:desserts                                    64
en:fresh-foods                                 40
es:pan-y-reposteria                            28
en:dairies                                      1
```

[57]:

```
wordcloud1 = WordCloud(width=600, height=500, background_color='white').
 ↪generate(' '.join(ss2['main_category']))
WordCloud.generate_from_frequencies

fig, ax2 = plt.subplots(1,1, figsize=(6,5))

# ax1.set_title('Sugary snacks and beverages', weight='bold', fontsize=15,␣
 ↪color='k')
# im1 = ax1.imshow(wordcloud1, aspect='auto')

pie = ax2.pie(np.ravel(ss2['countries_en'].value_counts().head(20).to_frame().
 ↪values),
        shadow=True, startangle=0, colors = plt.cm.tab20c.colors)
labels=np.array(ss2['countries_en'].value_counts().head(20).to_frame().index)
ax2.legend(pie[0], labels, bbox_to_anchor=(1.1,0.5), loc="center right",␣
 ↪fontsize=10,
           bbox_transform=plt.gcf().transFigure)
plt.show()
```



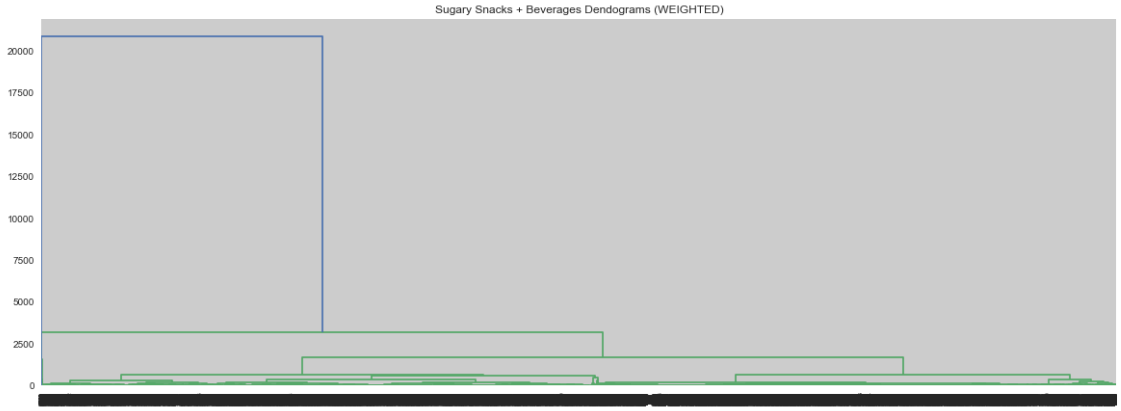
[8]:

```
from scipy.stats import skew
from sklearn.preprocessing import RobustScaler

ss2_sub = ss2[['energy_100g','fat_100g',
               'saturated-fat_100g', 'carbohydrates_100g',
               'sugars_100g', 'proteins_100g', 'salt_100g',
               'sodium_100g']]
ss2_sub = ss2_sub.dropna(axis=0).reset_index(); del ss2_sub['index']
numfeats = ss2_sub.dtypes[ss2_sub.dtypes != "object"].index

skewfeats = ss2_sub[numfeats].apply(lambda x: skew(x.dropna())) #compute␣
 ↪skewness
skewfeats = skewfeats[skewfeats > 0.75]
skewfeats = skewfeats.index

ss2_sub[skewfeats] = np.log1p(ss2_sub[skewfeats])
# scaler=RobustScaler()
# scaler.fit(ss2_sub)
```

[16]:

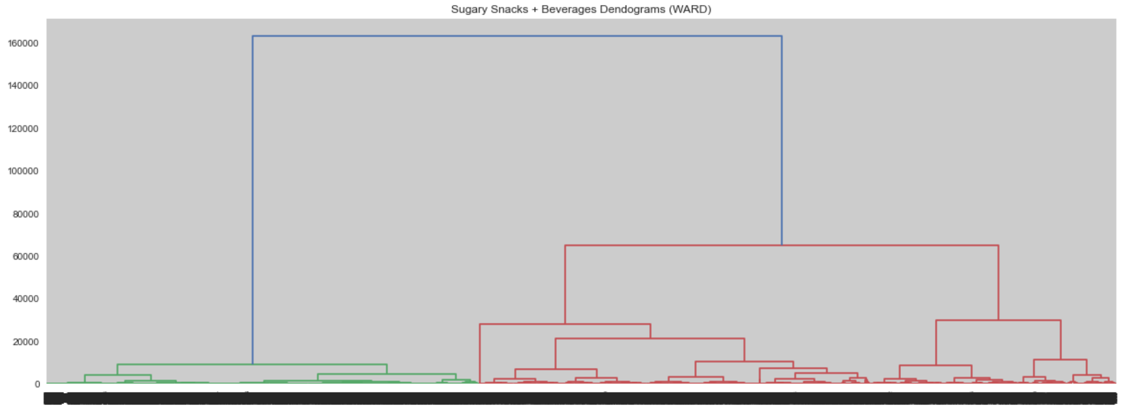
```
import sys
import scipy.cluster.hierarchy as shc
from sklearn.cluster import KMeans, AgglomerativeClustering
```

