AFDM sous Python avec scientisttools

Duvérier DJIFACK ZEBAZE

Ce tutoriel a pour objectif de présenter rapidement les principales fonctionnalités offertes par le package « scientisttools » pour réaliser une Analyse Factorielle des Données Mixtes.

Présentation des données

L'analyse factorielle des données mixtes traite les tableaux individus-variables, lesquelles sont composées d'un mix de quantitatives et qualitatives. Nous utilisons la base des données « Autos 2005 » accessible sur la page de cours de Pierre-Louis Gonzalez (https://maths.cnam.fr/spip.php?article50) au CNAM. Notre base comporte ($I = 38$) modèles de véhicules décrits par ($K_1 = 9$) variables quantitatives (puissance, cylindrée, vitesse, longueur, largeur, hauteur, poids, CO2 et prix) et ($K_2 = 3$) variables qualitatives (origie avec 3 modalités : France, Europe, Autres ; carburant avec 2 modalités : diesel, essence ; type4X4 avec 2 modalités : oui, non).

Importation des données

```
# Chargement des données
import pandas as pd
# Données actives
A = pd.read_excel("./donnee/autos2005.xlsx",sheet_name=0,index_col=0)
# Individus actifs
B = pd.read_excel("./donnee/autos2005.xlsx",sheet_name=1,index_col=0)
# Variables illustratives quantitatives
C = pd.read_excel("./donnee/autos2005.xlsx",sheet_name=2,index_col=0)
# Variables illustratives qualitatives
D = pd.read_excel("./donnee/autos2005.xlsx",sheet_name=3,index_col=0)
C.index = D.index = A.index
# Concaténation
Data = pd.concat([pd.concat([A,B],axis=0),C,D],axis=1)
# Affichage des caractéristiques
Data.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## Index: 45 entries, ALFA 156     to MERCEDES
## Data columns (total 16 columns):
##  #   Column        Non-Null Count  Dtype
## ---  ------        --------------  -----
##  0   puissance     45 non-null     int64
##  1   cylindree     45 non-null     int64
```

```
##  2   vitesse       45 non-null     int64
##  3   longueur      45 non-null     int64
##  4   largeur       45 non-null     int64
##  5   hauteur       45 non-null     int64
##  6   poids         45 non-null     int64
##  7   CO2           45 non-null     int64
##  8   prix          45 non-null     int64
##  9   origine       45 non-null     object
##  10  carburant     45 non-null     object
##  11  type4X4       45 non-null     object
##  12  coffre        38 non-null     float64
##  13  reservoir     38 non-null     float64
##  14  consommation  38 non-null     float64
##  15  surtaxe       38 non-null     object
## dtypes: float64(3), int64(9), object(4)
## memory usage: 6.0+ KB
```

Table 1 – Données Autos 2005

	puissance	cylindree	vitesse	longueur	largeur	hauteur	poids	CO2	prix	origine	carburant	type4X4	coffre	reservoir	consommation	surtaxe
ALFA 156	250	3179	250	443	175	141	1410	287	40800	Europe	Essence	type4X4_non	378	63	12.1	surtaxe_oui
AUDIA3	102	1595	185	421	177	143	1205	168	21630	Europe	Essence	type4X4_non	350	55	7.0	surtaxe_oui
AUDIA8	280	3697	250	506	203	145	1770	281	78340	Europe	Essence	type4X4_non	500	90	11.7	surtaxe_non
AVENSIS	115	1995	195	463	176	148	1400	155	26400	Autres	Diesel	type4X4_non	510	60	5.8	surtaxe_oui
BMW X5	218	2993	210	467	188	172	2095	229	52000	Europe	Diesel	type4X4_non	465	93	8.6	surtaxe_oui
BMW530	231	2979	250	485	185	147	1495	231	46400	Europe	Essence	type4X4_non	520	70	9.5	surtaxe_non
CHRYS300	340	5654	250	502	188	148	1835	291	54900	Autres	Essence	type4X4_non	442	72	12.2	surtaxe_non
CITRONC2	61	1124	158	367	166	147	932	141	10700	France	Essence	type4X4_non	224	41	5.9	surtaxe_non
CITRONC4	138	1997	207	426	178	146	1381	142	23400	France	Diesel	type4X4_non	314	60	5.4	surtaxe_oui
CITRONC5	210	2496	230	475	178	148	1589	238	33000	France	Essence	type4X4_non	471	65	10.0	surtaxe_non
CLIO	100	1461	185	382	164	142	980	113	17600	France	Diesel	type4X4_non	255	50	4.3	surtaxe_non
CORSA	70	1248	165	384	165	144	1035	127	13590	Europe	Diesel	type4X4_non	260	45	4.7	surtaxe_oui
FIESTA	68	1399	164	392	168	144	1138	117	14150	Europe	Diesel	type4X4_non	261	45	4.4	surtaxe_oui
GOLF	75	1968	163	421	176	149	1217	143	19140	Europe	Diesel	type4X4_non	350	55	5.4	surtaxe_oui
LAGUNA	165	1998	218	458	178	143	1320	196	25350	France	Essence	type4X4_non	430	70	8.2	surtaxe_non
LANDCRUI	204	4164	170	489	194	185	2495	292	67100	Autres	Diesel	type4X4_oui	403	96	11.1	surtaxe_oui
MAZDARX8	231	1308	235	443	177	134	1390	284	34000	Autres	Essence	type4X4_non	287	61	11.4	surtaxe_oui
MEGANECC	165	1998	225	436	178	141	1415	191	27800	France	Essence	type4X4_non	190	60	8.0	surtaxe_oui
MERC_A	140	1991	201	384	177	160	1340	141	24550	Europe	Diesel	type4X4_non	435	54	5.4	surtaxe_oui
MERC_E	204	3222	243	482	183	146	1735	183	46450	Europe	Diesel	type4X4_non	520	80	6.9	surtaxe_non
MODUS	113	1598	188	380	170	159	1170	163	16950	France	Essence	type4X4_non	198	49	6.8	surtaxe_oui
MONDEO	145	1999	215	474	194	143	1378	189	23100	Europe	Essence	type4X4_non	500	59	7.9	surtaxe_oui
MURANO	234	3498	200	477	188	171	1870	295	44000	Autres	Essence	type4X4_oui	438	82	12.3	surtaxe_oui
MUSA	100	1910	179	399	170	169	1275	146	17900	Europe	Diesel	type4X4_non	320	47	5.5	surtaxe_non
OUTLAND	202	1997	220	455	178	167	1595	237	29990	Autres	Diesel	type4X4_oui	402	60	10.0	surtaxe_oui
P1007	75	1360	165	374	169	161	1181	153	13600	France	Essence	type4X4_non	178	50	6.4	surtaxe_oui
P307CC	180	1997	225	435	176	143	1490	210	28850	France	Essence	type4X4_non	204	50	8.8	surtaxe_non
P407	136	1997	212	468	182	145	1415	194	23400	France	Essence	type4X4_non	407	66	8.2	surtaxe_non
P607	204	2721	230	491	184	145	1723	223	40550	France	Diesel	type4X4_non	468	80	8.4	surtaxe_oui
PANDA	54	1108	150	354	159	154	860	135	8070	Europe	Essence	type4X4_non	206	35	5.7	surtaxe_non
PASSAT	150	1781	221	471	175	147	1360	197	27740	Europe	Essence	type4X4_non	475	62	8.2	surtaxe_non
PTCRUISER	223	2429	200	429	171	154	1595	235	27400	Autres	Essence	type4X4_non	210	57	9.9	surtaxe_oui
SANTA_FE	125	1991	172	450	185	173	1757	197	27990	Autres	Diesel	type4X4_oui	833	65	7.5	surtaxe_oui
TWINGO	60	1149	151	344	163	143	840	143	8950	France	Essence	type4X4_non	168	40	6.0	surtaxe_non
VECTRA	150	1910	217	460	180	146	1428	159	26550	Europe	Diesel	type4X4_non	500	61	5.9	surtaxe_oui
VELSATIS	150	2188	200	486	186	158	1735	188	38250	France	Diesel	type4X4_non	460	80	7.1	surtaxe_oui
X-TRAIL	136	2184	180	446	177	168	1520	190	29700	Autres	Diesel	type4X4_oui	350	60	7.2	surtaxe_oui
YARIS	65	998	155	364	166	150	880	134	10450	Autres	Essence	type4X4_non	205	45	5.6	surtaxe_non
CORVETTE	404	5970	300	444	185	125	1517	310	63350	Autres	Essence	type4X4_non	NaN	NaN	NaN	NA
TAHOE	290	5327	170	506	223	196	2463	340	49600	Autres	Essence	type4X4_oui	NaN	NaN	NaN	NA
OPEL	339	4188	227	351	192	181	2235	166	77810	France	Diesel	type4X4_non	NaN	NaN	NaN	NA
RENAULT	183	4439	160	387	186	137	1177	228	31681	France	Essence	type4X4_oui	NaN	NaN	NaN	NA
CITROEN	88	4161	231	367	182	139	947	185	51161	Europe	Essence	type4X4_non	NaN	NaN	NaN	NA
TOYOTA	83	1880	232	373	195	149	1229	118	51233	Autres	Diesel	type4X4_non	NaN	NaN	NaN	NA
MERCEDES	311	5361	188	375	159	145	874	339	72876	Europe	Diesel	type4X4_oui	NaN	NaN	NaN	NA

