Classification automatique sous Python

CAH et K-Means

Ricco.Rakotomalala
http://eric.univ-lyon2.fr/~ricco/cours

Importation des données, description

DONNÉES

Objectif de l'étude

Classification automatique de fromages

Objectifs de l'étude

Ce document retranscrit une démarche de classification automatique d'un ensemble de fromages (29 observations) décrits par leurs propriétés nutritives (ex. protéines, lipides, etc. ; 9 variables). L'objectif est d'identifier des groupes de fromages homogènes, partageant des caractéristiques similaires.

Nous utiliserons essentiellement deux approches en nous appuyant sur deux procédures des packages spécialisés pour **Python** : la classification ascendante hiérarchique (CAH – Package SciPy) ; la méthode des centres mobiles (k-Means – Package Scikit-Learn).

Le fichier « fromage.txt » provient de la page de cours de Marie Chavent de l'Université de Bordeaux. Les excellents supports et exercices corrigés que l'on peut y trouver compléteront à profit ce tutoriel qui se veut avant tout un guide simple pour une première prise en main de Python dans le contexte de la classification automatique.

Traitements réalisés

- Chargement et description des données
- Classification automatique
- Pistes pour la détection du nombre adéquat de classes
- Description – interprétation des groupes

Données disponibles

Fromages	calories	sodium	calcium	lipides	retinol	folates	proteines	cholesterol	magnesium
CarredelEst	314	353.5	72.6	26.3	51.6	30.3	21	70	20
Babybel	314	238	209.8	25.1	63.7	6.4	22.6	70	27
Beaufort	401	112	259.4	33.3	54.9	1.2	26.6	120	41
Bleu	342	336	211.1	28.9	37.1	27.5	20.2	90	27
Camembert	264	314	215.9	19.5	103	36.4	23.4	60	20
Cantal	367	256	264	28.8	48.8	5.7	23	90	30
Chabichou	344	192	87.2	27.9	90.1	36.3	19.5	80	36
Chaource	292	276	132.9	25.4	116.4	32.5	17.8	70	25
Cheddar	406	172	182.3	32.5	76.4	4.9	26	110	28
Comte	399	92	220.5	32.4	55.9	1.3	29.2	120	51
Coulomniers	308	222	79.2	25.6	63.6	21.1	20.5	80	13
Edam	327	148	272.2	24.7	65.7	5.5	24.7	80	44
Emmental	378	60	308.2	29.4	56.3	2.4	29.4	110	45
Fr.chevrepatemolle	206	160	72.8	18.5	150.5	31	11.1	50	16
Fr.fondu.45	292	390	168.5	24	77.4	5.5	16.8	70	20
Fr.frais20nat.	80	41	146.3	3.5	50	20	8.3	10	11
Fr.frais40nat.	115	25	94.8	7.8	64.3	22.6	7	30	10
Maroilles	338	311	236.7	29.1	46.7	3.6	20.4	90	40
Morbier	347	285	219	29.5	57.6	5.8	23.6	80	30
Parmesan	381	240	334.6	27.5	90	5.2	35.7	80	46
Petitsuisse40	142	22	78.2	10.4	63.4	20.4	9.4	20	10
PontlEveque	300	223	156.7	23.4	53	4	21.1	70	22
Pyrenees	355	232	178.9	28	51.5	6.8	22.4	90	25
Reblochon	309	272	202.3	24.6	73.1	8.1	19.7	80	30
Rocquefort	370	432	162	31.2	83.5	13.3	18.7	100	25
SaintPaulin	298	205	261	23.3	60.4	6.7	23.3	70	26
Tome	321	252	125.5	27.3	62.3	6.2	21.8	80	20
Vacherin	321	140	218	29.3	49.2	3.7	17.6	80	30
Yaourtlaitent.nat.	70	91	215.7	3.4	42.9	2.9	4.1	13	14