```
ss2_sub = ss2_sub.fillna(0, axis=1)
plt.figure(figsize=(20, 7))
plt.title("Sugary Snacks + Beverages Dendograms (WEIGHTED)", fontsize=12)
plt.xticks(rotation=90)

sys.setrecursionlimit(200000)
dend = shc.dendrogram(shc.linkage(ss2_sub, method='average'))
```



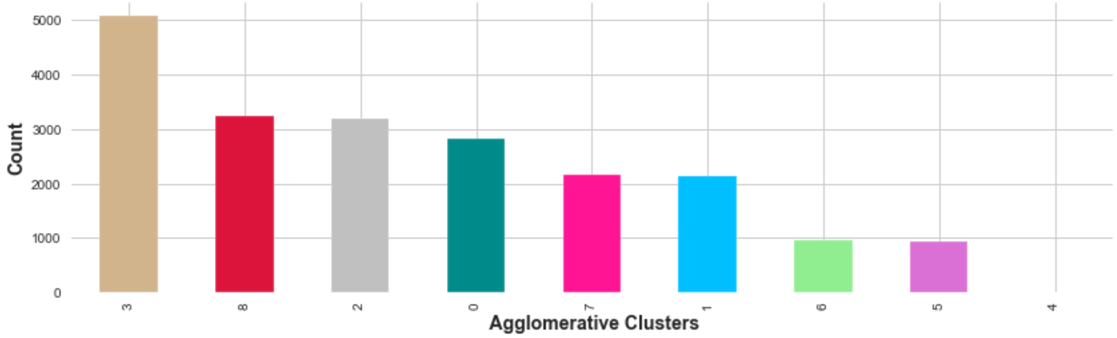
[12]:

```
import sys
import scipy.cluster.hierarchy as shc
from sklearn.cluster import KMeans, AgglomerativeClustering

ss2_sub = ss2_sub.fillna(0, axis=1)
plt.figure(figsize=(20, 7))
plt.title("Sugary Snacks + Beverages Dendograms (WARD)", fontsize=12)
plt.xticks(rotation=90)

sys.setrecursionlimit(100000)
dend = shc.dendrogram(shc.linkage(ss2_sub, method='ward'))
```



[14]:

```
from sklearn.cluster import KMeans, AgglomerativeClustering
aggclust = AgglomerativeClustering(n_clusters = 9, affinity='euclidean',␣
 ↪memory=None,
                              connectivity=None, compute_full_tree='auto',␣
 ↪linkage='ward')
pred_agg = aggclust.fit_predict(ss2_sub)
ss2_sub['agglom_clust'] = pred_agg

plt.figure(figsize=(14,4))
ss2_sub['agglom_clust'].value_counts().plot(kind='bar', color=['tan',␣
 ↪'crimson', 'silver', 'darkcyan',
                                                         'deeppink',␣
 ↪'deepskyblue','lightgreen', 'orchid'])
plt.ylabel("Count",fontsize=14, weight='bold')
plt.xlabel(' Agglomerative Clusters', fontsize=14, weight='bold')
plt.show()
```



[63]:

```
from sklearn.cluster import KMeans, AgglomerativeClustering, SpectralClustering
import numpy as np

kmeans = KMeans(n_clusters = 9, init = 'k-means++', n_init = 10,
                tol = 0.0001, n_jobs = -1, random_state = 1).fit(ss2_sub)
labels2 = kmeans.labels_

# spec = SpectralClustering(n_clusters=9, n_init = 10,
#                  eigen_tol = 0.0001, assign_labels='kmeans', random_state = 1).
 ↪fit(ss2_sub)
# labels3 = spec.labels_

centers=kmeans.cluster_centers_
pred_km = kmeans.fit_predict(ss2_sub)
```

```
# pred_sp = spec.fit_predict(ss2_sub)
ss2_sub['kmeans_clust'] = pred_km
# ss2_sub['spec_clust'] = pred_sp
```

```
C:\Users\arkam\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:939:
FutureWarning: 'n_jobs' was deprecated in version 0.23 and will be removed in
0.25.
  " removed in 0.25.", FutureWarning)
C:\Users\arkam\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:939:
FutureWarning: 'n_jobs' was deprecated in version 0.23 and will be removed in
0.25.
  " removed in 0.25.", FutureWarning)
```

[21]:

```
ss2_nona = ss2[['energy_100g','fat_100g',
               'saturated-fat_100g', 'carbohydrates_100g',
               'sugars_100g', 'nutrition-score-fr_100g',
               'nutrition-score-uk_100g', 'pnns_groups_2', 'main_category']]
ss2_nona = ss2_nona.dropna(axis=0).reset_index()
ss2_nona['agglom_clust'] = aggclust.fit_predict(ss2_sub)
ss2_nona['kmeans_clust'] = kmeans.fit_predict(ss2_sub)
ss2_nona['spec_clust'] = spec.fit_predict(ss2_sub)
ss2_nona.head(5)
```

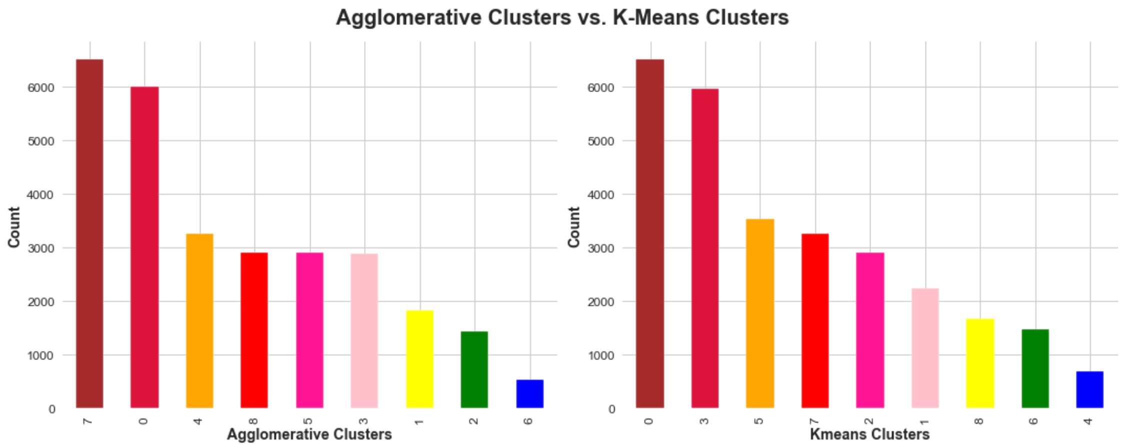
```
C:\Users\arkam\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:939:
FutureWarning: 'n_jobs' was deprecated in version 0.23 and will be removed in
0.25.
  " removed in 0.25.", FutureWarning)
```

[21]:

```
   index  energy_100g  fat_100g  saturated-fat_100g  carbohydrates_100g  \
0    177       1520.0      14.4                 0.0                54.1
1    182       1090.0      10.7                 2.0                38.7
2    185        215.0       0.0                 0.0                14.2
3    186          0.0       0.0                 0.0                 0.0
4    189       1667.0       0.0                 0.0                93.3

   sugars_100g  nutrition-score-fr_100g  nutrition-score-uk_100g  \
0         28.1                      0.0                      0.0
1         24.7                      9.0                      9.0
2         13.6                     18.0                      3.0
3          0.0                      0.0                      0.0
4         93.3                     14.0                     14.0

         pnns_groups_2     main_category  agglom_clust  kmeans_clust
0   Biscuits and cakes  en:sugary-snacks             5             2
1   Biscuits and cakes  en:sugary-snacks             2             6
2  Sweetened beverages      en:beverages             0             3
3   Biscuits and cakes  en:sugary-snacks             7             0
4               Sweets  en:sugary-snacks             1             4
```

[22]:

```
fig = plt.figure(figsize=(15,6))

fig.add_subplot(1,2,1)
fig.suptitle("Agglomerative Clusters vs. K-Means Clusters", fontsize=20,␣
 ↪weight='bold')

ss2_nona['agglom_clust'].value_counts().plot(kind='bar', color=['brown',␣
 ↪'crimson', 'orange', 'red', 'deeppink', 'pink', 'yellow', 'green', 'blue'])
plt.ylabel("Count",fontsize=14, weight='bold')
plt.xticks(fontsize=12); plt.yticks(fontsize=12)
plt.xlabel('Agglomerative Clusters', fontsize=14, weight='bold')

fig.add_subplot(1,2,2)
ss2_nona['kmeans_clust'].value_counts().plot(kind='bar', color=['brown',␣
 ↪'crimson', 'orange', 'red', 'deeppink', 'pink', 'yellow', 'green', 'blue'])
plt.ylabel("Count",fontsize=14, weight='bold')
plt.xlabel('Kmeans Clusters', fontsize=14, weight='bold')
plt.xticks(fontsize=12); plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```



[25]:

```
#Clusters column
agclust0=ss2_nona[ss2_nona['agglom_clust']==7]
agclust1=ss2_nona[ss2_nona['agglom_clust']==0]
agclust2=ss2_nona[ss2_nona['agglom_clust']==4]
agclust3=ss2_nona[ss2_nona['agglom_clust']==8]
agclust4=ss2_nona[ss2_nona['agglom_clust']==5]
agclust5=ss2_nona[ss2_nona['agglom_clust']==3]
agclust6=ss2_nona[ss2_nona['agglom_clust']==1]
agclust7=ss2_nona[ss2_nona['agglom_clust']==2]
agclust8=ss2_nona[ss2_nona['agglom_clust']==6]
```

```

agclust7.head(2)
```

[25]:
```
    index  energy_100g  fat_100g  saturated-fat_100g  carbohydrates_100g  \
1     182       1090.0      10.7                 2.0                38.7
18    370        586.0       0.0                 0.0                34.0

    sugars_100g  nutrition-score-fr_100g  nutrition-score-uk_100g  \
1          24.7                      9.0                      9.0
18         24.0                      6.0                      6.0

         pnns_groups_2     main_category  agglom_clust  kmeans_clust
1   Biscuits and cakes  en:sugary-snacks             2             6
18              Sweets  en:sugary-snacks             2             6
```