Les variables coffre, reservoir et consommation seront utilisées comme illustratives quantitatives et « surtaxe » comme illustrative qualitative. Certaines voitures ont été mises en illustratives.

Les questions usuelles que l'on se pose sont les suivantes :

1. Quelles sont les véhicules qui se ressemblent, c'est - à - dire qui présentent des caractéristiques similaires ? Il sera question d'étudier les proximités entre les individus.
2. Sur quelles caractéristiques sont basées les ressemblances et dissemblances, avec la difficulté ici de les comptabiliser de manière différenciée selon que les variables incriminées sont quantitatives ou qualitatives.

3. Quelles sont les relations entre les variables ? Entre quantitatives, l'idée de la corrélation s'impose ; entre qualitatives, le χ^2 de contingence. Mais comment faire entre quantitatives et qualitatives ? (rapport de corrélation, etc.)

AFDM

Objectifs

L'objectif est de trouver un système de représentation (répère factoriel) qui préserve au mieux les distances entre les individus, qui permet de discerner le mieux possible les individus entre eux, qui maximise les (le carré des) écarts à l'origine.

Chargement de scientisttools

```
from scientisttools.decomposition import FAMD
```

Individus et variables actifs

On crée une instance de la classe FAMD, en lui passant ici des étiquettes pour les lignes et les variables. Ces paramètres sont facultatifs ; en leur absence, le programme détermine automatiquement des étiquettes.

```
# Instanciation
my_famd = FAMD(normalize=True,
               n_components=None,
               row_labels= A.index,
               quanti_labels= list(A.columns[:9]),
               quali_labels=list(A.columns[9:]),
               row_sup_labels=None,
               quanti_sup_labels=None,
               quali_sup_labels=None,
               parallelize=False)
```

On estime le modèle en appliquant la méthode `fit` de la classe FAMD sur le jeu de données.

```
# Estimation du modèle
my_famd.fit(A)
```

```
## FAMD(quali_labels=['origine', 'carburant', 'type4X4'],
##      quanti_labels=['puissance', 'cylindree', 'vitesse', 'longueur', 'largeur',
##                     'hauteur', 'poids', 'CO2', 'prix'],
##      row_labels=Index(['ALFA 156    ', 'AUDIA3      ', 'AUDIA8     ', 'AVENSIS     ',
##        'BMW X5      ', 'BMW530      ', 'CHRYS300    ', 'CITRONC2    ',
##        'CITRONC4    ', 'CITRONC5    ', 'CLIO        ', 'CORSA       ',
##        'FIESTA      ', 'GOLF        ', 'LAGUNA      ', 'LANDCRUI    ',
##        'MAZDARX8    ', 'MEGANECC      ', 'MERC_A      ', 'MERC_E      ',
##        'MODUS       ', 'MONDEO      ', 'MURANO      ', 'MUSA        ',
```

```
##        'OUTLAND     ', 'P1007       ', 'P307CC      ', 'P407        ',
##        'P607        ', 'PANDA       ', 'PASSAT      ', 'PTCRUISER   ',
##        'SANTA_FE    ', 'TWINGO      ', 'VECTRA      ', 'VELSATIS    ',
##        'X-TRAIL     ', 'YARIS       '],
##       dtype='object', name='Modele'))
```