Label des observations

Variables actives

Fichier de données

Importation, statistiques descriptives et graphiques

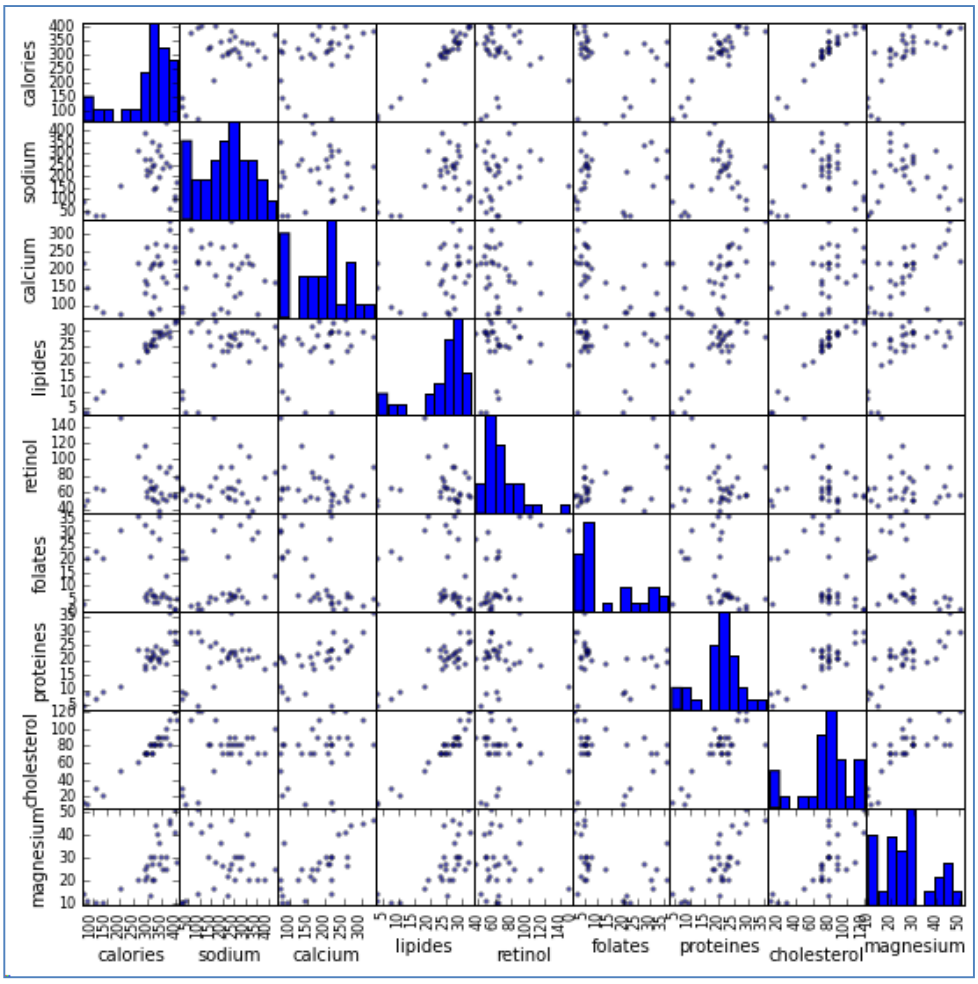
```
#modification du dossier par défaut
import os
os.chdir("…")

#importation des données
import pandas
fromage = pandas.read_table("fromage.txt",sep="\t",header=0,index_col=0)

#dimension des données
print(fromage.shape)

#statistiques descriptives
print(fromage.describe())

#graphique - croisement deux à deux des variables
from pandas.tools.plotting import scatter_matrix
scatter_matrix(fromage,figsize=(9,9))
```



Ce type de graphique n'est jamais anodin. Nous constatons par exemple que (1) « lipides » est fortement corrélé avec « calories » et « cholestérol » (sans trop de surprises) (remarque : la même information va peser 3 fois dans l'analyse) ; (2) dans certaines configurations, des groupes semblent apparaître naturellement (ex. croisement de « protéines » et « cholestérol », avec une corrélation inter-groupes assez marquée).