[27]:
```
#Clusters column
kmclust0=ss2_nona[ss2_nona['kmeans_clust']==5]
kmclust1=ss2_nona[ss2_nona['kmeans_clust']==3]
kmclust2=ss2_nona[ss2_nona['kmeans_clust']==6]
kmclust3=ss2_nona[ss2_nona['kmeans_clust']==0]
kmclust4=ss2_nona[ss2_nona['kmeans_clust']==8]
kmclust5=ss2_nona[ss2_nona['kmeans_clust']==7]
kmclust6=ss2_nona[ss2_nona['kmeans_clust']==1]
kmclust7=ss2_nona[ss2_nona['kmeans_clust']==2]
kmclust8=ss2_nona[ss2_nona['kmeans_clust']==4]

kmclust7.head(2)
```

[27]:
```
   index  energy_100g  fat_100g  saturated-fat_100g  carbohydrates_100g  \
0    177       1520.0      14.4                 0.0                54.1
7    223       2257.0      33.3                21.1                53.8

   sugars_100g  nutrition-score-fr_100g  nutrition-score-uk_100g  \
0         28.1                      0.0                      0.0
7         51.5                     26.0                     26.0

        pnns_groups_2                     main_category  agglom_clust  \
0  Biscuits and cakes                  en:sugary-snacks             5
7              Sweets  en:plant-based-foods-and-beverages             5

   kmeans_clust
0             2
7             2
```

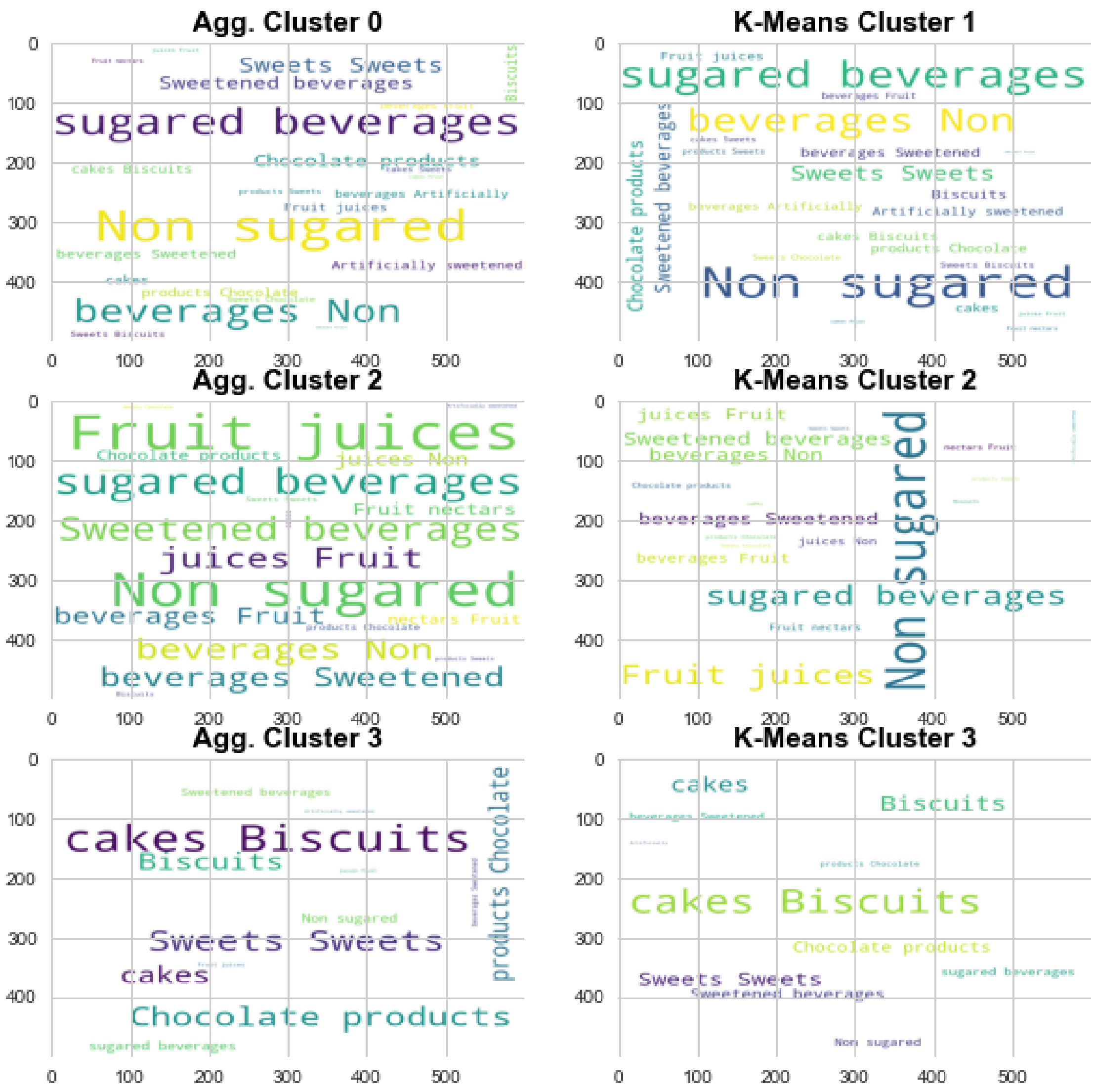
[55]:
```
# print(agclust0['pnns_groups_2'].value_counts().head(2))
# print(agclust1['pnns_groups_2'].value_counts().head(2))
# print(agclust2['pnns_groups_2'].value_counts().head(2))
# print(agclust3['pnns_groups_2'].value_counts().head(2))
# print(agclust4['pnns_groups_2'].value_counts().head(2))
# print(agclust5['pnns_groups_2'].value_counts().head(2))
# print(agclust6['pnns_groups_2'].value_counts().head(2))
# print(agclust7['pnns_groups_2'].value_counts().head(2))