Les valeurs propres

L'exécution de la méthode `my_famd.fit(A)` provoque le calcul des attributs parmi lesquels `my_famd.eig_` pour les valeurs propres.

```
# Valeurs propres
print(my_famd.eig_)
```

```
## [[6.60229295e+00 2.51003566e+00 1.30489598e+00 8.66766581e-01
##   5.57532258e-01 3.89580887e-01 2.67694431e-01 1.72089295e-01
##   1.40012335e-01 9.66725743e-02 5.08904567e-02 3.10797814e-02
##   1.04568068e-02]
##  [4.09225728e+00 1.20513968e+00 4.38129402e-01 3.09234324e-01
##   1.67951371e-01 1.21886456e-01 9.56051353e-02 3.20769600e-02
##   4.33397610e-02 4.57821177e-02 1.98106752e-02 2.06229746e-02
##   1.04568068e-02]
##  [5.07868688e+01 1.93079666e+01 1.00376614e+01 6.66743524e+00
##   4.28870967e+00 2.99677605e+00 2.05918793e+00 1.32376381e+00
##   1.07701796e+00 7.43635187e-01 3.91465051e-01 2.39075242e-01
##   8.04369757e-02]
##  [5.07868688e+01 7.00948355e+01 8.01324969e+01 8.67999321e+01
##   9.10886418e+01 9.40854178e+01 9.61446058e+01 9.74683696e+01
##   9.85453875e+01 9.92890227e+01 9.96804878e+01 9.99195630e+01
##   1.00000000e+02]]
```

L'attribut `my_famd.eig_` contient :

- en 1ère ligne : les valeurs propres en valeur absolue
- en 2ème ligne : les différences des valeurs propres
- en 3ème ligne : les valeurs propres en pourcentage de la variance totale (proportions)
- en 4ème ligne : les valeurs propres en pourcentage cumulé de la variance totale.

La fonction `get_eig` retourne les valeurs propres sous forme de tableau de données.

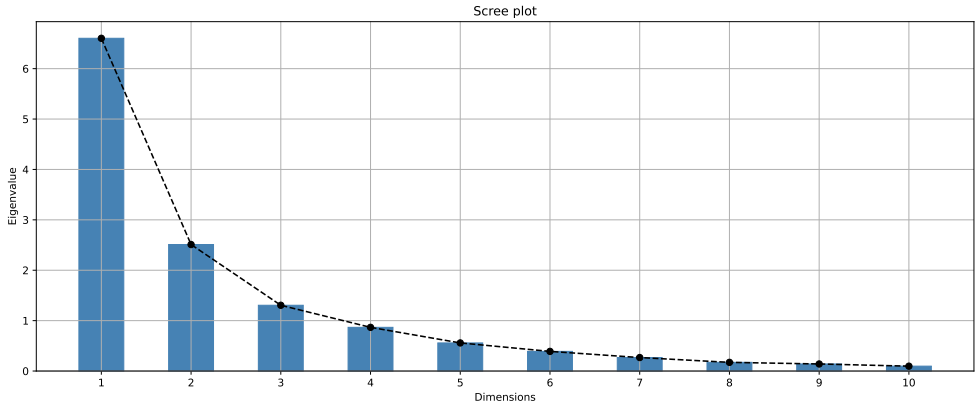
```
# Valeurs propres
from scientisttools.extractfactor import get_eig
print(get_eig(my_famd))
```

```
##         eigenvalue  difference  proportion  cumulative
## Dim.1     6.602293    4.092257   50.786869   50.786869
## Dim.2     2.510036    1.205140   19.307967   70.094835
## Dim.3     1.304896    0.438129   10.037661   80.132497
## Dim.4     0.866767    0.309234    6.667435   86.799932
## Dim.5     0.557532    0.167951    4.288710   91.088642
## Dim.6     0.389581    0.121886    2.996776   94.085418
```

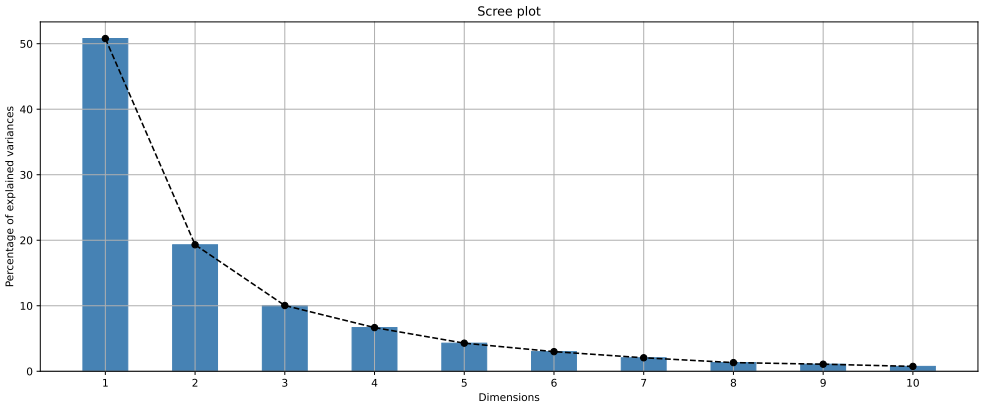
```
## Dim.7     0.267694    0.095605    2.059188   96.144606
## Dim.8     0.172089    0.032077    1.323764   97.468370
## Dim.9     0.140012    0.043340    1.077018   98.545388
## Dim.10    0.096673    0.045782    0.743635   99.289023
## Dim.11    0.050890    0.019811    0.391465   99.680488
## Dim.12    0.031080    0.020623    0.239075   99.919563
## Dim.13    0.010457    0.010457    0.080437  100.000000
```

Les valeurs propres peuvent être représentées graphiquement :

```
from scientisttools.pyplot import plot_eigenvalues
import matplotlib.pyplot as plt
fig, axe = plt.subplots(figsize=(16,6))
plot_eigenvalues(my_famd,choice="eigenvalue",ax=axe)
plt.show()
```




```
fig, axe = plt.subplots(figsize=(16,6))
plot_eigenvalues(my_famd,choice="proportion",ax=axe)
plt.show()
```



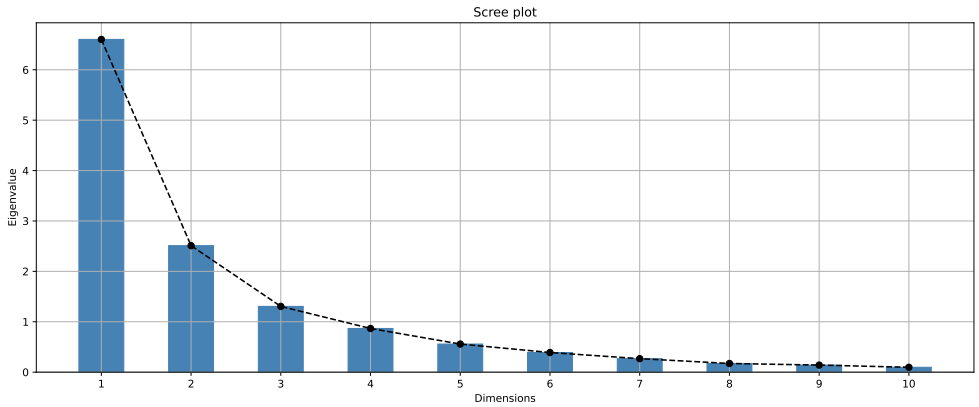
Même si le mécanisme sous - jacent de l'AFDM repose sur une ACP, nous ne pouvons pas vraiment utiliser les stratégies usuelles.

