Analyse exploratoire multidimensionnelle des données

Application sous Python avec scientisttools 0.1.6

Duvérier DJIFACK ZEBAZE

6 Analyse en Composantes Principales Mixte

Sommaire

Ce chapitre a pour objectif de présenter rapidement les principales fonctionnalités offertes par le package « scientisttools » pour réaliser une Analyse en Composantes Principales Mixte (MPCA, *Mixed Principal Component Analysis*)

6.1 Données et problématique

6.1.1 Présentation des données

L'analyse en composantes principales mixte traite les tableaux individus-variables, lesquelles sont composées d'un mix de quantitatives et qualitatives. Elle a été développée par Abdesselam (2006). Nous illustrons cette méthode à l'aide des données de l'article et qui sont disponibles dans scientisttools.

```
# Chargement des données
from scientisttools import load_carsacpm
cars = load_carsacpm()
cars.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## Index: 27 entries, AS2 to VW3
## Data columns (total 9 columns):
##  #   Column  Non-Null Count  Dtype
## ---  ------  --------------  -----
##  0   CONS    27 non-null     float64
##  1   CYLI    27 non-null     int64
##  2   VITE    27 non-null     int64
##  3   VOLU    27 non-null     int64
##  4   RP/P    27 non-null     float64
```

```
##  5   LONG    27 non-null     float64
##  6   FISC    27 non-null     object
##  7   MARQ    27 non-null     object
##  8   PRIX    27 non-null     object
## dtypes: float64(3), int64(3), object(3)
## memory usage: 2.1+ KB
```

TABLE 6.1 - Données cars

	CONS	CYLI	VITE	VOLU	RP/P	LONG	FISC	MARQ	PRIX
AS2	6.2	998	140	955	23.2	3.40	4CV	ETRA	CP1
CI4	5.6	954	145	1170	19.4	3.50	4CV	FRAN	CP1
PE6	6.7	993	145	1151	20.8	3.61	4CV	FRAN	CP2
FI3	6.3	999	140	1088	21.8	3.64	4CV	ETRA	CP1
FI5	6.2	999	145	968	21.5	3.64	4CV	ETRA	CP2
FI8	8.9	1301	200	968	11.0	3.64	6CV	ETRA	CP4
FID	7.7	1302	165	968	16.0	3.64	6CV	ETRA	CP3
FO1	7.0	1117	137	900	22.7	3.64	4CV	ETRA	CP1
RE7	9.3	1597	180	973	12.0	3.64	6CV	FRAN	CP4
NI1	6.4	988	140	375	17.0	3.64	4CV	ETRA	CP1
OP1	7.2	993	143	845	22.4	3.62	4CV	ETRA	CP1
PE1	6.8	954	134	1200	23.8	3.70	4CV	FRAN	CP1
PE3	5.8	1124	142	1200	21.4	3.70	5CV	FRAN	CP3
DA2	9.2	1360	170	1200	13.9	3.70	6CV	ETRA	CP3
PE9	8.7	1580	190	1200	11.2	3.70	6CV	FRAN	CP4
RE1	6.3	956	115	950	33.1	3.67	4CV	FRAN	CP1
RE3	6.3	1108	120	950	28.4	3.67	5CV	FRAN	CP2
RE4	5.8	1108	143	915	20.6	3.59	5CV	FRAN	CP2
FO9	7.9	1397	167	915	13.8	3.59	6CV	ETRA	CP3
RE8	8.7	1397	200	915	10.2	3.59	6CV	FRAN	CP4
SE4	8.8	1461	175	1200	14.7	3.63	6CV	ETRA	CP3
SE9	7.3	903	131	1088	23.4	3.46	4CV	ETRA	CP1
SZ2	6.4	993	145	400	18.4	3.58	4CV	ETRA	CP1
SZ3	6.5	1324	163	400	14.0	3.58	5CV	ETRA	CP3
TO1	6.1	999	150	202	19.5	3.70	4CV	ETRA	CP2
TO3	6.8	1295	170	202	15.0	3.70	5CV	ETRA	CP3
VW3	7.8	1272	170	1040	14.3	3.65	6CV	ETRA	CP3

Il s'agit d'un échantillon de 27 petites voitures du marché belge. Ces voitures sont décrites par 9 variables dont 6 sont quantitatives : la consommation urbaine (*CONS*),la cylindrée (*CYLI*), la vitesse maximale (*VITE*), le volume du coffre (*VOLU*), le rapport pois/puissance (*RP/P*) et la longueur (*LONG*) ; et 3 sont qualitatives : la puissance fiscale (4CV, 5CV, 6CV), la marque du constructeur (Française, Etrangère) et la classe de prix (CP1, CP2, CP3, CP4).

6.1.2 Problématique

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.)