# print(agclust1['pnns_groups_2'].value_counts().head(2))

# print(kmclust0['pnns_groups_2'].value_counts().head(2))
# print(kmclust1['pnns_groups_2'].value_counts().head(2))
# print(kmclust2['pnns_groups_2'].value_counts().head(2))
# print(kmclust3['pnns_groups_2'].value_counts().head(2))
# print(kmclust4['pnns_groups_2'].value_counts().head(2))
# print(kmclust5['pnns_groups_2'].value_counts().head(2))
# print(kmclust6['pnns_groups_2'].value_counts().head(2))
# print(kmclust7['pnns_groups_2'].value_counts().head(2))
# print(kmclust8['pnns_groups_2'].value_counts().head(2))
# np.mean(agclust0['nutrition-score-fr_100g'].values[np.ravel(np.
 ↪where(agclust0['nutrition-score-fr_100g']>0))])
# np.mean(agclust1['nutrition-score-fr_100g'].values[np.ravel(np.
 ↪where(agclust1['nutrition-score-fr_100g']>0))])
# np.mean(agclust2['nutrition-score-fr_100g'].values[np.ravel(np.
 ↪where(agclust2['nutrition-score-fr_100g']>0))])
# np.mean(agclust3['nutrition-score-fr_100g'].values[np.ravel(np.
 ↪where(agclust3['nutrition-score-fr_100g']>0))])
# np.mean(agclust4['nutrition-score-fr_100g'].values[np.ravel(np.
 ↪where(agclust4['nutrition-score-fr_100g']>0))])
# np.mean(agclust5['nutrition-score-fr_100g'].values[np.ravel(np.
 ↪where(agclust5['nutrition-score-fr_100g']>0))])
# np.mean(agclust6['nutrition-score-fr_100g'].values[np.ravel(np.
 ↪where(agclust6['nutrition-score-fr_100g']>0))])
# np.mean(agclust7['nutrition-score-fr_100g'].values[np.ravel(np.
 ↪where(agclust7['nutrition-score-fr_100g']>0))])
# np.mean(agclust8['nutrition-score-fr_100g'].values[np.ravel(np.
 ↪where(agclust8['nutrition-score-fr_100g']>0))])
```

```
[55]: 16.05093996361431
```