— Avec la règle de Kaiser, nous sélectionnerons les facteurs tels que $\lambda_\alpha \geq 1$, c'est - à - dire $H = 3$. On se rend compte en pratique que ce critère est trop permissif, nous retenons un nombre excessif de facteurs parce qu'une partie de l'information est rédondante comme en ACM.
— Avec la règle de Karlis - Saporta - Spinaki, la valeur seuil est surestimé parce que le nombre de colonnes K des données présentées à l'ACP est « surévalué », des redondances ont été artificiellement introduites.

De fait, les critères de l'ACP ne s'appliquent pas ici parce que les données ne sont pas composées de variables nativement quantitatives, certaines colonnes sont liées entre elles avec le codage disjonctif complet des variables qualitatives.

Finalement, on en revient au diagramme des valeurs propres et la recherche de « coudes », annonciateurs de changement significatif de structure dans les données.

```
fig, axe = plt.subplots(figsize=(16,6))
plot_eigenvalues(my_famd,choice="eigenvalue",ax=axe)
plt.show()
```



Nous avons un « coude » au niveau de $h = 2$. Nous choisissons de retenir $H = 2$.

On peut obtenir un résumé des principaux résultats en utilisant la fonction `summaryFAMD`.

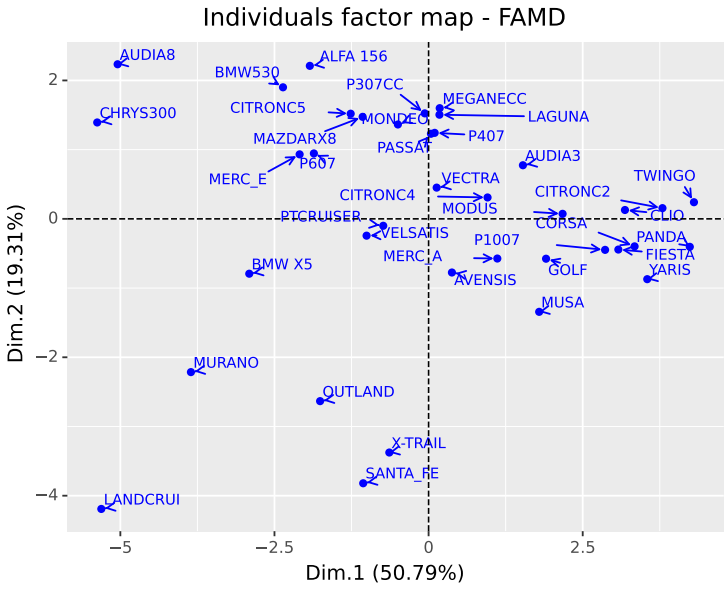
```
from scientisttools.extractfactor import summaryFAMD
summaryFAMD(my_famd)
```

```
##                      Factor Analysis of Mixed Data - Results
##
## Importance of components
##                            Dim.1   Dim.2   Dim.3  ...  Dim.11  Dim.12  Dim.13
## Variance                   6.602   2.510   1.305  ...   0.051   0.031    0.01
## Difference                 4.092   1.205   0.438  ...   0.020   0.021    0.01
## % of var.                 50.787  19.308  10.038  ...   0.391   0.239    0.08
## Cumulative of % of var.   50.787  70.095  80.132  ...  99.680  99.920  100.00
##
## [4 rows x 13 columns]
##
## Individuals (the 10 first)
```

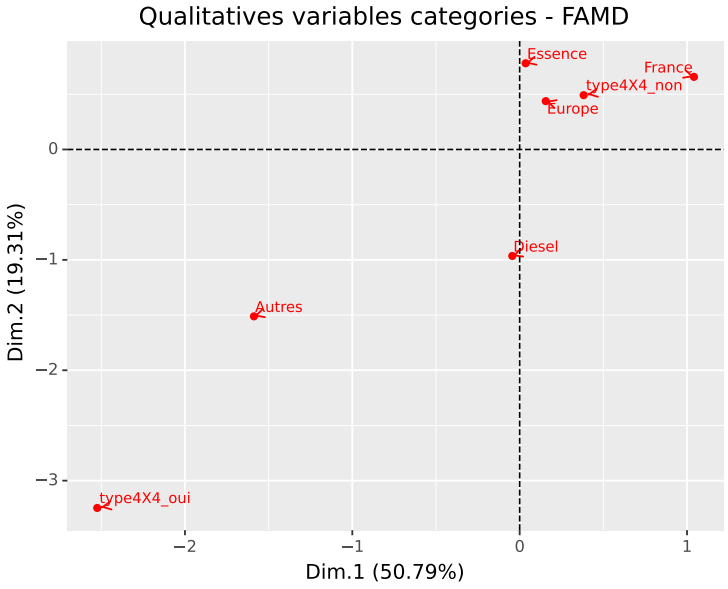
```
##
##               d(i,G)   p(i)  I(i,G)  Dim.1  ...   cos2  Dim.3    ctr   cos2
## Modele                                       ...
## ALFA 156       3.577  0.026   0.337 -1.926  ...  0.382  0.137  0.038  0.001
## AUDIA3         2.321  0.026   0.142  1.530  ...  0.111 -0.396  0.316  0.029
## AUDIA8         5.900  0.026   0.916 -5.044  ...  0.143 -0.887  1.586  0.023
## AVENSIS        2.376  0.026   0.149  0.379  ...  0.106 -0.289  0.169  0.015
## BMW X5         3.977  0.026   0.416 -2.909  ...  0.040 -1.942  7.607  0.239
## BMW530         3.283  0.026   0.284 -2.362  ...  0.335 -0.541  0.591  0.027
## CHRYS300       6.191  0.026   1.009 -5.375  ...  0.051  1.124  2.549  0.033
## CITRONC2       4.079  0.026   0.438  3.793  ...  0.001  1.192  2.866  0.085
## CITRONC4       2.203  0.026   0.128  0.956  ...  0.020 -0.492  0.489  0.050
## CITRONC5       2.530  0.026   0.168 -1.264  ...  0.361  1.003  2.028  0.157
##
## [10 rows x 12 columns]
##
## Continuous variables
##
##            Dim.1     ctr   cos2  Dim.2     ctr   cos2  Dim.3    ctr   cos2
## puissance -0.916  12.704  0.839  0.271   2.936  0.074  0.126  1.220  0.016
## cylindree -0.888  11.951  0.789  0.039   0.061  0.002 -0.046  0.160  0.002
## vitesse   -0.703   7.495  0.495  0.598  14.263  0.358 -0.039  0.115  0.001
## longueur  -0.899  12.229  0.807  0.142   0.804  0.020 -0.130  1.292  0.017
## largeur   -0.876  11.621  0.767  0.039   0.060  0.001 -0.187  2.693  0.035
## hauteur   -0.288   1.260  0.083 -0.846  28.519  0.716 -0.028  0.059  0.001
## poids     -0.915  12.693  0.838 -0.249   2.469  0.062 -0.099  0.757  0.010
## CO2       -0.891  12.037  0.795  0.090   0.323  0.008  0.327  8.202  0.107
## prix      -0.940  13.397  0.884  0.062   0.154  0.004 -0.126  1.218  0.016
##
## Categories
##
##                       d(k,G)   p(k)  I(k,G)  Dim.1  ...  Dim.3     ctr   cos2  vtest
## origine_Autres         1.673  0.038   0.105 -1.588  ...  0.896  12.409  0.287  2.851
## origine_Europe         1.238  0.056   0.086  0.156  ... -1.096  27.832  0.783 -4.712
## origine_France         1.387  0.049   0.094  1.041  ...  0.575   6.643  0.172  2.208
## carburant_Diesel       1.111  0.064   0.079 -0.044  ... -0.837  18.405  0.567 -4.010
## carburant_Essence      0.900  0.079   0.064  0.036  ...  0.678  14.899  0.567  4.010
## type4X4_type4X4_non    0.389  0.124   0.019  0.382  ... -0.103   0.539  0.070 -1.406
## type4X4_type4X4_oui    2.569  0.019   0.124 -2.524  ...  0.679   3.558  0.070  1.406
##
## [7 rows x 15 columns]
##
## Categorical variables
##
##            eta2.1  cos2.1  eta2.2  cos2.2  eta2.3  cos2.3
## origine     0.158   0.079   0.329   0.164   0.612   0.306
## carburant   0.000   0.000   0.300   0.300   0.435   0.435
## type4X4     0.146   0.146   0.637   0.637   0.053   0.053
```