Classification ascendante hiérarchique

CAH

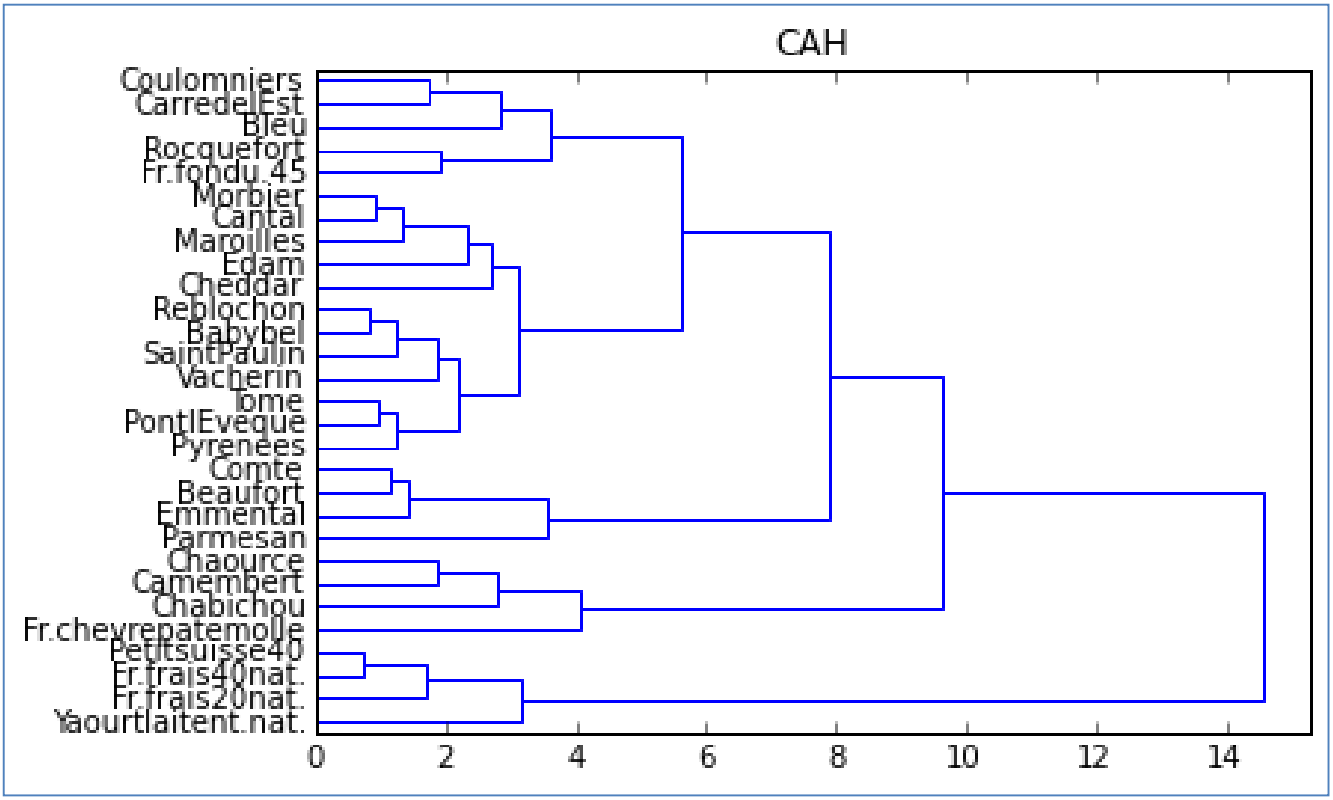
Classification ascendante hiérarchique

Utilisation du package « **scipy** »

```
#librairies pour la CAH
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage

#générer la matrice des liens
Z = linkage(fromage_cr,method='ward',metric='euclidean')

#affichage du dendrogramme
plt.title("CAH")
dendrogram(Z,labels=fromage.index,orientation='left',color_threshold=0)
plt.show()
```



Le dendrogramme « suggère » un découpage en 4 groupes. On note qu'une classe de fromages, les « fromages frais » (tout à gauche), se démarque fortement des autres au point qu'on aurait pu envisager aussi un découpage en 2 groupes seulement. Nous y reviendrons plus longuement lorsque nous mixerons l'analyse avec une analyse en composantes principales (ACP).