[26]:

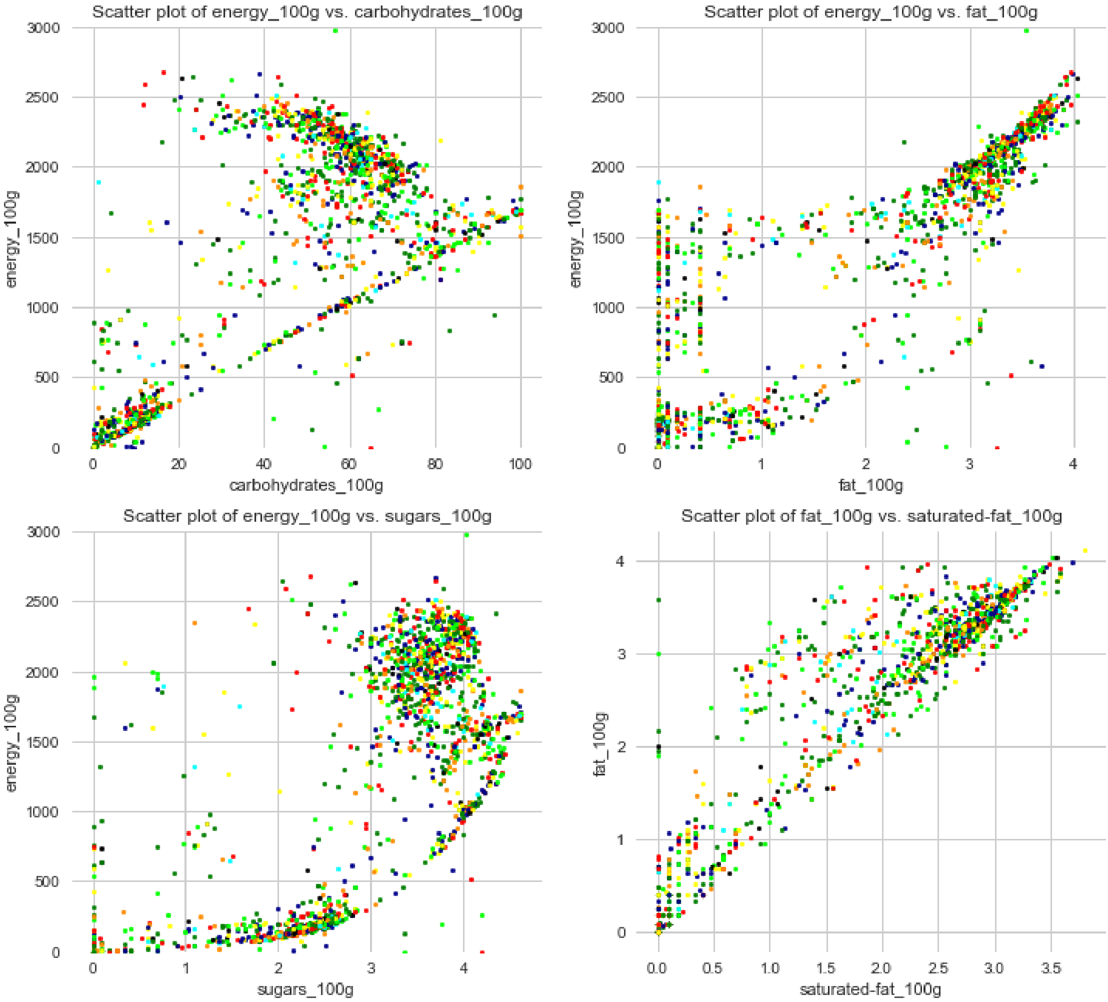
```
fig, axes = plt.subplots(3,2, figsize=(10,10))

ax1 = axes[0,0]
ax1.set_title('Agg. Cluster 0\1', weight='bold', fontsize=15, color='k')
wordcloud1 = WordCloud(width=600, height=500, background_color='white').
 ↪generate(' '.join(agclust0['pnns_groups_2']))
WordCloud.generate_from_frequencies
im1 = ax1.imshow(wordcloud1, aspect='auto')
```

```
ax2 = axes[0,1]
ax2.set_title('K-Means Cluster 1', weight='bold', fontsize=15, color='k')
wordcloud1 = WordCloud(width=600, height=500, background_color='white').
 ↪generate(' '.join(kmclust0['pnns_groups_2']))
WordCloud.generate_from_frequencies
im2 = ax2.imshow(wordcloud1, aspect='auto')

ax3 = axes[1,0]
ax3.set_title('Agg. Cluster 2', weight='bold', fontsize=15, color='k')
wordcloud1 = WordCloud(width=600, height=500, background_color='white').
 ↪generate(' '.join(agclust1['pnns_groups_2']))
WordCloud.generate_from_frequencies
im3 = ax3.imshow(wordcloud1, aspect='auto')

ax4 = axes[1,1]
ax4.set_title('K-Means Cluster 2', weight='bold', fontsize=15, color='k')
wordcloud1 = WordCloud(width=600, height=500, background_color='white').
 ↪generate(' '.join(kmclust1['pnns_groups_2']))
WordCloud.generate_from_frequencies
im4 = ax4.imshow(wordcloud1, aspect='auto')

ax3 = axes[2,0]
ax3.set_title('Agg. Cluster 3', weight='bold', fontsize=15, color='k')
wordcloud1 = WordCloud(width=600, height=500, background_color='white').
 ↪generate(' '.join(agclust2['pnns_groups_2']))
WordCloud.generate_from_frequencies
im3 = ax3.imshow(wordcloud1, aspect='auto')

ax4 = axes[2,1]
ax4.set_title('K-Means Cluster 3', weight='bold', fontsize=15, color='k')
wordcloud1 = WordCloud(width=600, height=500, background_color='white').
 ↪generate(' '.join(kmclust2['pnns_groups_2']))
WordCloud.generate_from_frequencies
im4 = ax4.imshow(wordcloud1, aspect='auto')
```