Représentation graphique

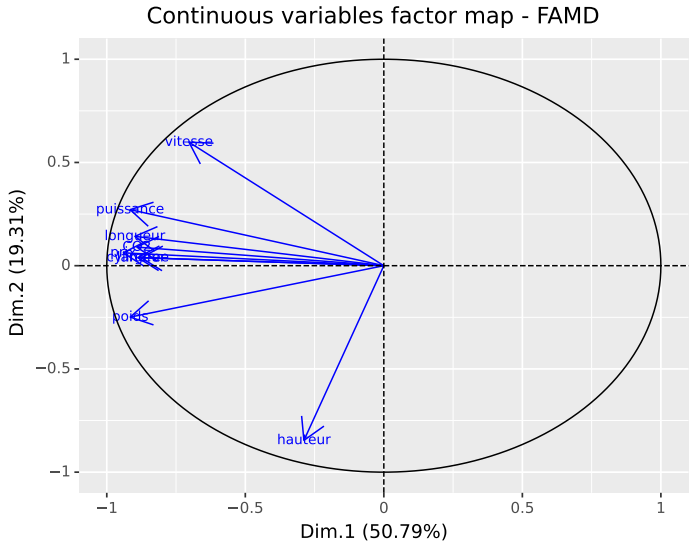
```
# Carte des individus
from scientisttools.ggplot import fviz_famd_ind
print(fviz_famd_ind(my_famd,color="blue",repel=True))
```



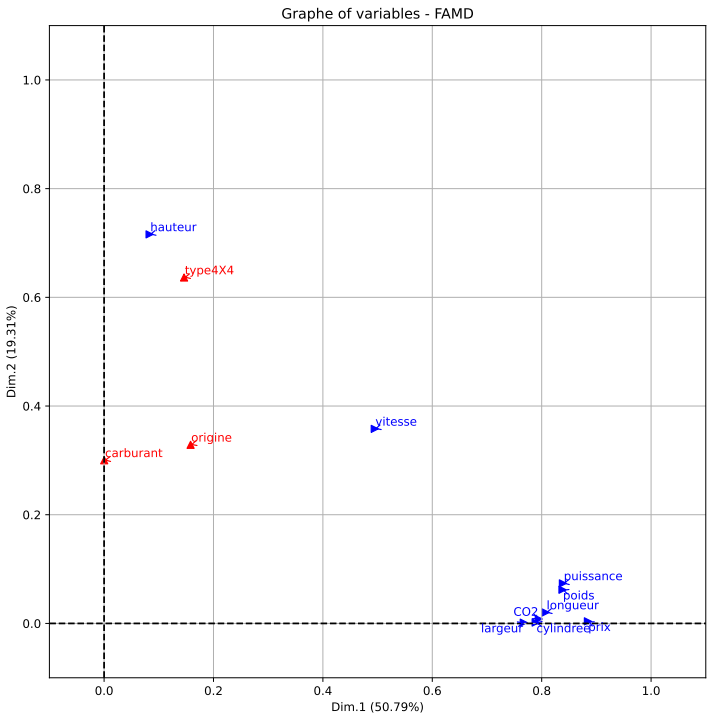

```
# Carte des modalités - variables qualitatives
from scientisttools.ggplot import fviz_famd_mod
print(fviz_famd_mod(my_famd,color="red",repel=True))
```



```
# Cercle des corrélations - variables quantitatives
from scientisttools.ggplot import fviz_famd_col
print(fviz_famd_col(my_famd,color="blue"))
```




```
# Carte des variables - rapport de corrélation et cosinus carré
from scientisttools.pyplot import plotFAMD
fig, axe = plt.subplots(figsize=(10,10))
plotFAMD(my_famd,choice="var",repel=True)
plt.show()
```



AFDM avec les éléments supplémentaires

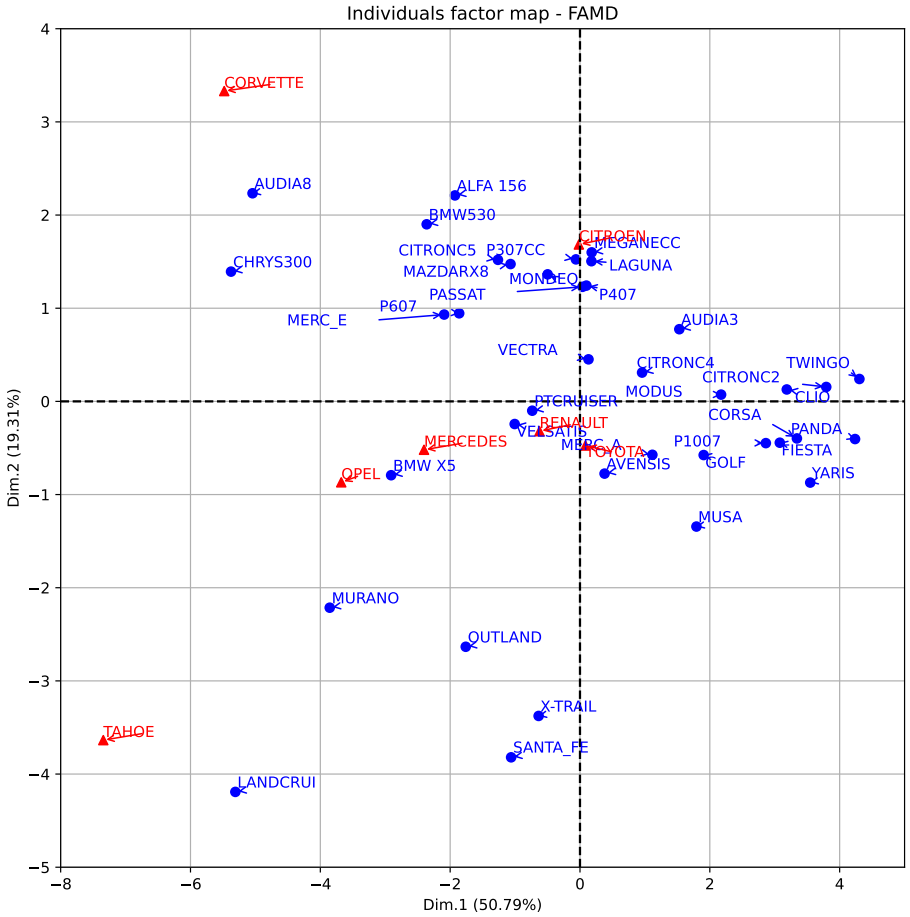
Les individus illustratifs et les variables illustratives n'influencent pas la construction des composantes principales de l'analyse. Ils/Elles aident à l'interprétation des dimensions de variabilité.

On peut ajouter deux types de variables : continues et qualitatives. Tapez la ligne de code suivante :

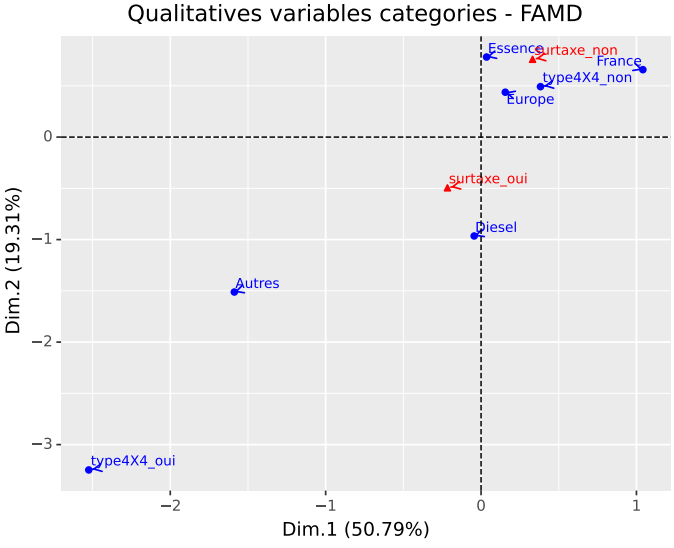
```
# AFDM avec les éléments supplémentaires
my_famd2 = FAMD(normalize=True,
                n_components=None,
                row_labels= A.index,
                quanti_labels= list(A.columns[:9]),
                quali_labels=list(A.columns[9:]),
                row_sup_labels=B.index,
                quanti_sup_labels=list(C.columns),
                quali_sup_labels=list(D.columns),
                parallelize=False)
# Estimation
my_famd2.fit(Data)
```

```
## FAMD(quali_labels=['origine', 'carburant', 'type4X4'],
##      quali_sup_labels=['surtaxe'],
##      quanti_labels=['puissance', 'cylindree', 'vitesse', 'longueur', 'largeur',
##                     'hauteur', 'poids', 'CO2', 'prix'],
##      quanti_sup_labels=['coffre', 'reservoir', 'consommation'],
##      row_labels=Index(['ALFA 156    ', 'AUDIA3      ', 'AUDIA8     ', 'AVENSIS     ',
##        'BMW X5      ', 'BMW530      ', 'CHRYS300    ', 'CITRONC2    ',
##        'CITRONC4    ', 'CITR...
##        'MAZDARX8    ', 'MEGANECC      ', 'MERC_A      ', 'MERC_E      ',
##        'MODUS       ', 'MONDEO      ', 'MURANO      ', 'MUSA        ',
##        'OUTLAND     ', 'P1007       ', 'P307CC      ', 'P407        ',
##        'P607        ', 'PANDA       ', 'PASSAT      ', 'PTCRUISER   ',
##        'SANTA_FE    ', 'TWINGO      ', 'VECTRA      ', 'VELSATIS    ',
##        'X-TRAIL     ', 'YARIS       '],
##       dtype='object', name='Modele'),
##      row_sup_labels=Index(['CORVETTE    ', 'TAHOE       ', 'OPEL', 'RENAULT', 'CITROEN',
##        'MERCEDES'],
##       dtype='object', name='Modele'))
```

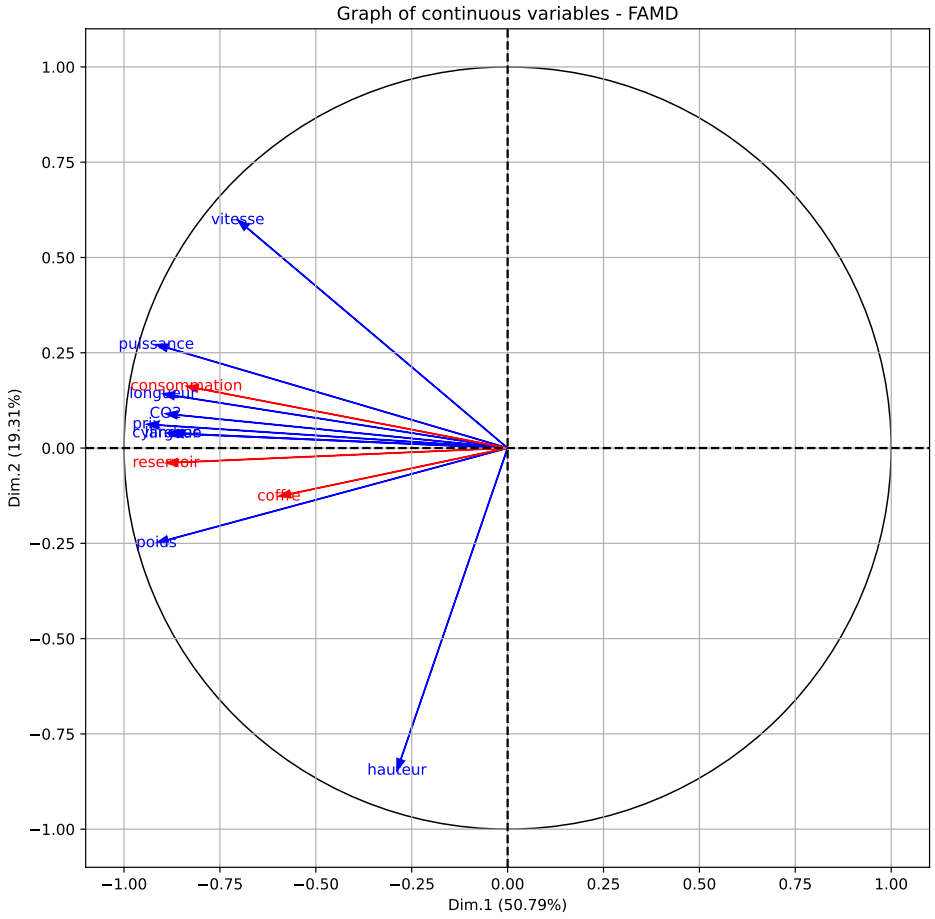
```
# Carte des individus
from scientisttools.pyplot import plotFAMD
fig, axe = plt.subplots(figsize=(10,10))
plotFAMD(my_famd2,choice="ind",ind_sup=True,xlim=(-8,5),ylim=(-5,4),
         repel=True,ax=axe)
plt.show()
```




```
# Carte des modalités
print(fviz_famd_mod(my_famd2,color="blue",repel=True))
```



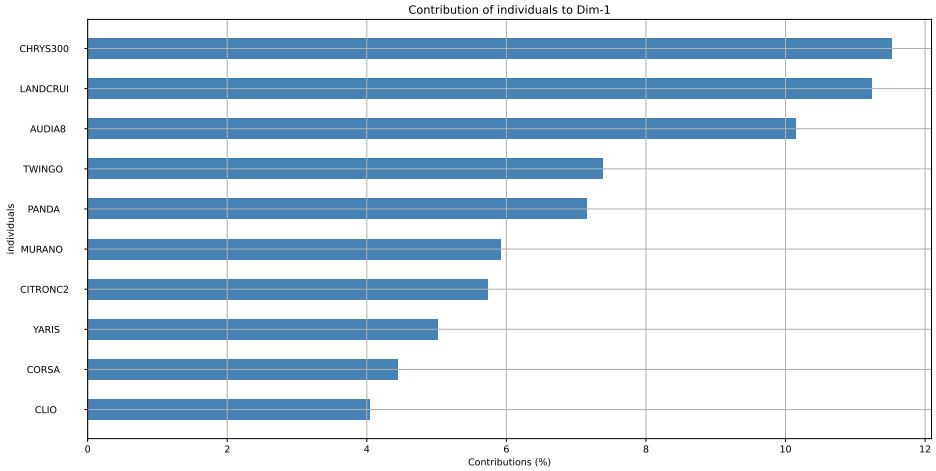
```
# Cercle des corrélations des variables
fig, axe = plt.subplots(figsize=(10,10))
plotFAMD(my_famd2,choice="col",color="blue",ax=axe)
plt.show()
```



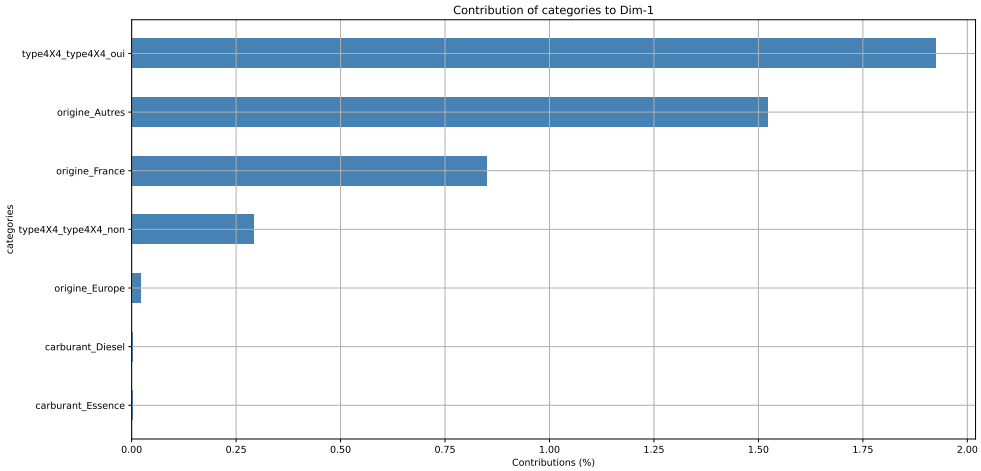
Interprétation des axes

Des graphiques qui permettent d'interpréter rapidement les axes : on choisit un axe factoriel (le 1er axe dans notre exemple) et on observe quels sont les points lignes et colonnes qui présentent les plus fortes contributions et cos2 pour cet axe.

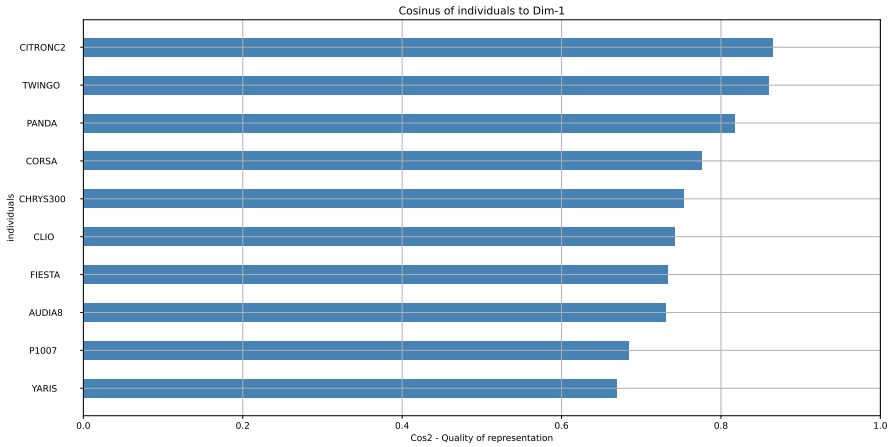
```
# Classement des points lignes en fonction de leur contribution au 1er axe
from scientisttools.pyplot import plot_contrib, plot_cosines
fig, axe = plt.subplots(figsize=(16,8))
plot_contrib(my_famd2,choice="ind",axis=0,top_contrib=10,ax=axe)
plt.show()
```



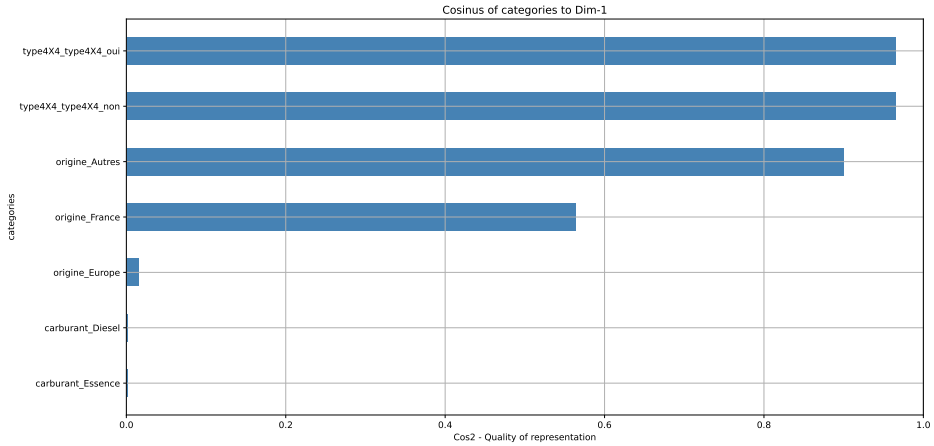

```
# Classement des modalités en fonction de leur contribution au 1er axe
fig, axe = plt.subplots(figsize=(16,8))
plot_contrib(my_famd2,choice="mod",axis=0,ax=axe)
plt.show()
```




```
# Classement des individus en fonction de leur cos2 sur le 1er axe
fig, axe = plt.subplots(figsize=(16,8))
plot_cosines(my_famd2,choice="ind",axis=0,top_cos2=10,ax=axe)
plt.show()
```




```
# Classement des modalités en fonction de leur cos2 sur le 1er axe
fig, axe = plt.subplots(figsize=(16,8))
plot_cosines(my_famd2,choice="mod",axis=0,ax=axe)
plt.show()
```



Approche Machine Learning

Ici, l'objectif est d'utiliser l'Analyse Factorielle des Données Mixtes en tant que méthode de prétraitement.

La classe FAMD implémente les méthodes `fit`, `transform` et `fit_transform` bien connues des utilisateurs de scikit-learn.

```
my_famd.transform(A)[:5,:2]
```

```
## array([[-1.92635548,  2.2106966 ],
##        [ 1.53012691,  0.77467409],
##        [-5.04419565,  2.23379347],
##        [ 0.37854815, -0.77513574],
##        [-2.90897055, -0.79259248]])
```

```
my_famd.fit_transform(A)[:5,:2]
```

```
## array([[-1.92635548,  2.2106966 ],
##        [ 1.53012691,  0.77467409],
##        [-5.04419565,  2.23379347],
##        [ 0.37854815, -0.77513574],
##        [-2.90897055, -0.79259248]])
```

Intégration dans une Pipeline de scikit-learn

La class `FAMD` peut être intégrée dans une Pipeline de scikit-learn. Dans le cadre de notre exemple, nous cherchons à prédire la variable (variable "Prix") à partir des 11 autres variables du jeu de données (données actives).

"prix" est une variable quantitative. Pour la prédire, nous allons utiliser un modèle de régression linéaire qui prendra en input des axes issus d'une Analyse Factorielle des Données Mixtes pratiquée sur les données brutes.

Dans un premier temps, et de façon tout à fait arbitraire, nous fixons le nombre de composantes extraites à 4.

```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import GridSearchCV

# X = features
X = A.drop(columns=["prix"])
# y = target
y = A.prix

# Construction de la Pipeline
# On enchaine une Analyse Factorielle des Données Mixtes (4 axes retenus)
# puis une régression linéaire
pipe = Pipeline([("famd", FAMD(n_components=4)),
                 ("ols", LinearRegression())])
# Estimation du modèle
pipe.fit(X, y)
```

```
## Pipeline(steps=[('famd', FAMD(n_components=4)), ('ols', LinearRegression())])
```

On prédit

```
# Prédiction sur l'échantillon de test
print(pipe.predict(B))
```

```
## [58779.0497945  69412.02123729 46817.29617417 29828.26194279
##  27688.40210294 26546.10674283 36394.992253  ]
```

Le paramètre `n_components` peut faire l'objet d'une optimisation via GridSearchCV de scikit-learn.

Nous reconstruisons donc une Pipeline, sans spécifier de valeur a priori pour n_components.

```
# Reconstruction d'une Pipeline, sans spécifier de valeur
# a priori pour n_components
pipe2 = Pipeline([("famd", FAMD()),
                  ("ols", LinearRegression())])

# Paramétrage de la grille de paramètres
# Attention à l'étendue des valeurs possibles pour famd__n_components !!!
param = [{"famd__n_components": [x + 1 for x in range(12)]}]

# Construction de l'objet GridSearchCV
grid_search = GridSearchCV(pipe2,
                           param_grid=param,
                           scoring="neg_mean_squared_error",
                           cv=5,
                           verbose=0)
# Estimation du modèle
grid_search.fit(X, y)
```

```
## GridSearchCV(cv=5,
##              estimator=Pipeline(steps=[('famd', FAMD()),
##                                        ('ols', LinearRegression())]),
##              param_grid=[{'famd__n_components': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
##                                                  11, 12]}],
##              scoring='neg_mean_squared_error')
```

```
# Affichage du score optimal
grid_search.best_score_
```

```
## -49681609.976405315
```

```
# Affichage du RMSE optimal
import numpy as np
print(np.sqrt(-grid_search.best_score_))
```

```
## 7048.518282334615
```

```
# Affichage du paramètre optimal
grid_search.best_params_
```

```
## {'famd__n_components': 6}
```

```
# Prédiction sur l'échantillon de test
grid_search.predict(B)
```

```
## array([59166.75977397, 75330.61691695, 55918.63883543, 31786.01147151,
##        30135.88354508, 22455.79889542, 41006.95089521])
```

Pour plus d'informations sur l'AFDM sous scientisttools, consulter le notebook https://github.com/enfantbenidedieu/scientisttools/blob/master/notebooks/famd_example.ipynb.