Classification ascendante hiérarchique

Découpage en classes – Matérialisation des groupes

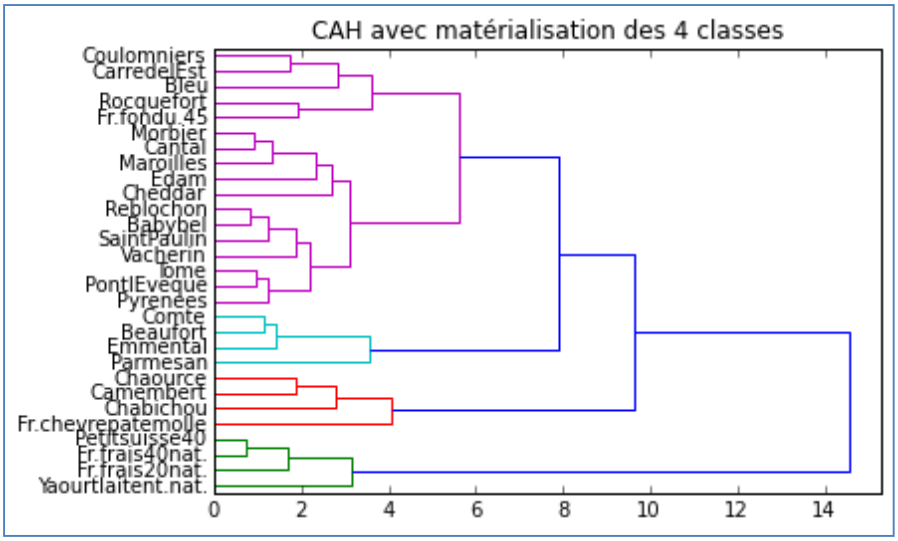
```
#matérialisation des 4 classes (hauteur t = 7)
plt.title('CAH avec matérialisation des 4 classes')
dendrogram(Z,labels=fromage.index,orientation='left',color_threshold=7)
plt.show()

#découpage à la hauteur t = 7 ==> identifiants de 4 groupes obtenus
groupes_cah = fcluster(Z,t=7,criterion='distance')
print(groupes_cah)

#index triés des groupes
import numpy as np
idg = np.argsort(groupes_cah)

#affichage des observations et leurs groupes
print(pandas.DataFrame(fromage.index[idg],groupes_cah[idg]))
```

Groupe	Fromage
1	Yaourtlaitent.nat.
1	Fr.frais20nat.
1	Petitsuisse40
1	Fr.frais40nat.
2	Fr.chevrepatemolle
2	Camembert
2	Chabichou
2	Chaource
3	Emmental
3	Parmesan
3	Beaufort
3	Comte
4	Pyrenees
4	PontlEveque
4	Rocquefort
4	SaintPaulin
4	Tome
4	Reblochon
4	CarredelEst
4	Maroilles
4	Vacherin
4	Edam
4	Coulomniers
4	Cheddar
4	Cantal
4	Bleu
4	Babybel
4	Morbier
4	Fr.fondu.45



Le 1er groupe est constitué de fromages frais.
Le 2nd de fromages à pâte molle.
Le 3ème de fromages « durs ».
Le 4ème est un peu fourre-tout (de mon point de vue).

Mes compétences en fromage s'arrêtent là (merci à Wikipédia). Pour une caractérisation à l'aide des variables de l'étude, il faut passer par des techniques statistiques univariées (simples à lire) ou multivariées (tenant compte des relations entre les variables).

Méthode des centres mobiles

K-MEANS

Méthode des centres mobiles

Utilisation du package « **scikit-learn** »

```
#k-means sur les données centrées et réduites
from sklearn import cluster
kmeans = cluster.KMeans(n_clusters=4)
kmeans.fit(fromage_cr)

#index triés des groupes
idk = np.argsort(kmeans.labels_)

#affichage des observations et leurs groupes
print(pandas.DataFrame(fromage.index[idk],kmeans.labels_[idk]))

#distances aux centres de classes des observations
print(kmeans.transform(fromage_cr))

#correspondance avec les groupes de la CAH
pandas.crosstab(groupes_cah,kmeans.labels_)
```

Groupes issus du clustering

Classe	Fromages
0	CarredelEst
0	Camembert
0	Fr.chevrepatemolle
0	Chabichou
0	Chaource
0	Coulomniers
1	Petitsuisse40
1	Fr.frais40nat.
1	Fr.frais20nat.
1	Yaourtlaitent.nat.
2	Parmesan
2	Edam
2	Emmental
2	Beaufort
2	Comte
3	Tome
3	SaintPaulin
3	Rocquefort
3	Reblochon
3	Pyrenees
3	PontlEveque
3	Cheddar
3	Morbier
3	Maroilles
3	Bleu
3	Vacherin
3	Cantal
3	Babybel
3	Fr.fondu.45