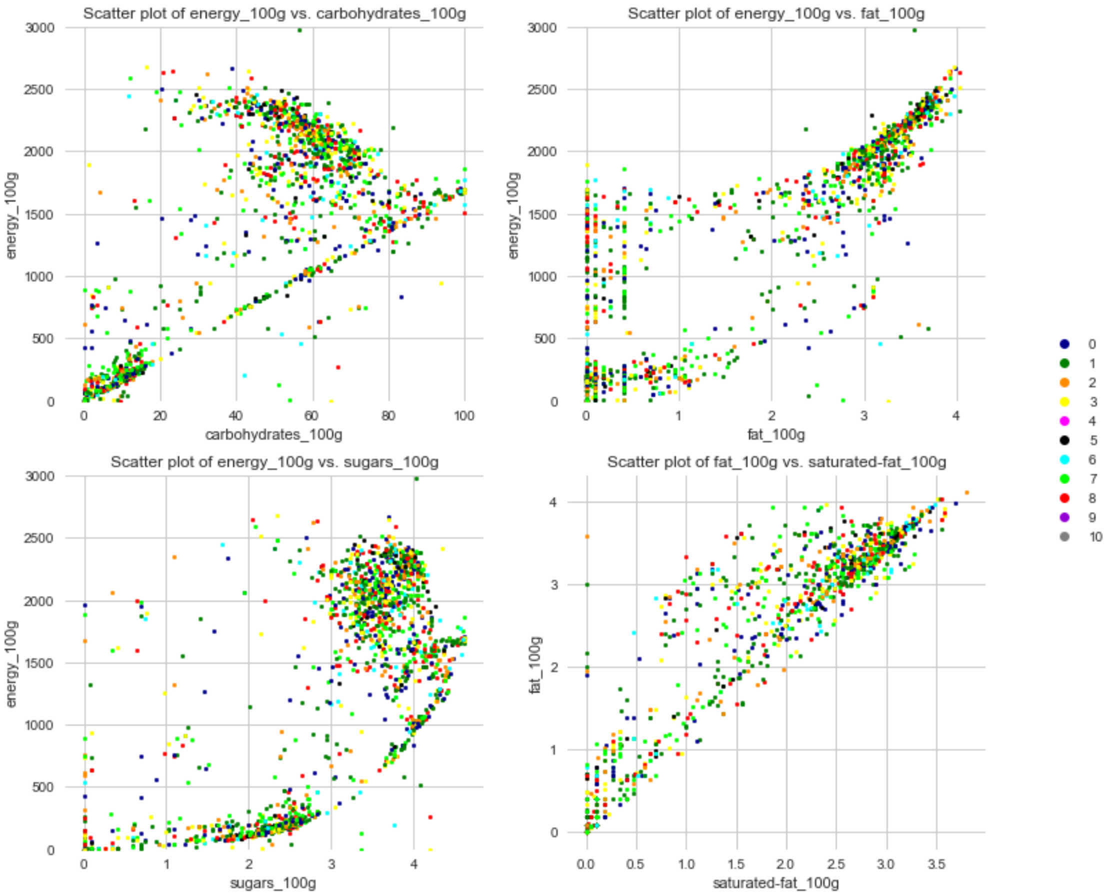


```
[337]: def plot_km_cluster(data, color):
           fig, ax = plt.subplots(2, 2, figsize=(12,11)) # define plot area
           x_cols = ['carbohydrates_100g', 'fat_100g', 'sugars_100g',␣
        ↪'saturated-fat_100g']
           y_cols = ['energy_100g', 'energy_100g', 'energy_100g', 'fat_100g']
           for x_col,y_col,i,j in zip(x_cols,y_cols,[0,0,1,1],[0,1,0,1]):
               for x,y,c in zip(data[x_col], data[y_col], color_km):
                   ax[i,j].scatter(x,y, color = c, s=8)
               ax[i,j].set_title('Scatter plot of ' + y_col + ' vs. ' + x_col) # Give␣
        ↪the plot a main title
               ax[i,j].set_xlabel(x_col) # Set text for the x axis
               ax[i,j].set_ylabel(y_col)# Set text for y axis
               if np.logical_and(i==1, j==1) == False:
```

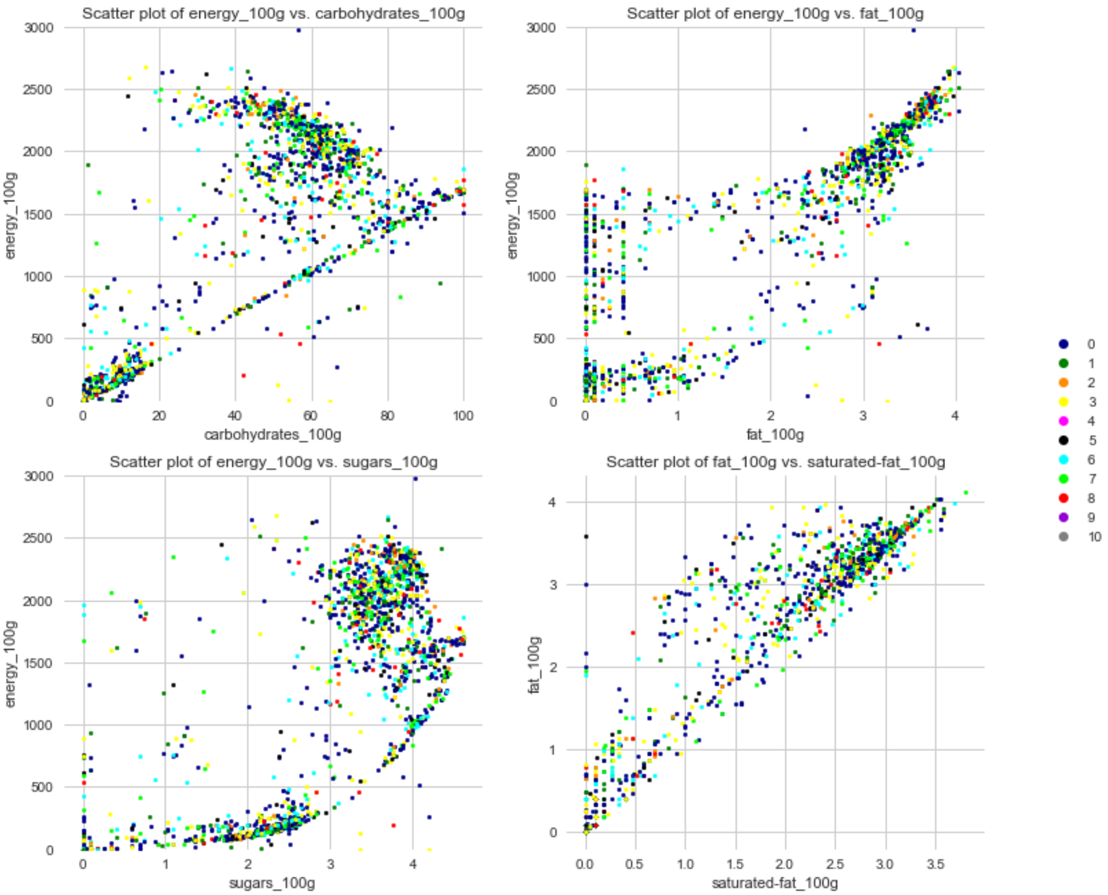
```
            ax[i,j].set_ylim(0,3000)
    plt.show()

#Create color dictionary for clusters
col_dic = {0:'darkblue',1:'green',2:'darkorange',3:'yellow',
           4:'magenta',5:'black', 6:'cyan', 7:'lime', 8:'red', 9:'darkviolet',␣
 ↪10:'grey'}
# kpred = kmeans.fit_predict(ss2_sub)
ind = (np.random.uniform(0,1,2000)*len(kpred)).astype(int)
color_km = [col_dic[x] for x in kpred[ind]]
plot_km_cluster(ss2_nona, color_km)
```