6.2 Analyse en composantes principales mixte

6.2.1 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.

```
from scientisttools import MPCA
```

6.2.2 Individus et variables actifs

On crée une instance de la classe `MPCA`.

```
# Instanciation
my_mpca = MPCA()
```

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

```
# Estimation du modèle
my_mpca.fit(cars)
```

```
## MPCA()
```

6.2.2.1 Les valeurs propres

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

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

```
##         eigenvalue  difference  proportion  cumulative
## Dim.1     6.153650    3.397784   41.024332   41.024332
## Dim.2     2.755866    0.407014   18.372438   59.396770
## Dim.3     2.348852    1.139269   15.659013   75.055783
## Dim.4     1.209583    0.272588    8.063886   83.119668
## Dim.5     0.936995    0.128760    6.246630   89.366299
## Dim.6     0.808235    0.500608    5.388230   94.754529
## Dim.7     0.307627    0.097812    2.050846   96.805375
## Dim.8     0.209815    0.111817    1.398765   98.204140
## Dim.9     0.097998    0.021374    0.653320   98.857460
## Dim.10    0.076624    0.004898    0.510826   99.368286
## Dim.11    0.071726    0.048695    0.478174   99.846461
## Dim.12    0.023031         NaN    0.153539  100.000000
```

L'attribut `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 import get_eig
print(get_eig(my_mpca))
```

```
##          eigenvalue  difference  proportion  cumulative
## Dim.1      6.153650    3.397784   41.024332   41.024332
## Dim.2      2.755866    0.407014   18.372438   59.396770
## Dim.3      2.348852    1.139269   15.659013   75.055783
## Dim.4      1.209583    0.272588    8.063886   83.119668
## Dim.5      0.936995    0.128760    6.246630   89.366299
## Dim.6      0.808235    0.500608    5.388230   94.754529
## Dim.7      0.307627    0.097812    2.050846   96.805375
## Dim.8      0.209815    0.111817    1.398765   98.204140
## Dim.9      0.097998    0.021374    0.653320   98.857460
## Dim.10     0.076624    0.004898    0.510826   99.368286
## Dim.11     0.071726    0.048695    0.478174   99.846461
## Dim.12     0.023031         NaN    0.153539  100.000000
```

Les valeurs propres peuvent être représentées graphiquement :

```
# Ebouli des valeurs propres (=Scree plot)
from scientisttools import fviz_screeplot
print(fviz_screeplot(my_mpca,choice="eigenvalue"))
```

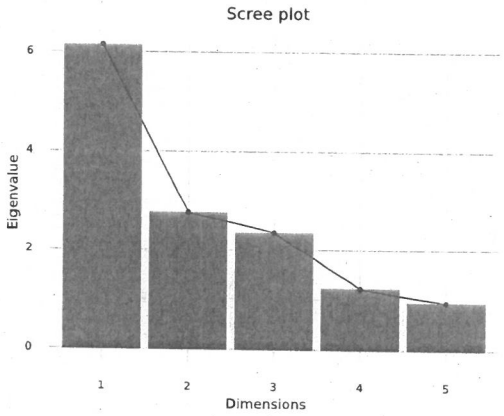


FIGURE 6.1 – Ebouli des valeurs propres

```
# Ebouli des proportions
print(fviz_screeplot(my_mpca,choice="proportion"))
```

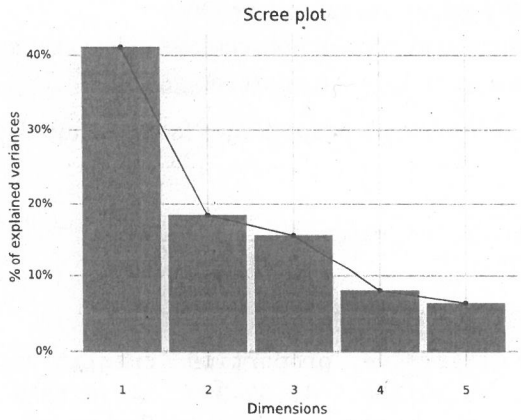


FIGURE 6.2 – Ebouli des proportions

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

```
from scientisttools import summaryMPCA
summaryMPCA(my_mpca)
```

```
##                 Mixed Principal Component Analysis - Results
##
## Importance of components
##                            Dim.1   Dim.2   Dim.3   Dim.4   Dim.5
## Variance                   6.154   2.756   2.349   1.210   0.937
## Difference                 3.398   0.407   1.139   0.273   0.129
## % of var.                 41.024  18.372  15.659   8.064   6.247
## Cumulative of % of var.   41.024  59.397  75.056  83.120  89.366
##
## Individuals (the 10 first)
##
##      Weight  Sq. Dist.  Inertia  Dim.1  ...   cos2  Dim.3    ctr   cos2
## AS2   0.037     17.530    0.649 -2.627  ...  0.165  1.404  3.108  0.112
## CI4   0.037     14.563    0.539 -2.397  ...  0.064  1.758  4.876  0.212
## PE6   0.037     12.707    0.471 -1.631  ...  0.397  0.330  0.171  0.009
## FI3   0.037      7.926    0.294 -2.087  ...  0.153  0.838  1.107  0.089
## FI5   0.037     10.489    0.388 -1.770  ...  0.002 -0.580  0.531  0.032
## FI8   0.037     20.677    0.766  3.766  ...  0.021  1.355  2.896  0.089
## FID   0.037      8.978    0.333  2.161  ...  0.180 -0.715  0.807  0.057
## FO1   0.037      6.807    0.252 -1.796  ...  0.224  0.752  0.893  0.083
## RE7   0.037     24.727    0.916  4.003  ...  0.133  1.850  5.398  0.138
## NI1   0.037     10.308    0.382 -1.913  ...  0.302  0.131  0.027  0.002
##
## [10 rows x 12 columns]
##
## Continuous variables
##
##       Dim.1     ctr   cos2  Dim.2    ctr   cos2  Dim.3    ctr   cos2
## CONS  0.859  11.991  0.738 -0.118  0.502  0.014  0.346  5.094  0.120
## CYLI  0.950  14.680  0.903  0.068  0.166  0.005 -0.059  0.150  0.004
## VITE  0.928  14.005  0.862 -0.101  0.372  0.010  0.103  0.453  0.011
```

```
## VOLU  0.152   0.375  0.023  0.375  5.112  0.141  0.477  9.668  0.227
## RP/P -0.865  12.150  0.748  0.223  1.806  0.050 -0.010  0.004  0.000
## LONG  0.292   1.386  0.085  0.270  2.639  0.073 -0.335  4.775  0.112
##
## Categories
##
##        Dim.1     ctr   cos2  vtest  Dim.2  ...  vtest  Dim.3     ctr   cos2  vtest
## 4CV   -0.830  11.184  0.688 -1.643 -0.226  ... -0.668  0.400   6.827  0.160  1.284
## 5CV   -0.044   0.032  0.002 -0.043  0.378  ...  0.553 -0.791  26.668  0.626 -1.255
## 6CV    0.916  13.630  0.839  1.331 -0.072  ... -0.157  0.228   2.208  0.052  0.536
## ETRA   0.023   0.009  0.001  0.061 -0.946  ... -3.788 -0.195   1.625  0.038 -0.847
## FRAN  -0.023   0.009  0.001 -0.036  0.946  ...  2.228  0.195   1.625  0.038  0.498
## CP1   -0.702   8.017  0.493 -1.107 -0.302  ... -0.712  0.505  10.841  0.255  1.288
## CP2   -0.307   1.533  0.094 -0.301  0.451  ...  0.660 -0.362   5.577  0.131 -0.574
## CP3    0.490   3.901  0.240  0.653 -0.299  ... -0.597 -0.593  14.986  0.352 -1.281
## CP4    0.661   7.098  0.437  0.567  0.303  ...  0.388  0.472   9.501  0.223  0.655
##
## [9 rows x 12 columns]
##
## Categoricals variables (eta2)
##
##        Dim.1  Dim.2  Dim.3
## FISC   0.918  0.146  0.628
## MARQ   0.001  0.895  0.038
## PRIX   0.929  0.364  0.705
```

6.2.3 Représentation graphique

6.2.3.1 Nuage des individus

```
# Carte des individus
from scientisttools import fviz_mpca_ind
print(fviz_mpca_ind(my_mpca,repel=True))
```

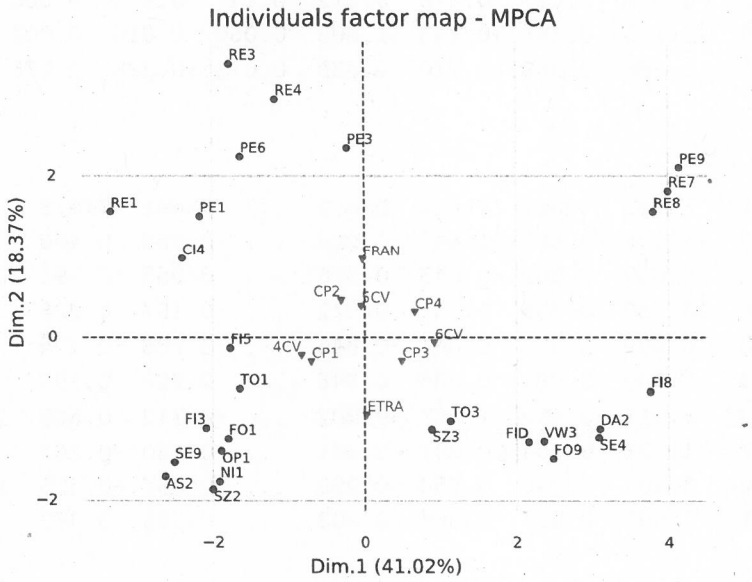


FIGURE 6.3 – Nuage des individus

6.2.3.2 Nuage des modalités

```
# Carte des modalités - variables qualitatives
from scientisttools import fviz_mpca_mod
print(fviz_mpca_mod(my_mpca,repel=True))
```

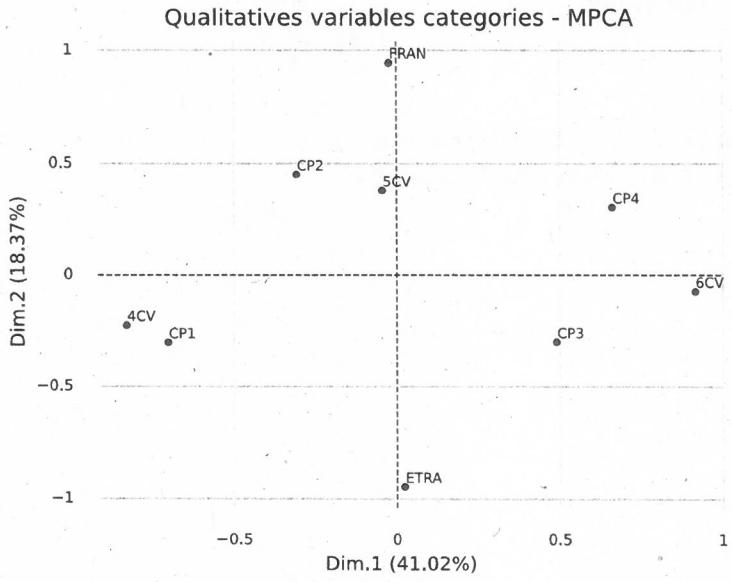


FIGURE 6.4 – Nuage des modalités

6.2.3.3 Cercle des corrélations

```
# Cercle des corrélations - variables quantitatives
from scientisttools import fviz_mpca_col
print(fviz_mpca_col(my_mpca))
```

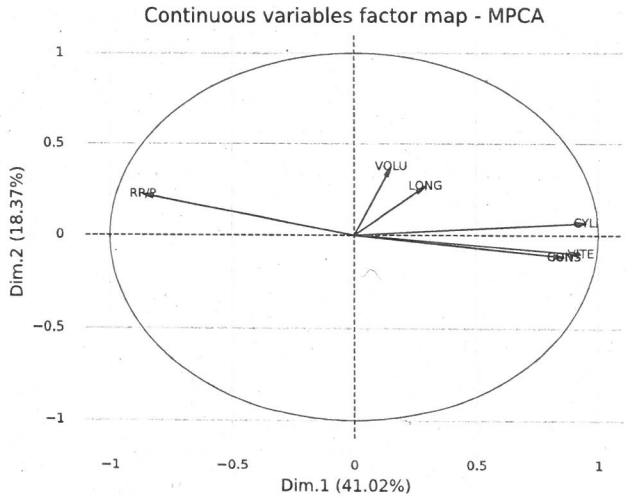


FIGURE 6.5 – Cercle des corrélations

6.2.3.4 Graphe des variables

```
# Carte des variables - rapport de corrélation et cosinus carré
from scientisttools import fviz_mpca_var
print(fviz_mpca_var(my_mpca,repel=True))
```

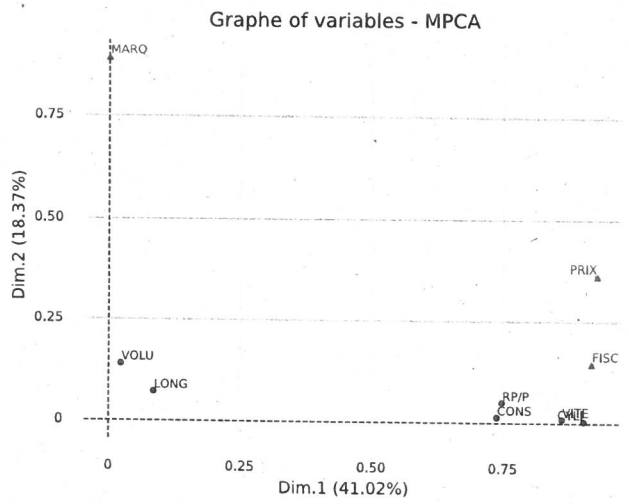


FIGURE 6.6 – Graphe des variables

6.2.4 Eléments supplémentaires

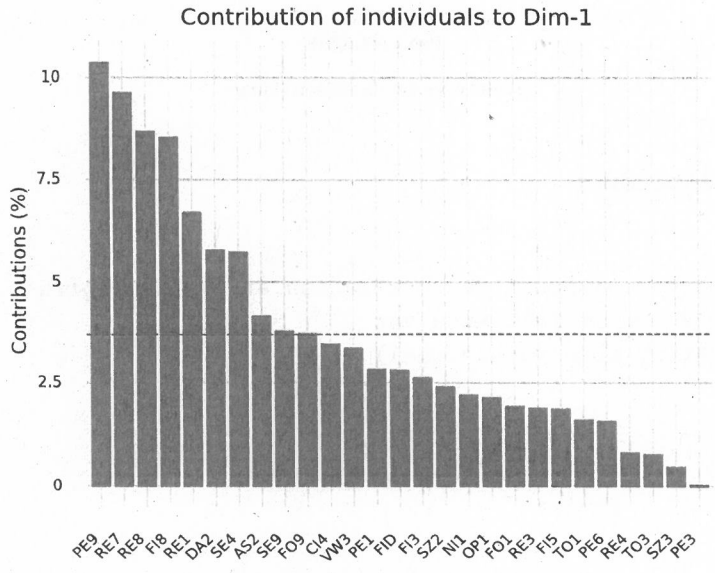
Les individus illustratifs et les variables illustratives n'influencent pas la construction des composantes principales de l'analyse. Ils/Elles aident à l'interprétation des dimensions de variabilité. On peut ajouter les éléments supplémentaires à l'aide des paramètres :

- `ind_sup` pour les individus supplémentaires
- `quanti_sup` pour les variables supplémentaires quantitatives
- `quali_sup` pour les variables supplémentaires qualitatives

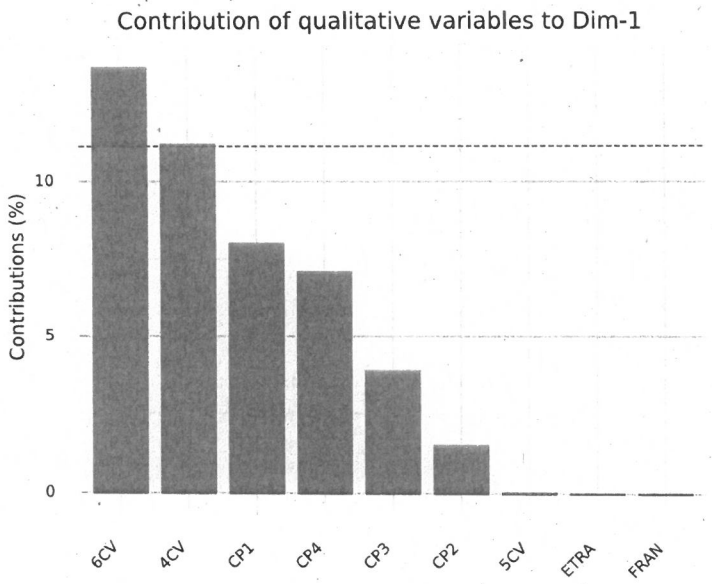
6.2.5 Interprétation des axes

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.

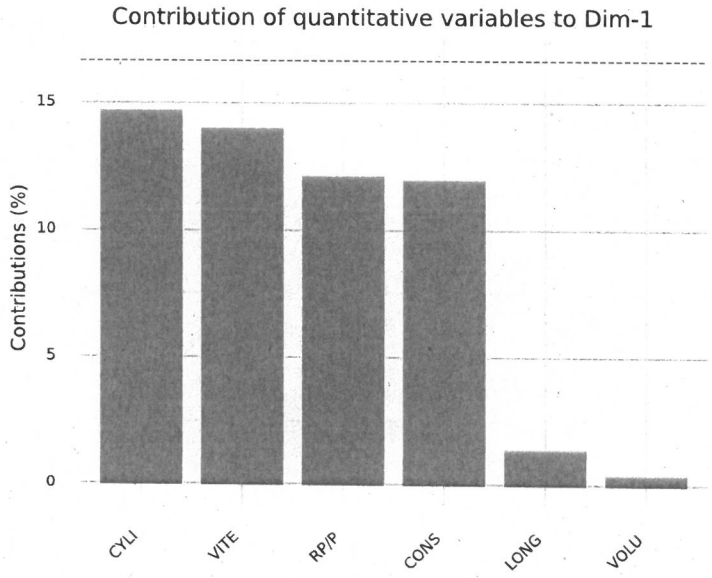
```
# Classement des points lignes en fonction de leur contribution au 1er axe
from scientisttools import fviz_contrib, fviz_cos2
print(fviz_contrib(my_mpca,choice="ind"))
```



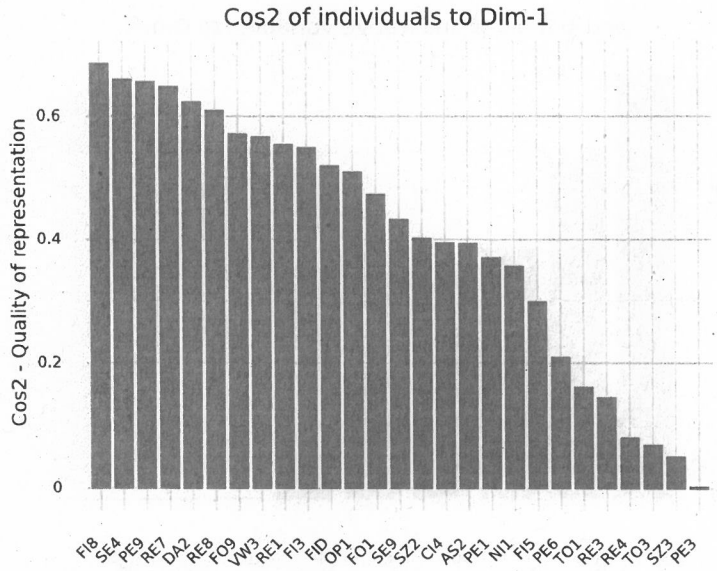

```
# Classement des modalités en fonction de leur contribution au 1er axe
print(fviz_contrib(my_mpca,choice="quali_var"))
```



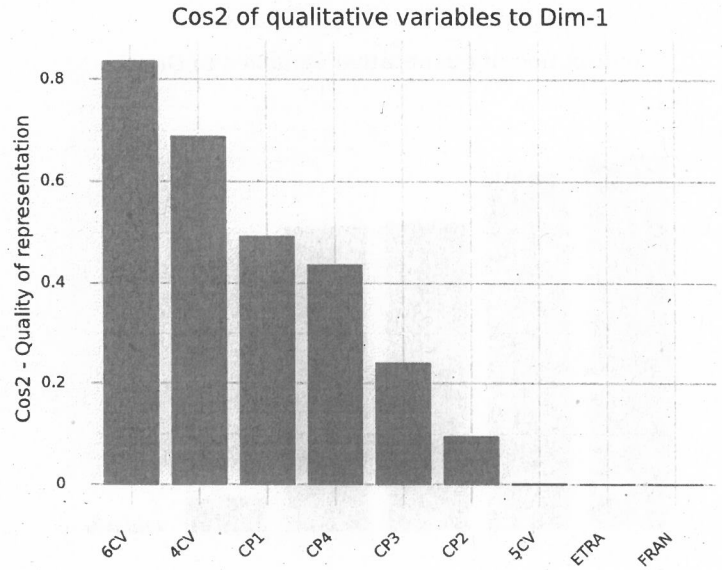

```
# Classement des variables quantitatives
# en fonction de leur contribution au 1er axe
print(fviz_contrib(my_mpca,choice="quanti_var"))
```



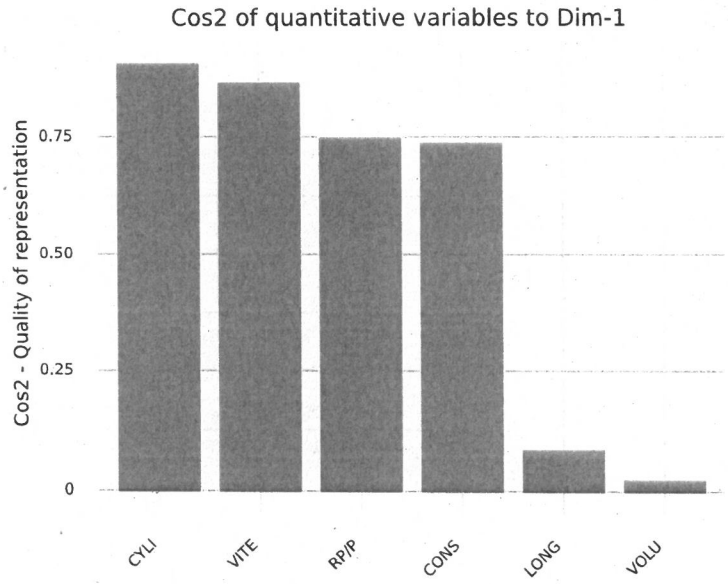

```
# Classement des individus en fonction de leur cos2 sur le 1er axe
print(fviz_cos2(my_mpca,choice="ind"))
```




```
# Classement des modalités en fonction de leur cos2 sur le 1er axe
print(fviz_cos2(my_mpca,choice="quali_var"))
```




```
# Classement des variables quantitatives
# en fonction de leur cos2 sur le 1er axe
print(fviz_cos2(my_mpca,choice="quanti_var"))
```



6.2.6 Description des axes

On peut décrire les dimensions données par les variables en calculant le ratio de corrélation (lorsque les variables sont qualitatives) ou le coefficient de corrélation (lorsque les variables sont quantitatives) entre une variable et une dimension et en réalisant un test de significativité.

```
from scientisttools import dimdesc
dim_desc = dimdesc(my_mpca)
dim_desc.keys()
```

```
## dict_keys(['Dim.1', 'Dim.2', 'Dim.3', 'Dim.4', 'Dim.5'])
```

```
dim_desc["Dim.1"].keys()
```

```
## dict_keys(['quanti', 'quali', 'category'])
```

```
dim_desc["Dim.1"]["quali"]
```

```
##                R2        pvalue
## PRIX  0.928500  2.548496e-13
## FISC  0.917614  9.777230e-14
```

```
dim_desc["Dim.1"]["quanti"]
```

```
##       correlation        pvalue
## CYLI     0.950441  3.419487e-14
## VITE     0.928352  3.042899e-12
## CONS     0.859012  9.770004e-09
## RP/P    -0.864684  6.038377e-09
```

6.3 Approche Machine Learning

Ici, l'objectif est d'utiliser l'analyse en composantes principales mixte en tant que méthode de pré-traitement.

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

```
my_mpca.transform(cars).head(6)
```

```
##         Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
## AS2 -2.626868 -1.700462  1.403855  0.374420 -1.431409
## CI4 -2.397295  0.966126  1.758490  0.827966 -1.324621
## PE6 -1.631235  2.247254  0.329646 -0.568445  1.283136
## FI3 -2.086566 -1.102789  0.837737  0.359289  0.400727
## FI5 -1.769700 -0.132189 -0.580175 -1.282820  1.868576
## FI8  3.766185 -0.661957  1.355248 -1.455975 -0.049329
```

6.3.1 Intégration dans une Pipeline de scikit-learn

La classe `MPCA` peut être intégrée dans une Pipeline de scikit-learn. Dans le cadre de notre exemple, nous cherchons à prédire la variable (variable "*CONS*") à partir des 8 autres variables du jeu de données (données actives).

"*CONS*" 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 en composantes principales mixte 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 = cars.drop(columns=["CONS"])
# y = target
y = cars["CONS"]

# Construction de la Pipeline
# On enchaine une analyse en composantes principales mixte (4 axes retenus)
# puis une régression linéaire
pipe = Pipeline([("mpca", MPCA(n_components=4)),
                 ("ols", LinearRegression())])
# Estimation du modèle
pipe.fit(X, y)
```

```
## Pipeline(steps=[('mpca', MPCA(n_components=4)), ('ols', LinearRegression())])
```

On prédit

```
# Prédiction sur l'échantillon d'apprentissage
print(pipe.predict(X))
```

```
## [6.71620282 6.82199628 6.34827361 6.69648777 6.30370296 8.87560171
##  7.9433029  6.66503076 8.84909623 6.57615455 6.61915448 6.50571707
##  6.5606888  8.15440488 9.00263606 5.97391966 5.59921646 6.06740429
##  8.10747797 8.92534366 8.28423609 6.57869009 6.617008   6.94640567
##  6.09981201 6.82381344 8.03822177]
```

6.3.2 GridsearchCV

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([("mpca", MPCA()),
                  ("ols", LinearRegression())])

# Paramétrage de la grille de paramètres
# Attention à l'étendue des valeurs possibles pour mpca__n_components !!!
param = [{"mpca__n_components": [x + 1 for x in range(my_mpca.eig_.shape[0])]}]

# 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=[('mpca', MPCA()),
##                                        ('ols', LinearRegression())]),
##              param_grid=[{'mpca__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_
```

```
## -0.2919876621901945
```

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

```
## 0.5403588272529602

# Affichage du paramètre optimal
grid_search.best_params_

## {'mpca__n_components': 7}

# Prédiction sur les individus supplémentaires
grid_search.predict(X)

## array([6.41894023, 6.20912462, 6.22384494, 7.04288543, 6.7384921 ,
##        9.22599728, 8.03972865, 6.96252269, 8.76212712, 6.41433597,
##        6.79737488, 6.57343749, 6.47707795, 8.45997212, 9.13727856,
##        6.14813001, 5.94774833, 5.9133333 , 7.99298653, 8.55345671,
##        8.43430936, 6.51946179, 6.3428685 , 6.46019451, 6.22175664,
##        6.54986542, 8.13274887])
```

6.4 Les fonctions predictMPCA et supvarMPCA

La version 0.1.6 de scientisttools innove en proposant deux fonctions supplémentaires à la fonction MPCA : predictMPCA et supvarMPCA.

6.4.1 La fonction predictMPCA

La fonction predictMPCA permet la projection des individus supplémentaires dans l'espace des composantes principales. Contrairement à la fonction transform, predictMPCA calcule également la qualité de la représentation (=cos2) et le carré de la distance des individus à l'origine.

Appliquons la fonction predictMPCA sur les données actives :

```
# function predictPCAMIX
from scientisttools import predictMPCA
predict = predictMPCA(my_mpca,X=cars)
predict.keys()

## dict_keys(['coord', 'cos2', 'dist'])
```

Comparons nos résultats avec ceux de la fonction PCAMIX

— **Coordonnées factorielles**

```
# Coordonnées factorielles avec la fonction MPCA
my_mpca.ind_["coord"].head(6)

##         Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
## AS2 -2.626868 -1.700462  1.403855  0.374420 -1.431409
## CI4 -2.397295  0.966126  1.758490  0.827966 -1.324621
## PE6 -1.631235  2.247254  0.329646 -0.568445  1.283136
```

```
## FI3 -2.086566 -1.102789  0.837737  0.359289  0.400727
## FI5 -1.769700 -0.132189 -0.580175 -1.282820  1.868576
## FI8  3.766185 -0.661957  1.355248 -1.455975 -0.049329
```

```
# Coordonnées factorielles avec la fonction predictMPCA
predict["coord"].head(6)
```

```
##         Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
## AS2 -2.626868 -1.700462  1.403855  0.374420 -1.431409
## CI4 -2.397295  0.966126  1.758490  0.827966 -1.324621
## PE6 -1.631235  2.247254  0.329646 -0.568445  1.283136
## FI3 -2.086566 -1.102789  0.837737  0.359289  0.400727
## FI5 -1.769700 -0.132189 -0.580175 -1.282820  1.868576
## FI8  3.766185 -0.661957  1.355248 -1.455975 -0.049329
```

— **Cosinus carré**

```
# Cosinus carré avec la fonction MPCA
my_mpca.ind_["cos2"].head(6)
```

```
##         Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
## AS2  0.393628  0.164946  0.112423  0.007997  0.116879
## CI4  0.394645  0.064096  0.212346  0.047075  0.120489
## PE6  0.209410  0.397437  0.008552  0.025430  0.129572
## FI3  0.549312  0.153441  0.088546  0.016287  0.020261
## FI5  0.298589  0.001666  0.032092  0.156894  0.332887
## FI8  0.685984  0.021192  0.088828  0.102522  0.000118
```

```
# Cosinus carré avec la fonction predictMPCA
predict["cos2"].head(6)
```

```
##         Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
## AS2  0.393628  0.164946  0.112423  0.007997  0.116879
## CI4  0.394645  0.064096  0.212346  0.047075  0.120489
## PE6  0.209410  0.397437  0.008552  0.025430  0.129572
## FI3  0.549312  0.153441  0.088546  0.016287  0.020261
## FI5  0.298589  0.001666  0.032092  0.156894  0.332887
## FI8  0.685984  0.021192  0.088828  0.102522  0.000118
```

— **Carré de la distance à l'origine**

```
# Carré de la distance à l'origine avec la fonction MPCA
my_mpca.ind_["infos"]["Sq. Dist."].head(6)
```

```
## AS2    17.530355
## CI4    14.562517
## PE6    12.706787
## FI3     7.925836
## FI5    10.488785
## FI8    20.677079
## Name: Sq. Dist., dtype: float64
```

```
# Carré de la distance à l'origine avec la fonction predictMPCA
predict["dist"].head(6)
```

```
## AS2     17.530355
## CI4     14.562517
## PE6     12.706787
## FI3      7.925836
## FI5     10.488785
## FI8     20.677079
## Name: Sq. Dist., dtype: float64
```

Les résultats sont identiques.

6.4.2 La fonction supvarMPCA

La fonction supvarMPCA permet la projection des variables supplémentaires (continues et qualitatives).

```
# Extraction des variables supplémentaires
from scientisttools import splitmix, supvarMPCA
X_quanti_sup = splitmix(cars)["quanti"]
X_quali_sup = splitmix(cars)["quali"]
supvar = supvarMPCA(my_mpca,X_quanti_sup=X_quanti_sup,X_quali_sup=X_quali_sup)
supvar.keys()
```

```
## dict_keys(['quanti', 'quali'])
```

6.4.2.1 Variables quantitatives supplémentaires

Nous comparons les résultats issus de la fonction supvarMPCA avec ceux de la fonction MPCA.

```
supvar["quanti"].keys()
```

```
## dict_keys(['coord', 'cor', 'cos2'])
```

— **Coordonnées factorielles**

```
# Coordonnées factorielles avec la fonction MPCA
my_mpca.quanti_var_["coord"].head(6)
```

```
##           Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
## CONS   0.859012 -0.117609  0.345895  0.036224  0.167043
## CYLI   0.950441  0.067598 -0.059370  0.029615 -0.073148
## VITE   0.928352 -0.101186  0.103113 -0.225809 -0.101447
## VOLU   0.151948  0.375336  0.476545  0.618212  0.279747
## RP/P  -0.864684  0.223102 -0.009520  0.234878  0.166710
## LONG   0.292079  0.269671 -0.334891 -0.069933  0.506087
```

```
# Coordonnées factorielles avec la fonction supvarMPCA
supvar["quanti"]["coord"].head(6)
```

```
##          Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
## CONS  0.859012 -0.117609  0.345895  0.036224  0.167043
## CYLI  0.950441  0.067598 -0.059370  0.029615 -0.073148
## VITE  0.928352 -0.101186  0.103113 -0.225809 -0.101447
## VOLU  0.151948  0.375336  0.476545  0.618212  0.279747
## RP/P -0.864684  0.223102 -0.009520  0.234878  0.166710
## LONG  0.292079  0.269671 -0.334891 -0.069933  0.506087
```

— Cosinus carré

```
# Cosinus carré avec la fonction MPCA
my_mpca.quanti_var_["cos2"].head(6)
```

```
##          Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
## CONS  0.737901  0.013832  0.119644  0.001312  0.027903
## CYLI  0.903338  0.004569  0.003525  0.000877  0.005351
## VITE  0.861837  0.010239  0.010632  0.050990  0.010292
## VOLU  0.023088  0.140877  0.227095  0.382186  0.078259
## RP/P  0.747678  0.049774  0.000091  0.055168  0.027792
## LONG  0.085310  0.072722  0.112152  0.004891  0.256124
```

```
# Cosinus carré avec la fonction supvarMPCA
supvar["quanti"]["cos2"].head(6)
```

```
##          Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
## CONS  0.737901  0.013832  0.119644  0.001312  0.027903
## CYLI  0.903338  0.004569  0.003525  0.000877  0.005351
## VITE  0.861837  0.010239  0.010632  0.050990  0.010292
## VOLU  0.023088  0.140877  0.227095  0.382186  0.078259
## RP/P  0.747678  0.049774  0.000091  0.055168  0.027792
## LONG  0.085310  0.072722  0.112152  0.004891  0.256124
```

Tous les résultats sont identiques.

6.4.2.2 Variables supplémentaires qualitatives

Nous comparons également les résultats issus de la fonction `supvarMPCA` avec ceux de la fonction `MPCA`.

```
supvar["quali"].keys()
```

```
## dict_keys(['coord', 'cos2', 'vtest', 'dist', 'eta2'])
```

— Coordonnées factorielles

```
# Coordonnées factorielles avec la fonction MPCA
my_mpca.quali_var_["coord"].head(6)
```

```
##          Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
## 4CV  -0.829601 -0.225533  0.400444 -0.169944  0.139799
## 5CV  -0.044295  0.377974 -0.791452  0.069307 -0.400827
## 6CV   0.915821 -0.072409  0.227730  0.123018  0.182113
## ETRA  0.022882 -0.945880 -0.195347 -0.095776  0.140474
## FRAN -0.022882  0.945880  0.195347  0.095776 -0.140474
## CP1  -0.702382 -0.302027  0.504614  0.154892 -0.221186
```

```
# Coordonnées factorielles avec la fonction supvarMPCA
supvar["quali"]["coord"].head(6)
```

```
##          Dim.1     Dim.2     Dim.3     Dim.4     Dim.5
## 4CV  -2.135640 -0.388536  0.636887 -0.193961  0.140432
## 5CV  -0.230486  1.316185 -2.544361  0.159890 -0.813865
## 6CV   3.212860 -0.169995  0.493586  0.191338  0.249301
## ETRA  0.043535 -1.204317 -0.229620 -0.080788  0.104289
## FRAN -0.074009  2.047339  0.390355  0.137340 -0.177292
## CP1  -2.271768 -0.653732  1.008350  0.222111 -0.279158
```

Les coordonnées ne sont pas identiques car la fonction `supvarMPCA` détermine les coordonnées des modalités sous forme de barycentre des individus qui la possèdent.

Pour plus d'informations sur MPCA sous scientisttools, consulter le notebook mpca_gironde.ipynb.