Groupe	0	1	2	3
CarredelEst	2.22	5.53	5.22	2.92
Babybel	3.02	5.19	2.79	0.74
Beaufort	5.16	7.51	1.15	2.86
Bleu	3.24	6.12	3.90	2.11
Camembert	1.93	5.40	5.10	3.54
Cantal	4.02	6.30	2.20	1.19
Chabichou	1.78	5.93	4.53	3.39
Chaource	1.03	5.55	5.09	3.46
Cheddar	3.75	6.82	2.29	1.95
Comte	5.44	7.84	1.35	3.42
Coulomniers	1.96	4.84	4.75	2.49
Edam	4.23	6.13	1.34	2.25
Emmental	5.53	7.48	0.90	3.40
Fr.chevrepatemolle	3.10	5.01	7.09	5.54
Fr.fondu.45	2.84	5.28	4.45	1.89
Fr.frais20nat.	5.33	0.68	7.61	6.00
Fr.frais40nat.	4.55	1.01	7.32	5.61
Maroilles	4.20	6.46	2.45	1.53
Morbier	3.47	6.06	2.50	0.65
Parmesan	5.23	7.94	2.11	3.60
Petitsuisse40	4.38	1.21	7.13	5.40
PontlEveque	3.08	4.69	3.52	1.26
Pyrenees	3.28	5.71	2.81	0.71
Reblochon	2.74	5.31	2.94	0.81
Rocquefort	3.02	6.81	4.22	2.18
SaintPaulin	3.47	5.12	2.67	1.37
Tome	2.73	5.25	3.67	1.26
Vacherin	3.79	5.27	2.63	1.50
Yaourtlaitent.nat.	6.09	1.93	7.46	5.94

Distances aux centres de classes pour chaque individu (avec en couleur les min. respectifs)

col_0	0	1	2	3
row_0				
1	0	4	0	0
2	4	0	0	0
3	0	0	4	0
4	2	0	1	14

Correspondance CAH – K-Means

Le groupe 1 de la CAH coïncide avec le groupe 1 des K-Means. Après, il y a certes des correspondances, mais elles ne sont pas exactes.

Méthode des centres mobiles

Aide à la détection du nombre adéquat de groupes

K-MEANS, à la différence de la CAH, ne fournit pas d'outils d'aide à la détection du nombre de classes. Nous devons les programmer sous Python ou utiliser des procédures proposées par des packages dédiés. Le schéma est souvent le même : on fait varier le nombre de groupes et on surveille l'évolution d'un indicateur de qualité de la solution c.-à-d. l'aptitude des individus à être plus proches de ses congénères du même groupe que des individus des autres groupes.

Dans ce qui suit, on calcule la métrique « silhouette » pour différents nombres de groupes issus de la méthode des centres mobiles.

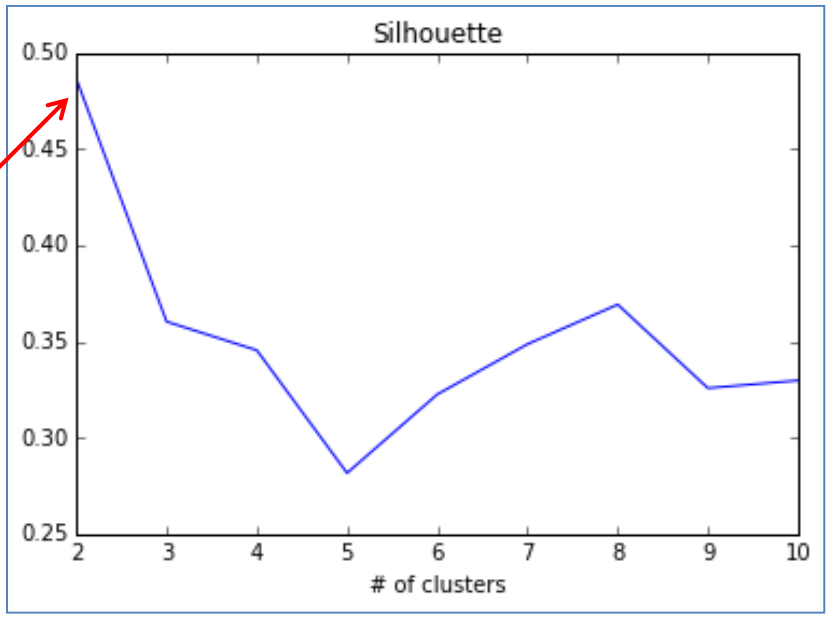
```
#librairie pour évaluation des partitions
from sklearn import metrics

#utilisation de la métrique "silhouette"
#faire varier le nombre de clusters de 2 à 10
res = np.arange(9,dtype="double")
for k in np.arange(9):
    km = cluster.KMeans(n_clusters=k+2)
    km.fit(fromage_cr)
    res[k] = metrics.silhouette_score(fromage_cr,km.labels_)

print(res)

#graphique
import matplotlib.pyplot as plt
plt.title("Silhouette")
plt.xlabel("# of clusters")
plt.plot(np.arange(2,11,1),res)
plt.show()
```

La partition en k = 2 groupes semble la meilleure au sens de la métrique « silhouette ».



Analyses univariées et multivariées

INTERPRÉTATION DES CLASSES

Interprétation des classes

Statistiques comparatives

L'idée est de comparer les moyennes des variables actives conditionnellement aux groupes. Il est possible de quantifier globalement l'amplitude des écarts avec la proportion de variance expliquée (carré du rapport de corrélation). La démarche peut être étendue aux variables illustratives. Pour les catégorielles, nous confronterions les distributions conditionnelles.

L'approche est simple et les résultats faciles à lire. Rappelons cependant que nous ne tenons pas compte des liaisons entre les variables dans ce cas.

```
#moyenne par variable
m = fromage.mean()
#TSS
TSS = fromage.shape[0]*fromage.var(ddof=0)
print(TSS)
#data.frame conditionnellement aux groupes
gb = fromage.groupby(kmeans.labels_)
#effectifs conditionnels
nk = gb.size()
print(nk)
#moyennes conditionnelles
mk = gb.mean()
print(mk)
#pour chaque groupe écart à la moyenne par
variable
EMk = (mk-m)**2
#pondéré par les effectifs du groupe
EM = EMk.multiply(nk,axis=0)
#somme des valeurs => BSS
BSS = np.sum(EM,axis=0)
print(BSS)
#carré du rapport de corrélation
#variance expliquée par l'appartenance aux groupes
#pour chaque variable
R2 = BSS/TSS
print(R2)
```

Effec. Cond.	
0	4
1	5
2	6
3	14

Carré Rapport de corr.	
calories	0.863799
sodium	0.599117
calcium	0.620108
lipides	0.851983
retinol	0.382815
folates	0.760722
proteines	0.810316
cholesterol	0.797596
magnesium	0.796207

La définition des groupes est – avant tout – dominée par les teneurs en graisses (lipides, cholestérol et calories relèvent de la même idée) et en protéines.

Le groupe n°0 est fortement déterminé par ces variables, les moyennes conditionnelles sont très différentes.

Moyennes conditionnelles

	calories	sodium	calcium	lipides	retinol	folates
0	101.750000	44.750000	133.75	6.275000	55.150000	16.475000
1	377.200000	130.400000	278.98	29.460000	64.560000	3.120000
2	288.000000	252.916667	110.10	23.866667	95.866667	31.266667
3	334.285714	267.428571	199.70	27.500000	60.050000	7.728571

	proteines	cholesterol	magnesium
0	7.200000	18.250000	11.250000
1	29.120000	102.000000	45.400000
2	18.883333	68.333333	21.666667
3	21.228571	83.571429	27.142857

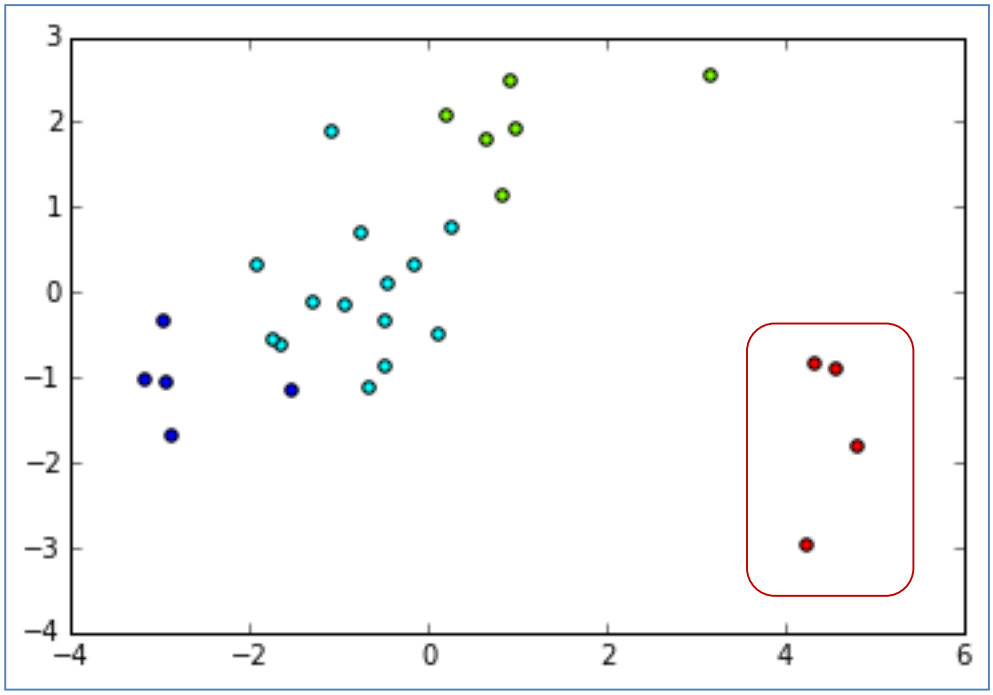
Interprétation des classes

Analyse en composantes principales (ACP)

Avec l'ACP, nous tenons compte des liaisons entre les variables. L'analyse est plus riche. Mais il faut savoir lire correctement les sorties de l'ACP.

```
#ACP
from sklearn.decomposition import PCA
acp = PCA(n_components=2).fit_transform(fromage_cr)

#projeter dans le plan factoriel
#avec un code couleur différent selon le groupe
#remarquer le rôle de zip() dans la boucle
for couleur,k in zip(['red','blue','lawngreen','aqua'],[0,1,2,3]):
   plt.scatter(acp[kmeans.labels_==k,0],acp[kmeans.labels_==k,1],c=couleur)
plt.show()
```



Il y a un problème. Le groupe des fromages frais (n° de groupe = 0) écrase l'information disponible et tasse les autres fromages dans un bloc qui s'oriente différemment.

De fait, si l'on comprend bien la nature du groupe n°0 des fromages frais, les autres sont plus compliqués à comprendre lorsqu'ils sont replacés dans le premier plan factoriel.

A la lumière des résultats de l'ACP

COMPLÉTER L'ANALYSE

Approfondir l'analyse

Retirer les fromages frais du jeu de données

Les fromages frais sont tellement particuliers – éloignés de l'ensemble des autres observations – qu'ils masquent des relations intéressantes qui peuvent exister entre ces produits. Nous reprenons l'analyse en les excluant des traitements.

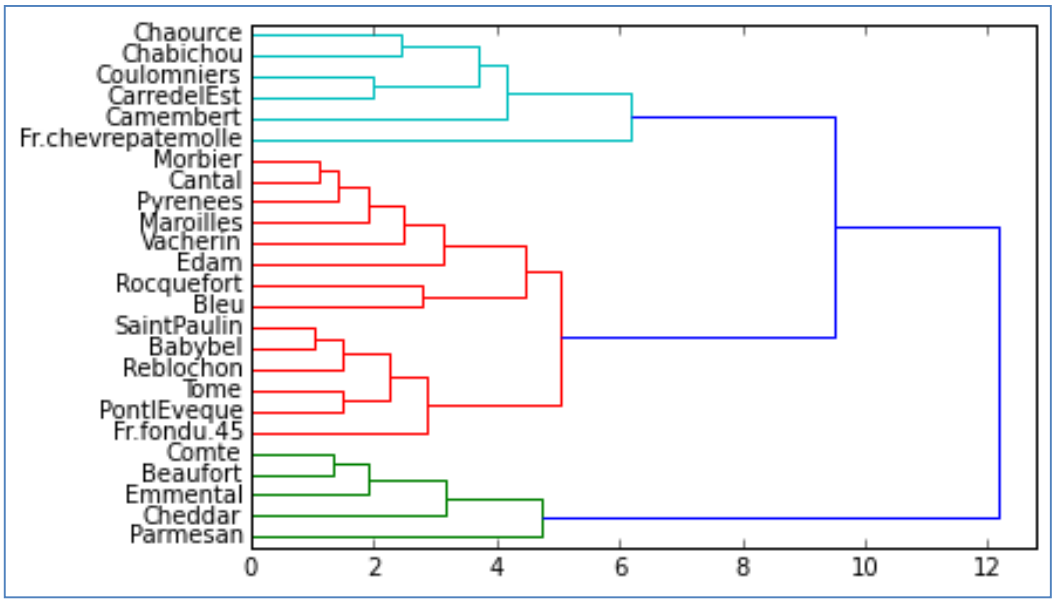
```
#retirer des observations le groupe n°0 du k-means précédent
fromage_subset = fromage.iloc[kmeans.labels_!=0,:]
print(fromage_subset.shape)

#centrer et réduire
fromage_subset_cr = preprocessing.scale(fromage_subset)

#générer la matrice des liens
Z_subset = linkage(fromage_subset_cr,method='ward',metric='euclidean')

#cah et affichage du dendrogramme
plt.title("CAH")
dendrogram(Z_subset,labels=fromage_subset.index,orientation='left',color_threshold=7)
plt.show()

#groupes
groupes_subset_cah = fcluster(Z_subset,t=7,criterion='distance')
print(groupes_subset_cah)
```



3 groupes se distinguent. On a moins le phénomène d'écrasement constaté dans l'analyse précédente.

Approfondir l'analyse

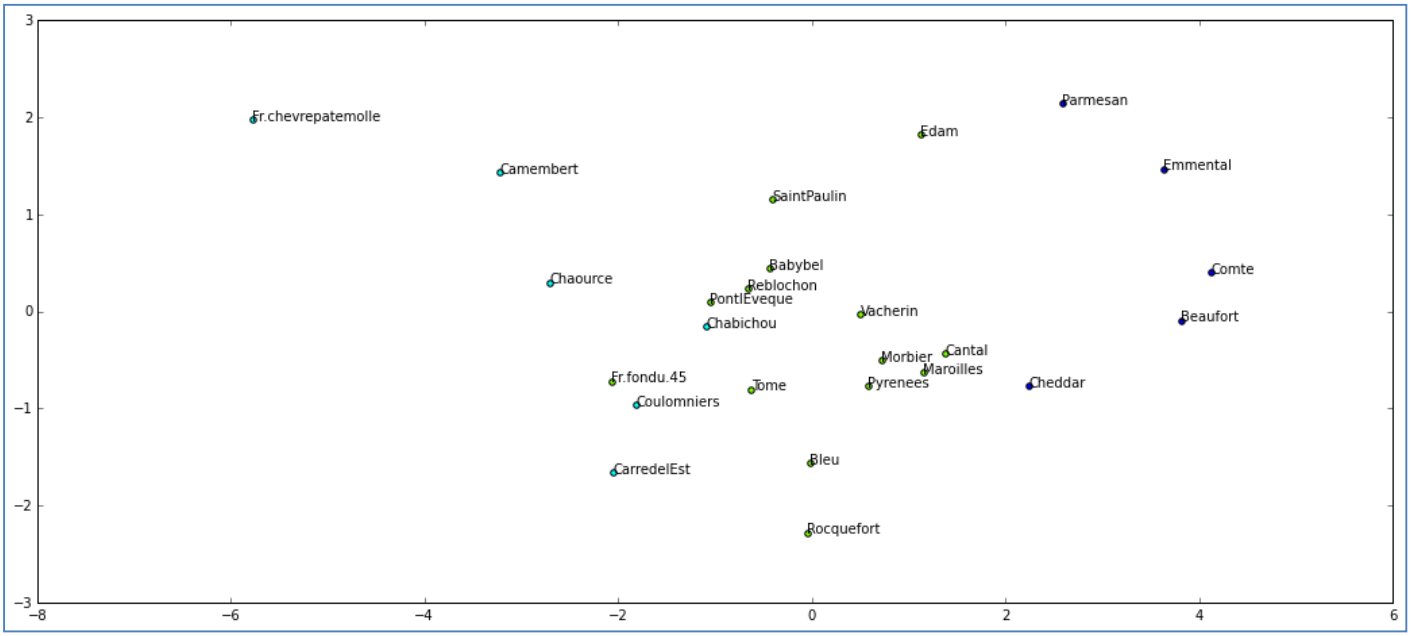
Retirer les fromages frais du jeu de données (2/2)

```
#ACP
acp_subset = PCA(n_components=2).fit_transform(fromage_subset_cr)

#projeter dans le plan factoriel
#avec un code couleur selon le groupe
#remarquer le rôle de zip()
plt.figure(figsize=(10,10))
for couleur,k in zip(['blue','lawngreen','aqua'],[1,2,3]):
            plt.scatter(acp_subset[groupes_subset_cah==k,0],acp_subset[groupes_subset_cah==k,1],c=couleur)

#mettre les labels des points
#remarquer le rôle de enumerate()
for i,label in enumerate(fromage_subset.index):
    plt.annotate(label,(acp_subset[i,0],acp_subset[i,1]))

plt.show()
```



Les groupes sont constitués essentiellement sur le 1er facteur.

Quelques fromages ont changé de camp par rapport à l'analyse précédente.

Et on peut faire bien d'autres choses encore...

Références :

1. Chavent M. , Page de cours - Source des données « fromages.txt »
2. Jörn's Blog, « SciPy Hierarchical Clustering and Dendrogram Tutorial ».
3. F. Pedregosa, G. Varoquaux, « Scikit-learn: machine learning in Python ».