[350]:
```
def plot_km_cluster(data, color):
    fig, ax = plt.subplots(2, 2, figsize=(12,11)) # define plot area
    x_cols = ['carbohydrates_100g', 'fat_100g', 'sugars_100g',␣
 ↪'saturated-fat_100g']
    y_cols = ['energy_100g', 'energy_100g', 'energy_100g', 'fat_100g']
    for x_col,y_col,i,j in zip(x_cols,y_cols,[0,0,1,1],[0,1,0,1]):
        for x,y,c in zip(data[x_col], data[y_col], color_km):
            ax[i,j].scatter(x,y, color = c, s=8)
        ax[i,j].set_title('Scatter plot of ' + y_col + ' vs. ' + x_col) # Give␣
 ↪the plot a main title
        ax[i,j].set_xlabel(x_col) # Set text for the x axis
        ax[i,j].set_ylabel(y_col)# Set text for y axis
        if np.logical_and(i==1, j==1) == False:
            ax[i,j].set_ylim(0,3000)
    markers = [plt.Line2D([0,0],[0,0],color=color, marker='o', linestyle='')␣
 ↪for color in col_dic.values()]
    plt.legend(markers, col_dic.keys(), numpoints=1, bbox_to_anchor=(1.01,0.5),␣
 ↪loc="center right", fontsize=10,
           bbox_transform=plt.gcf().transFigure)
    plt.show()

#Create color dictionary for clusters
col_dic = {0:'darkblue',1:'green',2:'darkorange',3:'yellow',
           4:'magenta',5:'black', 6:'cyan', 7:'lime', 8:'red', 9:'darkviolet',␣
 ↪10:'grey'}
# kpred = kmeans.fit_predict(ss2_sub)
ind = (np.random.uniform(0,1,2000)*len(kpred)).astype(int)
color_km = [col_dic[x] for x in kpred[ind]]
plot_km_cluster(ss2_nona, color_km)
```



```
[354]: def plot_ag_cluster(data, color):
           fig, ax = plt.subplots(2, 2, figsize=(12,11)) # define plot area
           x_cols = ['carbohydrates_100g', 'fat_100g', 'sugars_100g',␣
        ↪'saturated-fat_100g']
           y_cols = ['energy_100g', 'energy_100g', 'energy_100g', 'fat_100g']
           for x_col,y_col,i,j in zip(x_cols,y_cols,[0,0,1,1],[0,1,0,1]):
               for x,y,c in zip(data[x_col], data[y_col], color):
                   ax[i,j].scatter(x,y, color = c, s=8)
               ax[i,j].set_title('Scatter plot of ' + y_col + ' vs. ' + x_col) # Give␣
        ↪the plot a main title
               ax[i,j].set_xlabel(x_col) # Set text for the x axis
               ax[i,j].set_ylabel(y_col)# Set text for y axis
               if np.logical_and(i==1, j==1) == False:
                   ax[i,j].set_ylim(0,3000)
           markers = [plt.Line2D([0,0],[0,0],color=color, marker='o', linestyle='')␣
        ↪for color in col_dic.values()]
           plt.legend(markers, col_dic.keys(), numpoints=1, bbox_to_anchor=(1.01,0.5),␣
        ↪loc="center right", fontsize=10,
           bbox_transform=plt.gcf().transFigure)
    plt.show()

# pred_agg = aggclust.fit_predict(ss2_sub)
color_ag = [col_dic[x] for x in pred_agg[ind]]
plot_ag_cluster(ss2_nona, color_ag)
```



[ ]: