

Pratique des Méthodes Factorielles avec Python

Ricco Rakotomalala

To cite this version:

Ricco Rakotomalala. Pratique des Méthodes Factorielles avec Python. 2020. ⟨hal-04868625⟩

HAL Id: hal-04868625

https://hal.science/hal-04868625v1

Submitted on 6 Jan 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons CC BY-NC-ND 4.0 - Attribution - Non-commercial use - No Derivative Works - International License

Ricco Rakotomalala

Pratique des Méthodes Factorielles avec Python

Version 1.0

Université Lumière Lyon 2

Avant-propos

Mes premiers contacts avec les méthodes factorielles datent de la deuxième moitié des années 80. J'étais subjugué par ce mélange de statistique, de mathématiques appliquées et d'informatique que je nourrissais comme je pouvais avec les lectures que j'avais à disposition, principalement les livres scientifiques des Editions Mir. J'étais intrigué par les statistiques multidimensionnelles dont il est fait mention dans l'excellentissime livre d'Aïvazian, Enukov et Mekhalkine (« Eléments de modélisation et traitement primaire des données », 1986 ; section 10.5, « Visualisation des données multidimensionnelles »). Avec mon recul aujourd'hui, je vois bien que les auteurs parlent de l'analyse en composantes principales et du positionnement multidimensionnel. Mais faute d'exemples didactiques, faute d'illustrations des fameuses représentations graphiques qui en font tout le charme justement, je n'étais absolument pas en mesure d'en apprécier la teneur. Dommage, dommage, je me dis qu'il aurait été « fun » de (tenter de) coder en Basic 1.0 ces méthodes sur mon antique Thomson MO5 qui a accompagné ma découverte des statistiques à l'époque, et avec lequel – malgré ses limitations – j'ai passé des heures et des heures à programmer moult méthodes numériques.

Pour la petite histoire, je suis passé à un Amstrad CPC 6128 ensuite. Grand luxe. Contrairement à la Thomson qu'il fallait brancher sur la télé avec une prise péritel (la prise quoi ?), l'Amstrad possédait son propre moniteur (ce qui a aplani bien des énervements autour de la télé familiale) et un lecteur de disquette (non-standard, 3 pouces), un progrès énorme par rapport à l'antique lecteur de cassettes (lecteur de quoi ?) qui mettait des heures à charger les progiciels de quelques kilo-octets (ah bon, il y avait une unité kilo pour les octets ?). Je peux vous dire que j'ai beaucoup donné avec l'Amstrad aussi. Et comme les délestages n'étaient pas légion encore (ceux qui le vivent savent ce que c'est), je pouvais y passer des jours et des nuits sans restriction. Tout ce que j'ai appris de cette époque - rationaliser l'écriture du code, gratter les millisecondes, minimiser l'occupation mémoire, etc. - me sert encore aujourd'hui.

Pour en revenir aux méthodes factorielles, j'ai eu accès à la même période à un autre excellent ouvrage (pour l'économétrie) de Christian Labrousse (« Introduction à l'économétrie », Dunod, 1983). Il présente l'analyse en composantes principales (chapitre 7) et l'analyse factorielle des correspondances (chapitre 8). J'avoue que ça a été compliqué. Le féru d'économétrie que j'étais, qui ne carburait qu'à la somme des carrés des résidus et au maximum de vraisemblance, a été complètement dérouté par cette histoire de régression sans variable endogène, où les erreurs seraient comptabilisées orthogonalement. Il faut dire aussi que l'auteur n'a pas été tendre avec une présentation assez austère sans aucun exemple illustratif, et en expliquant en conclusion du chapitre 7 que ces méthodes s'avéraient souvent décevantes car : soit elles ne proposaient que des solutions évidentes ; soit la lecture des résultats reposait quasi entièrement sur les connaissances du domaine, exogènes aux données (page 108).

J'en étais resté à ces idées plutôt mitigées lorsque, dans la première moitié des années 90, deux événements majeurs m'ont ouvert les yeux, et m'ont amené à considérer sous un tout autre angle les techniques factorielles en général. J'ai découvert à la bibliothèque d'excellents ouvrages, ceux d'Escofier et Pagès (« Analyses factorielles simples et multiples », Dunod, 1988) et de Volle (« Analyse des données », Economica, 1985) qui présentent les méthodes factorielles sous un angle autrement plus sympathique. Ma perception globale des statistiques et de l'analyse exploratoire a connu un tournant décisif lorsque j'ai eu accès à l'édition de 1990 du livre de Gilbert Saporta (« Probabilités - Analyse des données et Statistique », Technip). Et, dans le même temps, j'ai découvert les logiciels SPAD.N, CHADOC (Chaîne d'Analyse de Données à Organisation Cohérente), et STATITCF dans les salles de libre accès de l'Université. Enfin, je pouvais toucher du doigt ce qu'on pouvait obtenir avec les analyses factorielles en les mettant en œuvre sur des vrais fichiers de données. Enfin, je pouvais générer moi-même ces fameux graphiques, en mode caractères certes, mais c'était les miens, à partir des bases de mon cru. Ça peut paraître extraordinaire aujourd'hui, mais savoir importer ses propres données sur ce type de logiciels était déjà une compétence en soi. Ces expériences ont été fondamentales. On ne peut vraiment comprendre les méthodes statistiques que si on les pratique.

Depuis, je n'ai eu cesse d'explorer le domaine, en lisant d'autres ouvrages – pour changer, les meilleurs sont en français, j'ai mis ceux dont j'ai apprécié la lecture en bibliographie – en testant

d'autres logiciels, en développant mes propres implémentations dans TANAGRA. Il y a quelques années, je m'étais dit qu'il fallait absolument capitaliser cette expertise accumulée en rédigeant une collection de supports et de tutoriels réunis sur mon blog (http://tutoriels-data-mining.blogspot.com/), alors que mes enseignements sont plutôt portés vers le « machine learning », les méthodes prédictives, et leurs applications (text mining, web mining, etc.). Mon idée était à terme d'écrire un fascicule de cours dédié à la pratique de l'analyse factorielle. Malheureusement, faute de disposer d'une plage de quiétude suffisamment étendue, ça n'a pas été possible.... jusqu'à cette période où, grâce ou à cause de circonstances que nous connaissons tous, j'ai la possibilité de travailler relativement tranquillement.

Cet ouvrage fait la synthèse dans un ensemble que j'espère cohérent ce que j'ai pu écrire sur les méthodes factorielles ces quinze dernières années. Le plan est relativement classique. Nous présentons tour à tour les différentes techniques qui font référence dans le domaine (ACP, MDS, AFC, ACM, AFDM). Par rapport à la très abondante littérature qui existe (tant en ouvrages qu'en supports accessibles en ligne), fidèle à mon habitude, je vais essayer de me démarquer en évitant déjà d'abreuver le lecteur de succession de formules matricielles arides, en simplifiant la présentation des techniques (ce n'est pas le plus facile), en retraçant dans le détail les aspects calculatoires à l'aide d'exemples illustratifs, en utilisant le langage Python et ses librairies scientifiques standards. Quelques requêtes Google semblent montrer (juin 2020) qu'il y a peu de documentation sur ce créneau, qui correspond pourtant à une véritable attente si j'en juge aux accès sur mon site des tutoriels où les documents relatifs aux méthodes factorielles d'une part, à la pratique de la data science sous Python d'autre part, font partie de ceux qui sont le plus téléchargés. A chaque fois que cela est nécessaire, nous validerons les résultats en les comparant à ceux de SAS (proc princomp, proc factor, proc corresp), R (factominer, ade4, ca, pcamixdata, psych) et TANAGRA (avec les composants de l'onglet « Factorial Analysis »).

Voilà. J'espère que ce fascicule, entièrement libre comme toute ma production scientifique, permettra aux fans de data science, aux étudiants en quête de connaissances en particulier, de progresser dans l'apprentissage des méthodes de l'exploration statistique des données.

Lyon, le 19 juillet 2020.

Table des matières

1 Analyse en composantes principales (ACP) 9
1.1 Principe de l'analyse en composantes principales 10
1.2 Organisations des calculs 37
1.3 Pratique de l'ACP avec « fanalysis » sous Python 44
1.4 Reconstitution des données 71
1.5 Projection des individus supplémentaires 74
1.6 Traitement des variables illustratives 80
1.7 ACP avec d'autres outils (SAS, TANAGRA, R) 90
2 Plus loin avec l'analyse en composantes principales 108
2.1 Techniques avancées pour identifier le nombre de facteurs 108
2.2 Rotation des facteurs – Rotation VARIMAX 120
2.3 Indices de compressibilité de l'information 128
2.4 ACP sur corrélations partielles 135
2.5 Clustering de variables – VARCLUS 146
2.6 Analyse en facteurs principaux 161
3 Positionnement multidimensionnel (MDS) 188
3.1 Position du problème 188
3.2 Positionnement multidimensionnel classique 191
3.3 Traitement des individus supplémentaires 203
3.4 Positionnement multidimensionnel classique et ACP 206
3.5 Plus loin avec le positionnement multidimensionnel 208
3.6 Un exemple – Distances routières entre villes de Madagascar 211
4 Analyse factorielle des correspondances (AFC) 218
4.1 Principe de l'analyse factorielle des correspondances 219
4.2 Organisation des calculs 251
4.3 Pratique de l'AFC avec « fanalysis » sous Python 255
4.4 Quelques formes caractéristiques 266

4.5 AFC avec d'autres outils (SAS, TANAGRA, R) 267
5 Analyse des correspondances multiples (ACM) 286
5.1 Principe de l'analyse des correspondances multiples 287
5.2 Organisation des calculs 302
5.3 Pratique de l'ACM avec « fanalysis » sous Python 308
5.4 ACM avec d'autres outils (SAS, TANAGRA, R) 344
6 Analyse factorielle des données mixtes (AFDM) 362
6.1 Principe de l'analyse factorielle des données mixtes 364
6.2 Organisation des calculs 366
6.3 Pratique de l'AFDM sous TANAGRA 374
6.4 AFDM avec d'autres outils – R et les packages spécialisés 386
7 Références 396
7.1 Références 396
7.2 Supports de cours 396
7.3 Tutoriels 397
8 Annexes 400
8.1 Gestion des versions 400
8.2 Fichier de données 400
8.3 Notebooks 401

1 Analyse en composantes principales (ACP)

L'analyse en composantes principales (ACP) est une technique exploratoire très populaire. Un signe qui ne trompe pas, la page consacrée aux supports et tutoriels de l'ACP est – de loin – la plus consultée sur mon blog.

Selon les points de vue, on peut la considérer : comme une technique factorielle où l'on essaie de résumer ou regrouper les descripteurs dans leurs dimensions les plus importantes représentées par des variables synthétiques appelées composantes ; comme une technique de visualisation où l'on essaie de préserver les proximités entre les individus dans un espace de représentation réduit ; comme une technique de compression de l'information ; etc.

Le succès de l'ACP repose en très grande partie sur la richesse des représentations graphiques qu'elle propose. Elle sont accompagnées d'outils d'aide à l'interprétation qui permettent de saisir la teneur des composantes (facteurs, axes factoriels) mises en évidence. Les très nombreux exemples illustratifs que l'on trouve ici ou là soulignent ses propriétés descriptives sur des jeux de données de taille réduite, elles permettent de mettre en relation les résultats observés avec des connaissances du domaine. Nous ne dérogerons pas à cette règle.

Mais le champ d'application de l'ACP est plus large. Elle joue un rôle important comme outil de défrichage des grandes bases de données, tant en nombre d'observations que de variables. En réduisant la dimensionnalité avec une perte d'information contrôlée, elle peut constituer une solution de prétraitement privilégiée avant l'utilisation des techniques de machine learning. Son

rôle complémentaire par rapport aux techniques de classification automatique (clustering) en particulier est souligné dans la littérature (Lebart et al., 2000 ; section 2.4).

1.1 Principe de l'analyse en composantes principales

1.1.1 Tableau individus-variables

L'analyse en composantes principales traite les tableaux individus-variables, lesquelles sont toutes quantitatives. Nous reprenons le jeu de données « Autos » (Saporta, 2006 ; page 248) de notre support de cours (COURS ACP, page 4). Il décrit les (p = 6) caractéristiques de (n = 18) véhicules des années 70.

Sous Python, à l'aide de la librairie Pandas, nous chargeons la feuille « DATA_ACP_ACTIF » du fichier « **Data_Methodes_Factorielles.xlsx** ». Nous affichons les propriétés de la base et la matrice des données.

```
#chargement -- index_col = 0 pour indiquer que la colonne n°0 est un label
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="DATA_ACP_ACTIF",index_col=0)

#affichage des caractéristiques de la base
print(D.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18 entries, Alfasud TI to Lada 1300
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   CYL     18 non-null     int64
 1   PUISS   18 non-null     int64
 2   LONG    18 non-null     int64
 3   LARG    18 non-null     int64
 4   POIDS   18 non-null     int64
 5   VMAX    18 non-null     int64
dtypes: int64(6)
memory usage: 1008.0+ bytes
None
```

```
#affichage de la matrice des données
print(D)
```

```
                CYL  PUISS  LONG  LARG  POIDS  VMAX
Modele
Alfasud TI     1350     79   393   161    870   165
Audi 100       1588     85   468   177   1110   160
Simca 1300     1294     68   424   168   1050   152
Citroen GS Club  1222     59   412   161    930   151
Fiat 132       1585     98   439   164   1105   165
```

Lancia Beta	1297	82	429	169	1080	160
Peugeot 504	1796	79	449	169	1160	154
Renault 16 TL	1565	55	424	163	1010	140
Renault 30	2664	128	452	173	1320	180
Toyota Corolla	1166	55	399	157	815	140
Alfetta 1.66	1570	109	428	162	1060	175
Princess 1800	1798	82	445	172	1160	158
Datsun 200L	1998	115	469	169	1370	160
Taunus 2000	1993	98	438	170	1080	167
Rancho	1442	80	431	166	1129	144
Mazda 9295	1769	83	440	165	1095	165
Opel Rekord	1979	100	459	173	1120	173
Lada 1300	1294	68	404	161	955	140

Les variables sont : CYL (cylindrée en cm^3), PUISS (puissance en chevaux din), LONG (longueur en cm), LARG (largeur en cm), POIDS (poids en kg), VMAX (vitesse maximale en km/h).

Dans ce chapitre, nous notons $X = (x_{ij})$ cette matrice composée des observations (i = 1, ..., n) et variables (j = 1, ..., p) actives, dans le sens où elles vont participer aux calculs lors de la construction du repère factoriel.

Dans notre base, $(x_{12} = 79)$ correspond à la puissance (PUISS) de l'Alfasud TI.

Les questions qui viennent naturellement à l'esprit à la lecture de ce type de tableau sont :

- Quels sont les véhicules qui se ressemblent ? Qui s'opposent ? Sur quelles caractéristiques (variables) sont fondées ces ressemblances et dissemblances.
- Quelles sont les relations entre les variables. A priori, surtout pour des véhicules de cette époque où il n'y avait pas (très rarement) de turbo, la cylindrée et la puissance sont certainement positivement corrélées. Mais qu'en est-il des autres ?
- Pour les férus d'automobiles, nous savons qu'elles sont subdivisées en segments (citadines, polyvalentes, familiales, routières, ...). Est-ce qu'il est possible de discerner cette typologie naturelle d'une manière ou d'une autre ? Dans une représentation graphique éventuellement ?

Il y a deux manières complémentaires d'appréhender l'information de ce type de tableau individus-variables : étudier la proximité entre les individus d'une part, analyser les liaisons entre les variables d'autre part.

1.1.2 Analyse des proximités entre individus

1.1.2.1 Représentation dans le plan (p = 2 variables)

Contentons-nous de traiter les variables CYL et PUISS dans un premier temps. Nous pouvons représenter les observations dans le plan.

```
#importer la librairie graphique
import matplotlib.pyplot as plt

#préparer le graphique
fig, ax = plt.subplots(figsize=(10,10))
ax.plot(D.CYL,D.PUISS,"wo")
ax.axis([1000,3000,50,140])
ax.set_xlabel("CYL")
ax.set_ylabel("PUISS")

#ajouter les labels des véhicules
for v in D.index:
    ax.text(D.CYL[v],D.PUISS[v],v)

#faire afficher
plt.show()
```

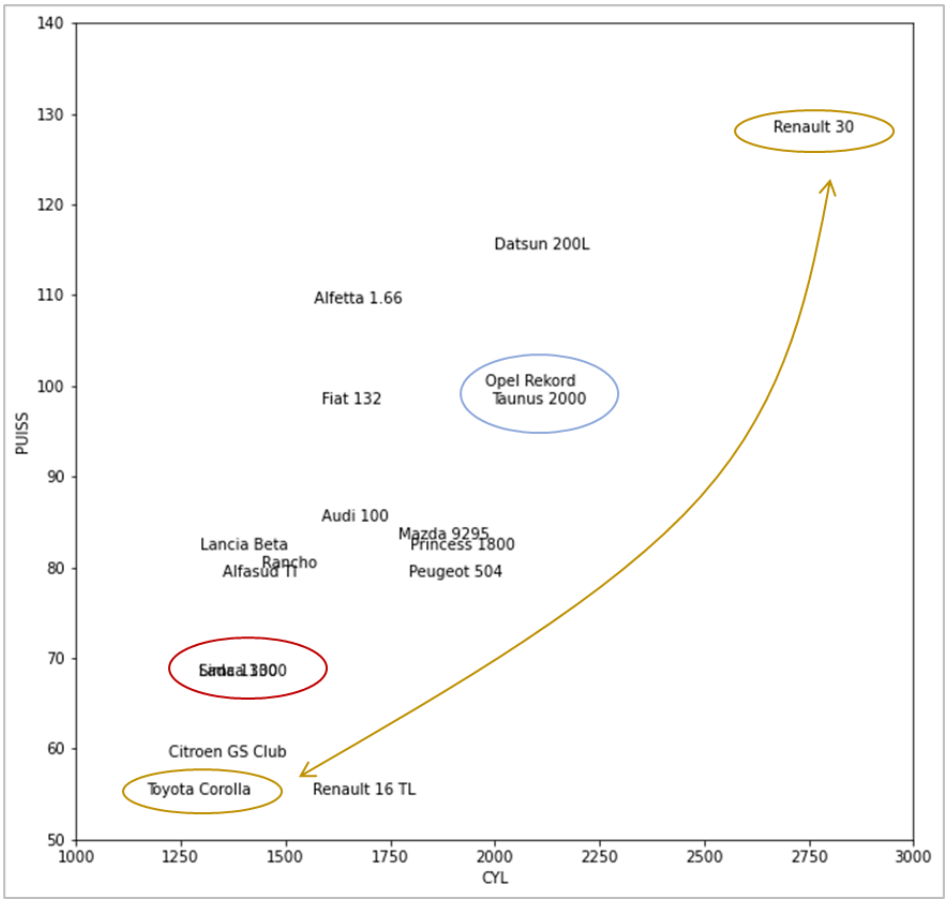


Figure 1 – ACP – Données "Autos" – Nuage de points (CYL vs. PUISS)

Dans ce repère, nous pouvons distinguer plusieurs informations très facilement :

- Les deux variables (CYL, PUISS) sont fortement liées positivement. Quand la cylindrée augmente, la puissance croît également.
- L'Opel Rekord et la (Ford) Taunus 2000 présentent des caractéristiques similaires. Ces véhicules sont en concurrence frontale sur le marché.
- La Lada 1300 et la Simca 1300 présentent des propriétés strictement identiques, raison pour laquelle ils se superposent dans le graphique. Quand on connaît un peu leur genèse, on n'est pas vraiment étonné.
- Les Renault 30 et Toyota Corolla en revanche sont complètement opposés (gros moteur puissant vs. petit moteur anémique).

Remarque : Deux variables sont liées lorsque le nuage de points associé forme une pente, positive ou négative. Dans ces différentes configurations ci-dessous (Figure 2) : (A) correspond à une liaison positive ; (B), (C) et (D) correspondent à une absence de liaison, quand la variable en abscisse varie, celle en ordonnée reste constante en moyenne. On s'intéresse principalement aux liaisons linéaires ici.

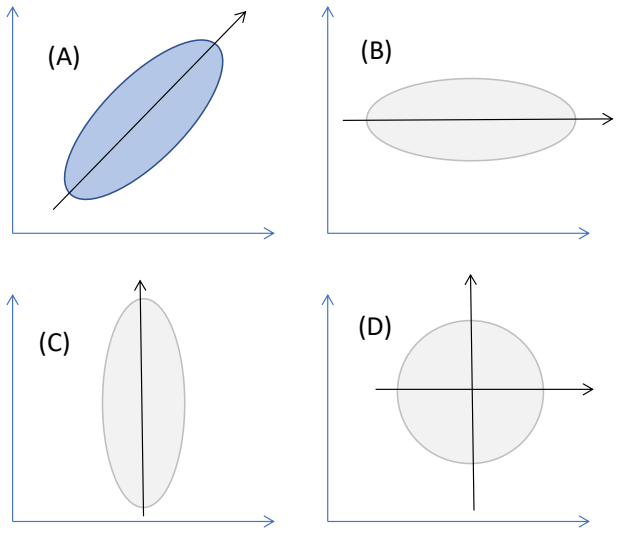


Figure 2 – Types de liens entre deux variables quantitatives

Le graphique (CYL, PUISS) ne fait que révéler des informations contenues dans les données. Nous aurions pu faire les mêmes constations en triant le tableau selon la cylindrée, mais de manière moins avenante reconnaissons-le.

```
#afficher les deux variables en triant selon CYL
Dbis = D.sort_values(by="CYL",ascending=True)[['CYL','PUISS']]
print(Dbis)
```

```
                  CYL  PUISS
Modele
Toyota Corolla   1166     55
Citroen GS Club  1222     59
Lada 1300        1294     68
Simca 1300       1294     68
Lancia Beta      1297     82
Alfasud TI       1350     79
Rancho           1442     80
Renault 16 TL    1565     55
Alfetta 1.66     1570    109
Fiat 132         1585     98
Audi 100         1588     85
Mazda 9295       1769     83
Peugeot 504      1796     79
Princess 1800    1798     82
Opel Rekord      1979    100
Taunus 2000      1993     98
Datsun 200L      1998    115
Renault 30       2664    128
```

1.1.2.2 Représentation lorsque (p > 2) variables

Lorsque (p = 3), un graphique en 3 dimensions reste gérable, même s'il est moins lisible. Au-delà, nous sommes démunis. Quand elles ne sont pas trop nombreuses, on utilise souvent les « pairplot » qui sont constitués de nuages de points des variables prises deux à deux. Espérer y déceler des informations fines sur les proximités entre les individus est illusoire. Lire plusieurs cadrans simultanément n'est pas facile, et étiqueter les observations rendrait le tout totalement illisible. Ce type de graphique sert surtout à effectuer un diagnostic rapide des données : identifier les configurations incongrues (des blocs dans les observations par exemple), les liaisons fortes, ou encore la présence de points atypiques.

Pour les données « Autos », nous observons (Figure 3) que certaines variables sont fortement liées [(CYL, PUISS), (LONG, LARG, POIDS)]. Et qu'une observation se démarque des autres par une inflation des valeurs de cylindrée, puissance, vitesse maximum. Après inspection des données, on se rend compte qu'il s'agit de la Renault 30, voiture de luxe française emblématique des années 70. Elle se positionne différemment par rapport aux autres automobiles.

Sous Python, nous utilisons la librairie « seaborn » pour réaliser un graphique potable en une ligne de commande (Figure 3).

```
#librairie graphique
import seaborn as sns

#pairplot
sns.pairplot(D)
```

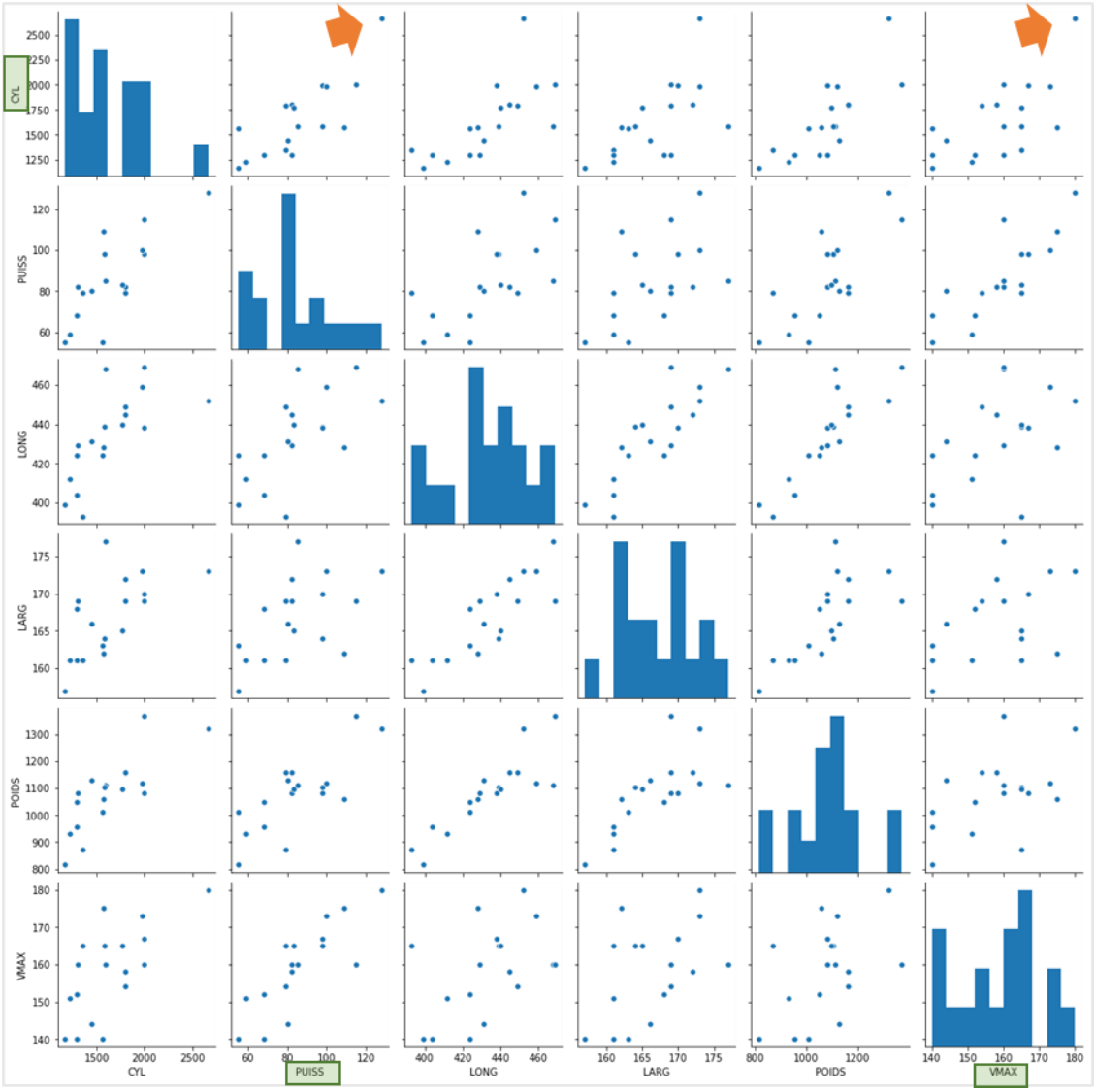


Figure 3 - ACP - Données "Autos" - Pairplot

L'analyse en composantes principales vise à réduire la dimensionnalité en créant des variables synthétiques, combinaisons linéaires des variables originelles, en nombre réduit ($q \ll p$), tout en préservant au mieux l'information. On comprend l'idée. Mais encore faut-il pouvoir : (1)

définir et mesurer l'information véhiculée par les données ; (2) quantifier la qualité de restitution du nouvel espace de représentation .

1.1.2.3 Distance entre individus et inertie



Dans la suite de cette section, nous nous en tenons au traitement des (p = 2) variables CYL et PUISS dans les exemples pour appréhender plus facilement les concepts manipulés (objet Dbis dans le code Python).

Distance euclidienne. La distance permet de mesure le degré d'éloignement entre les individus. La plus usuelle est la distance euclidienne. Entre deux individus (i1) et (i2), elle est définie comme suit :

$$d(i_1, i_2) = \sqrt{\sum_{j=1}^{p} \left(x_{i_1 j} - x_{i_2 j}\right)^2}$$

Aux fins d'illustrations, calculons les distances entre la « Audi 100 » et deux de ses voisins, la « Fiat 132 » et la « Mazda 9295 ».

```
#librairie numpy pour les manipulations matricielles
import numpy

#distance entre l'Audi et la Fiat
numpy.sqrt(numpy.sum((Dbis.loc['Audi 100']-Dbis.loc['Fiat 132'])**2))
```

```
13.341664064126334
```

```
#distance entre l'Audi et la Mazda
numpy.sqrt(numpy.sum((Dbis.loc['Audi 100']-Dbis.loc['Mazda 9295'])**2))
```

```
181.01104938649465
```

Les résultats ne correspondent pas à l'impression visuelle ci-dessus (Figure 3). Contrairement à ce que l'on perçoit, l'Audi serait nettement plus proche de la Fiat que de la Mazda ?

Distance euclidienne pondérée. Le hic vient de la différence d'unités entre les variables. La cylindrée est exprimée en cm^3, la puissance en chevaux DIN. Les valeurs n'ont absolument pas la même amplitude. Les outils graphiques font une mise à l'échelle automatique sur l'étendue (max – min) qui masque cette disparité. Pour retrouver un mécanisme similaire dans le calcul

des distances, une solution simple consiste à réduire les variables c.-à-d. à les diviser par leurs écarts-type respectifs. Cela revient à utiliser une distance euclidienne pondérée par l'inverse de la variance.

$$d(i_1, i_2) = \sqrt{\sum_{j=1}^{p} \frac{1}{\sigma_j^2} \left(x_{i_1 j} - x_{i_2 j}\right)^2}$$

Où σ_j^2 est l'estimation biaisée de la variance :

$$\sigma_j^2 = \frac{1}{n} \sum_{i=1}^{n} \left(x_{ij} - \bar{x}_j\right)^2$$

Et $\bar{x}_j$ est la moyenne de la variable (X_j) :

$$\bar{x}_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij}$$

Revenons à notre exemple ci-dessus. Nous calculons tout d'abord les écarts-type des variables puis nous appliquons la nouvelle formule de la distance.

```
#matrice X numpy - plus facile à manipuler
X = Dbis.values

#écarts-type des variables CYL et PUISS
sigmas = numpy.std(X,axis=0,ddof=0)
print(sigmas)
```

```
[363.39449027  19.80218531]
```

```
#distance pondérée entre l'Audi et la Fiat
numpy.sqrt(numpy.sum((1/sigmas**2)*(Dbis.loc['Audi 100']-Dbis.loc['Fiat 132'])**2))
```

```
0.6565451047185447
```

```
#distance entre l'Audi et la Mazda
numpy.sqrt(numpy.sum((1/sigmas**2)*(Dbis.loc['Audi 100']-Dbis.loc['Mazda 9295'])**2))
```

```
0.5082182304292723
```

Les résultats sont plus conformes à l'intuition visuelle, l'Audi est plus proche de la Mazda que de la Fiat au regard des caractéristiques de cylindrée et de puissance (Figure 1).

Remarque sur la correction par les écarts-type. Cette correction est quasi-automatique en analyse en composantes principales, on parle alors d'**ACP normée**. Elle est justifiée dans la plupart des cas. Mais il faut quand-même rester prudent. Si elles sont exprimées dans les mêmes unités (ex. des notes attribuées à des critères), ou du moins mesurées sur des échelles similaires, il est parfois intéressant de les traiter nativement pour que les variables qui présentent une amplitude supérieure pèsent davantage dans les calculs.

Inertie – Version 1. La quantité d'information contenue dans un jeu de données peut être exprimée par l'inertie. Elle indique la dispersion totale des données. Elle peut être définie par la moyenne des carrés des distances entre paires d'observations.

$$I_p = \frac{1}{2n^2} \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} d^2(i_1, i_2)$$

Sous Python, nous calculons l'inertie sur les donnes (CYL, PUISS) en effectuant une double boucle de manière à bien détailler les étapes. L'inertie totale est égale à $I_p = 132447.68$

```
#nombre d'observations
n = X.shape[0] #18

#inertie version 1 -- distances entre paires d'individus
Ip_v1 = 0

#double boucle (i1, i2)
for i1 in range(n):
    for i2 in range(n):
        Ip_v1 = Ip_v1 + numpy.sum((X[i1,:]-X[i2,:])**2)

#moyenne des écarts entre paires d'individus
Ip_v1 = (1/(2*n**2)) * Ip_v1
print("Inertie, Approche 1 = %.2f" % (Ip_v1))

Inertie, Approche 1 = 132447.68
```

Inertie – Version 2. Une autre manière de considérer l'inertie est de la voir comme une généralisation multidimensionnelle de la variance. Elle exprime alors la dispersion autour du barycentre **G** du nuage de points, défini par le vecteur composé des moyennes des (p) variables $\bar{x} = (\bar{x}_1, \dots, \bar{x}_p)$

$$I_p = \frac{1}{n} \sum_{i=1}^{n} d^2(i, G)$$

Calculons le vecteur moyenne. Dans le plan (CYL, PUISS), nous positionnons le barycentre et nous situons chaque point par rapport à cette référence (Figure 4).

```
#moyennes des variables -- Coordonnées de G
moyennes = numpy.mean(X,axis=0)
print(moyennes)

[1631.66666667   84.61111111]

#graphique des points avec le barycentre G
fig, ax = plt.subplots(figsize=(10,10))
ax.plot(D.CYL,D.PUISS,color='xkcd:light blue',marker='o',linestyle='None')
ax.axis([1000,3000,50,140])
ax.set_xlabel("CYL")
ax.set_ylabel("PUISS")
#ajouter des traits pointillés entre les points et le barycentre
for i in range(D.shape[0]):
  ax.plot([moyennes[0],D.CYL[i]],[moyennes[1],D.PUISS[i]],color='silver',linestyle='dashed')
#barycentre G
ax.text(moyennes[0],moyennes[1],"G",fontsize=14)
#faire afficher
plt.show()
```

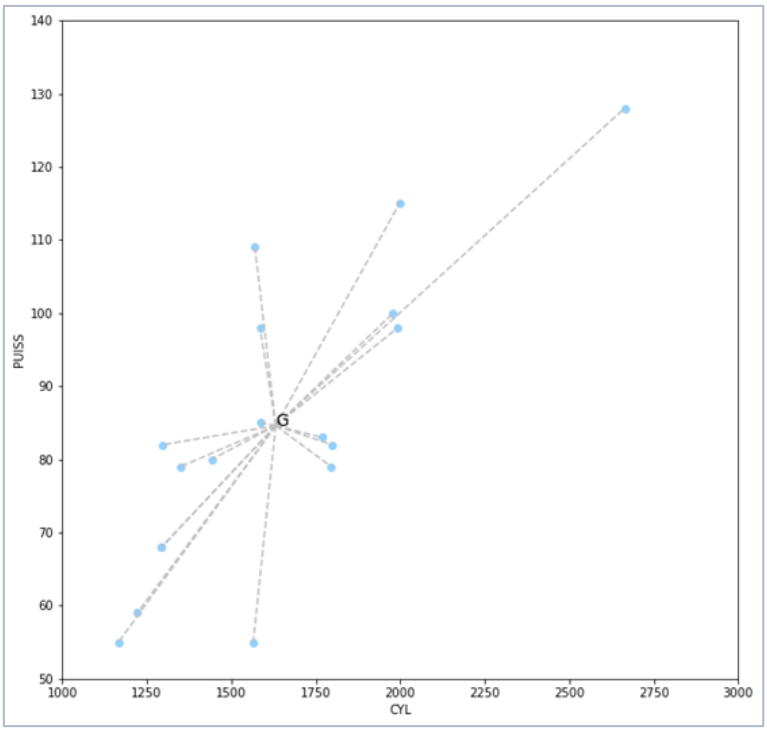


Figure 4 - Dispersion des points autour du barycentre G -- Données "Autos" (CYL, PUISS)

Le calcul de l'inertie est autrement plus simple par rapport à la version précédente.

```
#rappel - vecteur des moyennes
print(moyennes)

[1631.66666667   84.61111111]
```

```
#inertie version 2 - écarts au barycentre
Ip_v2 = numpy.mean(numpy.apply_along_axis(func1d=lambda x: numpy.sum((x-moyennes)**2),axis=1,arr=X))
print("Inertie, Approche 2 = %.2f" % (Ip_v2))


Inertie, Approche 2 = 132447.68
```

Cette écriture a pour avantage de mettre en lumière un résultat important : l'inertie est égale à la somme des variances des variables. En effet :

$$I_p = \frac{1}{n}\sum_{i=1}^{n} d^2(i,G)$$

$$= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p}\left(x_{ij} - \bar{x}_j\right)^2$$

$$= \sum_{j=1}^{p}\frac{1}{n}\sum_{i=1}^{n}\left(x_{ij} - \bar{x}_j\right)^2$$

$$= \sum_{j=1}^{p}\sigma_j^2$$

Nous avions calculé les écarts-type par variables ci-dessus. Il nous suffit de les passer au carré et de faire la somme pour le vérifier.

```
#rappel écarts-type par variable
print(sigmas)
[363.39449027  19.80218531]


#vérification -- somme des variances
print("Somme des variances = %.2f" % (numpy.sum(sigmas**2)))


Somme des variances = 132447.68
```

Oui, l'inertie correspond bien à la somme des variances des variables.

Cas des variables réduites. Lorsque les variables sont réduites, normalisées par les écarts-type, leur variance est égale à 1. L'inertie est alors égale au nombre de variables ($I_p = p$).

1.1.2.4 Principe de l'analyse en composantes principales (1)

Pour des raisons qui seront plus faciles à formuler dans l'étude des relations entre les variables (section 1.1.3), nous centrons et réduisons les descripteurs. Pour (X_j), la variable transformée est définie par

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{\sigma_j}$$

- Le centrage a pour conséquence de ramener le barycentre à l'origine c.-à-d. le vecteur des moyennes est composé de valeurs nulles.
- Avec la réduction, l'inertie est égale au nombre (p) de variables.

Objectif de l'ACP : L'objectif de l'ACP est de construire un système de représentation de dimension réduite ($q \ll p$) qui préserve au mieux les distances entre les individus ou, c'est la même chose, qui préserve au mieux la dispersion des données. Ce nouveau repère est décrit par une succession de (q) variables synthétiques appelées « composantes principales » (facteurs, axes principaux, axes factoriels). Elles sont définies par des combinaisons linéaires des variables (z_j). Pour la composante n°k :

$$F_k = a_{k1}z_1 + \cdots + a_{kp}z_p$$

L'enjeu est d'estimer les valeurs des coefficients (a_{kj}) à partir des données.

Nous noterons qu'il est possible d'exprimer les facteurs à partir des variables originelles. Un terme constant apparaît dans ce cas.

$$F_k = b_{k0} + b_{k1}x_1 + \cdots + a_{kp}x_p$$

La succession de facteurs construits par l'ACP répond au cahier des charges suivant :

1. La première composante F1 est construite de manière à maximiser la dispersion des points lorsqu'ils y sont projetés. La variance associée indique son pouvoir explicatif, elle correspond à une fraction de l'inertie totale. La composante est centrée c.-à-d. elle est de moyenne nulle.

2. La seconde composante F2 procède de la même manière, mais sur l'information résiduelle, non-expliquée par F1. Elle est orthogonale à la première. Elle vient compléter l'information restituée. Sa variance indique toujours la qualité de représentation qu'elle porte. L'addition des variances des deux premières composantes indique leur pouvoir de représentation lorsqu'elles sont combinées.
3. La troisième vient compléter les deux premières. Etc.
4. Nous avons restitué toute l'information disponible lorsque le nombre de facteurs (q) considérés est égal au nombre de variables (p) de la base. La somme des variances des facteurs est égale à l'inertie totale.

La compression est d'autant meilleure qu'un nombre (q) faible de facteurs permet de reproduire une fraction importante de l'information initiale.

Pour l'exemple (CYL, PUISS) centrées et réduites, nous matérialisons les deux facteurs de l'ACP (il ne peut y en avoir que 2 au maximum puisque p = 2) (Figure 5).

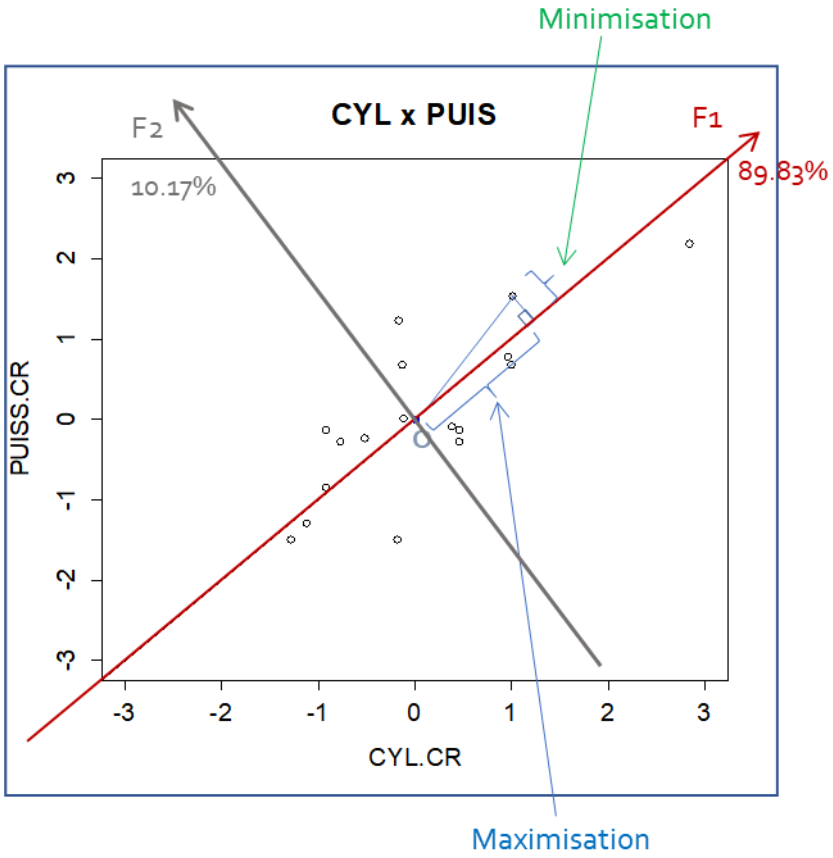


Figure 5 – Composantes principales – Données "Autos" (CYL, PUISS)

Sur l'axe F1, nous souhaitons que les projections des points soient le plus écartés possibles de l'origine. Ou, de manière équivalente puisque nous avons un triangle rectangle, les distances des points à la droite (F1) soient les plus faibles possibles. Ainsi, nous pouvons voir l'ACP comme un ajustement d'un nuage de points par une droite (plan, hyperplan), une régression sans variable cible où les erreurs sont comptabilisées orthogonalement.

Sous Python, nous standardisons les variables, puis nous allons à l'essentiel avec la classe PCA de la librairie « scikit-learn » (TUTO 1). Nous reviendrons sur le détail des calculs sous-jacents plus loin (section 1.2). Nous obtenons les coordonnées factorielles des individus dans un premier temps, nous les projetons dans le plan.

```
#données centrées et réduites
Z = (X - moyennes)/sigmas
print(pandas.DataFrame(Z,index=Dbis.index))
```

```
                        0         1
Modele
Toyota Corolla  -1.281436 -1.495346
Citroen GS Club -1.127333 -1.293348
Lada 1300       -0.929201 -0.838852
Simca 1300      -0.929201 -0.838852
Lancia Beta     -0.920946 -0.131860
Alfasud TI      -0.775099 -0.283358
Rancho          -0.521930 -0.232859
Renault 16 TL   -0.183455 -1.495346
Alfetta 1.66    -0.169696  1.231626
Fiat 132        -0.128419  0.676132
Audi 100        -0.120163  0.019639
Mazda 9295       0.377918 -0.081360
Peugeot 504      0.452217 -0.283358
Princess 1800    0.457721 -0.131860
Opel Rekord      0.955802  0.777131
Taunus 2000      0.994328  0.676132
Datsun 200L      1.008087  1.534623
Renault 30       2.840806  2.191116
```

```
#vérification moyennes - nulles
print(numpy.mean(Z,axis=0))
```

```
[-1.72701359e-16 -1.48029737e-16]
```

```
#vérification écarts-type - égaux à 1
print(numpy.std(Z,axis=0,ddof=0))
```

```
[1. 1.]
```

```
#outil pour l'ACP
from sklearn.decomposition import PCA
acp = PCA()

#coordonnées factorielles
```

```
coord = acp.fit_transform(Z)

#afficher les nouvelles coordonnées
print(pandas.DataFrame(coord,index=Dbis.index))
```

```
                         0         1
Modele
Toyota Corolla  -1.963481 -0.151257
Citroen GS Club -1.711680 -0.117390
Lada 1300       -1.250203  0.063886
Simca 1300      -1.250203  0.063886
Lancia Beta     -0.744446  0.557968
Alfasud TI      -0.748442  0.347713
Rancho          -0.533717  0.204405
Renault 16 TL   -1.187092 -0.927647
Alfetta 1.66     0.750898  0.990885
Fiat 132         0.387292  0.568903
Audi 100        -0.071082  0.098855
Mazda 9295       0.209698 -0.324759
Peugeot 504      0.119402 -0.520131
Princess 1800    0.230419 -0.416897
Opel Rekord      1.225369 -0.126340
Taunus 2000      1.181194 -0.224999
Datsun 200L      1.797968  0.372317
Renault 30       3.558106 -0.459400
```

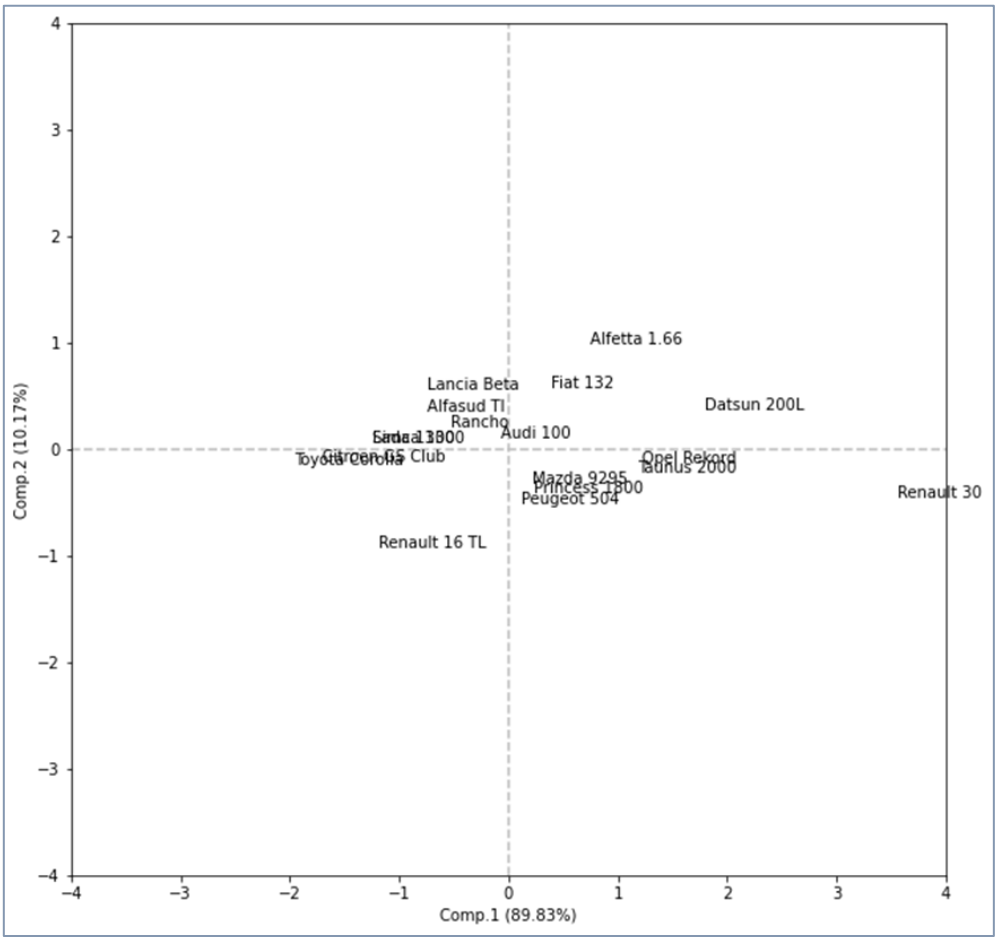


Figure 6 – Plan factoriel – Données ”Autos” (CYL, PUISS)

```
#position des véhicules dans le repère factoriel
fig, ax = plt.subplots(figsize=(10,10))
ax.plot(coord[:,0],coord[:,1],"wo")
ax.axis([-4,+4,-4,+4])
ax.plot([-4,+4],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-4,+4],color='silver',linestyle='--')
ax.set_xlabel("Comp.1 (89.83%)")
ax.set_ylabel("Comp.2 (10.17%)")

#ajouter les labels des véhicules
for i in range(n):
    ax.text(coord[i,0],coord[i,1],Dbis.index[i])

#faire afficher
plt.show()
```

Plusieurs commentaires quant à ce graphique (Figure 6) :

- C'est le même graphique que le précédent (Figure 5), mais après rotation. Nous avons la composante F1 en abscisse, F2 en ordonnée.
- Il est d'usage de faire apparaître les proportions d'inerties restituées sur les axes pour situer leurs importances relatives. Ces indications sont cruciales surtout lorsque les logiciels procèdent à une mise à l'échelle automatique de manière à ce que les min et max soient situés aux extrémités du graphique, laissant croire à une dispersion identique en abscisse et ordonnée.
- Une autre manière de se prémunir de ces distorsions intempestives est de fixer explicitement les mêmes limites en abscisse et ordonnée. Nous constatons ainsi pour notre exemple que la dispersion est surtout perceptible sur la première composante qui restitue 89.83% de l'information disponible.

La variance restituée sur les facteurs est égale à :

$$\lambda_k = \frac{1}{n}\sum_{i=1}^{n} F_{ik}^2$$

Où F_{ik} est la coordonnée de l'individu n°i sur la composante F_k. Rappelons que les moyennes des facteurs sont nulles par construction puisque les données ont été centrées, d'où cette formule simplifiée de la variance.

Sur nos données,

```
#calculer les lambda
lambada = numpy.mean(numpy.power(coord,2),axis=0)
print(lambada)

[1.79662771 0.20337229]
```

($\lambda_1 = 1.796 \ldots$) et ($\lambda_2 = 0.203 \ldots$).

L'information est plus parlante en proportions de variance restituée.

```
#part d'inertie expliquée par les axes
print(lambada/numpy.sum(lambada))

[0.89831386 0.10168614]
```

Nous avons 89.83% de l'information sur le premier facteur, 10.17% sur le second.

Et nous vérifions bien que la somme des variances est égale à l'inertie totale :

$$I_p = \sum_{k=1}^{p} \lambda_k$$

```
#somme des lambda = inertie totale
print("Somme des lambda = %f" % (numpy.sum(lambada)))

Somme des lambda = 2.000000
```

1.1.2.5 Préservation des distances

Outre la restitution de la dispersion, nous disions plus haut que l'objectif de l'analyse en composantes principales était aussi de réduire la dimensionnalité tout en préservant les distances entre les individus. Voyons ce qu'il en est sur quelques exemples en comparant les distances euclidiennes sur les données centrées et réduites (Z) et celles dans le repère factoriel (coord), selon le nombre de composantes que nous prenons en compte.

Distance approchée. Dans un premier temps, nous nous intéressons à la qualité d'approximation lorsque nous prenons en compte qu'une partie des facteurs, le premier ici en l'occurrence (F1). Nous mesurons (le carré de) la distance entre la « Toyota Corolla » et la « Citroën GS Club » sur les variables centrées et réduites.

$$\begin{aligned} d^2(Corolla, GS) &= \left(-1.2814 - (-1.1273)\right)^2 + \left(-1.4953 - (-1.2933)\right)^2 \\ &= 0.06455 \end{aligned}$$

Avec les coordonnées factorielles sur le premier facteur (F1).

$$d^2_{\{F_1\}}(Corolla, GS) = (1.9335 - 1.7117)^2$$
$$= 0.06340$$

L'approximation est plutôt bonne dans le cas présent. On s'y attendait un peu, sachant que la qualité de représentation de la composante (F1) est plutôt intéressante (89.83%).

Maintenant que nous avons compris le principe, effectuons les calculs sous Python. Renouvelons la comparaison entre la Corolla et la GS.

```
#data frame des données centrées et réduites
DZ = pandas.DataFrame(Z,index=Dbis.index)

#data frame des coordonnées factorielles
DC = pandas.DataFrame(coord,index=Dbis.index)

#(distance euclidienne)^2 entre les Toyota Corolla et Citroen GS
#données originelles, centrées et réduites
print(numpy.sum((DZ.loc['Toyota Corolla']-DZ.loc['Citroen GS Club'])**2))

0.06455073567278283

#(distance euclidienne)^2 en en prenant en compte que la première composante
print(numpy.sum((DC.loc['Toyota Corolla'][0]-DC.loc['Citroen GS Club'][0])**2))

0.06340375071089865
```

Voyons ce qu'il en est pour l'Audi 100 et la Fiat 132 qui étaient des concurrents directs sur le segment des berlines familiales.

```
#distance initiale entre Audi 100 et Fiat 132
print(numpy.sum((DZ.loc['Audi 100']-DZ.loc['Fiat 132'])**2))

0.4310514745298847

#distance approchée (Facteur F1)
print(numpy.sum((DC.loc['Audi 100'][0]-DC.loc['Fiat 132'][0])**2))

0.2101060634608986
```

Patatras ! Bien que le facteur (F1) ait un pouvoir de restitution élevé, l'approximation est mauvaise. On en déduit que la justesse de la restitution de la distance dépend certes du mérite des composantes mises en jeu, mais aussi de la qualité de représentation des points étudiés c.-à-d. de leurs positions sur ces composantes. Nous approfondirons ces notions lors de l'étude des outils d'aide à l'interprétation de l'analyse en composantes principales (section 1.3).

Distance exacte. Lorsque nous prenons en compte tous les facteurs (F1, F2), nous retrouvons exactement les distances dans l'espace initial. Pour la Corolla et la GS par exemple.

$$\begin{aligned} d^2_{\{F_1,F_2\}}(Corolla, GS) &= (1.9335 - 1.7117)^2 + (0.1513 - 0.1174)^2 \\ &= 0.06455 \\ &= d^2(Corolla, GS) \end{aligned}$$

Pour l'Audi et la Fiat maintenant.

```
#distance si on prend en compte (F1, F2)
print(numpy.sum((DC.loc['Audi 100']-DC.loc['Fiat 132'])**2))
```

```
0.43105147452988485
```

Une des questions clés de l'ACP est de définir le nombre de composantes « q » à retenir pour disposer d'une approximation suffisamment satisfaisante des proximités pour une fraction suffisamment importante des points, afin que les résultats – et surtout les graphiques – soient exploitables.

1.1.3 Analyse des relations entre les variables



Dans cette section, nous revenons à l'ensemble des (**p = 6**) variables de la base « Autos ».

Examiner les relations entre les variables est le second prisme de l'exploration des données. Pour la base « Autos », selon la pente et l'allongement des nuages de points, nous avions observé dans le « pairplot » qu'elles étaient plus ou moins liées (Figure 3). Pour traiter les grandes bases, il nous faut de nouveau une approche synthétique pour mesurer la quantité d'information disponible à partir de ce point de vue, puis la décomposer par la suite sur les facteurs de l'ACP.

Dans cette section, nous présentons dans un premier temps différents indicateurs permettant de restituer l'existence et l'intensité de la liaison entre les variables. Nous définirons une approche alternative pour définir les composantes principales de l'ACP à partir de cette nouvelle perspective d'analyse dans un second temps.

1.1.3.1 Matrice des covariances

Matrice de covariance. La covariance mesure le sens et l'intensité de la liaison entre deux variables. Elle identifie leur tendance à être simultanément au-dessus ou en dessous de leurs espérances respectives. Elle caractérise une liaison monotone (Rakotomalala R., « Analyse de corrélation. Etude des dépendances – Variables quantitatives », version 1.1, mars 2015). Elle peut être positive ou négative. Plus sa valeur est élevée en valeur absolue, plus forte est la relation.

Dans un cadre multivarié, la matrice des variances covariances V permet de rendre compte des liaisons des variables prises deux-à-deux. Elle est de terme :

$$v_{j_1,j_2} = \frac{1}{n}\sum_{i=1}^{n}(x_{ij_1} - \bar{x}_{j_1})(x_{ij_2} - \bar{x}_{j_2})$$

Nous observons les variances des variables sur la diagonale de la matrice V. La trace de cette matrice V – la somme des éléments diagonaux - correspond ainsi à la somme des variances c.-à-d. la quantité totale d'information disponible :

$$Tr(V) = \sum_{j=1}^{p} v_{jj}$$

Nous constatons ainsi que les deux points de vue sur les données - l'analyse des proximités entre les individus et l'analyse des liaisons entre les variables - sont cohérents. Passer par l'inertie ou passer par la matrice des covariances permettent de quantifier de manière identique l'information portée par les données.

Sous Python, nous formons la structure X avec l'ensemble des (p = 6) variables et nous faisons appel à la fonction cov() de la libraire Numpy pour former la matrice V. Nous calculons sa trace.

```
#former la matrice X avec (p=6) variables maintenant
X = D.values

#calculer la matrice de covariance
#rowvar = False pour dire que les variables sont organisées en colonnes
#ddof = 0 pour utiliser (1/n)
V = numpy.cov(X,ddof=0,rowvar=False)
numpy.set_printoptions(precision=2,suppress=True)
print(V)
```

```
[[132055.56   5732.54   5476.56   1181.78  38159.61   2850.87]
 [  5732.54    392.13    272.86     53.26   2017.05    197.27]
 [  5476.56    272.86    461.58     94.22   2482.36    120.64]
 [  1181.78     53.26     94.22     26.67    492.72     28.81]
 [ 38159.61   2017.05   2482.36    492.72  17715.36    749.99]
 [  2850.87    197.27    120.64     28.81    749.99    139.2 ]]

#calculer la trace de la matrice V
trace = V.trace()
print(trace)

150790.49382716054
```

Pour vérifier la cohérence avec l'analyse des proximités entre les individus, nous produisons l'inertie totale sous la forme de la somme des distances au carré au barycentre, dont nous calculons tout d'abord les coordonnées.

```
#vecteur moyenne pour (p = 6) variables
moyennes = numpy.mean(X,axis=0)

#inertie par l'écart au barycentre
print(numpy.mean(numpy.apply_along_axis(func1d=lambda x: numpy.sum((x-moyennes)**2),axis=1,arr=X)))

150790.4938271605
```

S'il y avait un doute, il est levé. Ce sont bien là deux faces de la même pièce.

Produit matriciel des variables centrées. Le centrage des variables est fondamental en ACP. Il permet de ramener le barycentre des données à l'origine. Aux descripteurs originels (X_j), nous substituons (Z_j) avec :

$$z_{ij} = x_{ij} - \bar{x}_j$$

La matrice de covariance peut être alors obtenue par le produit matriciel (Z^TZ) à un facteur près :

$$V = \frac{1}{n} Z^T Z$$

Nous vérifions cela très facilement sur nos données. Nous centrons d'abord les variables puis nous formons le produit matriciel, nous obtenons le résultat de cov().

```
#matrice Z des variables centrées
Z = X - moyennes
print(pandas.DataFrame(Z,index=D.index))

                           0         1     2         3          4  \
Modele
Alfasud TI       -281.666667  -5.611111 -40.5  -5.666667 -208.833333
```

```
Audi 100          -43.666667   0.388889  34.5  10.333333   31.166667
Simca 1300       -337.666667 -16.611111  -9.5   1.333333  -28.833333
Citroen GS Club  -409.666667 -25.611111 -21.5  -5.666667 -148.833333
...
                         5
Modele
Alfasud TI        6.722222
Audi 100          1.722222
Simca 1300       -6.277778
Citroen GS Club  -7.277778
...
```

```
#(1/n) (Z'Z) ➔ V
print(numpy.dot(numpy.transpose(Z),Z)/n)
```

```
[[132055.56   5732.54   5476.56   1181.78  38159.61   2850.87]
 [  5732.54    392.13    272.86     53.26   2017.05    197.27]
 [  5476.56    272.86    461.58     94.22   2482.36    120.64]
 [  1181.78     53.26     94.22     26.67    492.72     28.81]
 [ 38159.61   2017.05   2482.36    492.72  17715.36    749.99]
 [  2850.87    197.27    120.64     28.81    749.99    139.2 ]]
```

1.1.3.2 Matrice des corrélations

Lorsque les variables sont centrées et réduites....

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{\sigma_j}$$

... nous aboutissons à la matrice de corrélations,

$$R = \frac{1}{n} Z^T Z$$

De terme :

$$r_{j_1,j_2} = \frac{v_{j_1,j_2}}{\sigma_{j_1} \times \sigma_{j_2}}$$

Son intérêt est que ses valeurs sont bornées : $(-1 \leq r_{j_1,j_2} \leq +1)$. Nous situons d'emblée l'intensité des relations linéaires entre les paires de descripteurs. Et surtout, comme signalé plus haut (section 1.1.2.3, « Distance euclidienne pondérée »), la standardisation permet de dépasser les problèmes de disparités d'échelles entre les variables.

Puisque la diagonale contient exclusivement des valeurs égales à 1 (la corrélation d'une variable avec elle-même est égale à 1), la trace de la matrice (R) est égale à (p), le nombre de variables. Elle correspond également à l'inertie totale.

Sous Python, la fonction corrcoef() de la librairie « Numpy » fait très bien l'affaire.

```
#matrice des corrélations
R = numpy.corrcoef(X,rowvar=False)
print(R)

[[1.   0.8  0.7  0.63 0.79 0.66]
 [0.8  1.   0.64 0.52 0.77 0.84]
 [0.7  0.64 1.   0.85 0.87 0.48]
 [0.63 0.52 0.85 1.   0.72 0.47]
 [0.79 0.77 0.87 0.72 1.   0.48]
 [0.66 0.84 0.48 0.47 0.48 1.  ]]
```

Nous vérifions que la même matrice peut obtenue par l'intermédiaire du produit (Z^TZ) :

```
#vecteur ecart-type pour (p = 6) variables
sigmas = numpy.std(X,axis=0)

#centrage et réduction
Z = (X - moyennes)/sigmas

#correspondance : produit matriciel : (1/n) (Z'Z)
print(numpy.dot(numpy.transpose(Z),Z)/n)

[[1.   0.8  0.7  0.63 0.79 0.66]
 [0.8  1.   0.64 0.52 0.77 0.84]
 [0.7  0.64 1.   0.85 0.87 0.48]
 [0.63 0.52 0.85 1.   0.72 0.47]
 [0.79 0.77 0.87 0.72 1.   0.48]
 [0.66 0.84 0.48 0.47 0.48 1.  ]]
```

Un « heatmap » permet de visualiser rapidement les corrélations les plus élevées.

```
#heatmap pour identifier visuellement les corrélations fortes
sns.heatmap(R,xticklabels=D.columns,yticklabels=D.columns,vmin=-1,vmax=+1,center=0
,cmap="RdBu",linewidths=0.5)
```

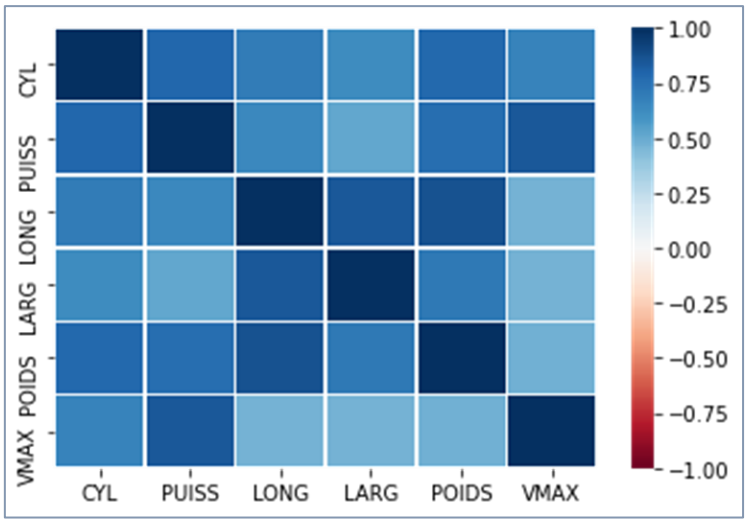


Figure 7 - "Heatmap" des corrélations croisées - Données "Autos"

Nous notons que les plus fortes corrélations ont lieu entre (PUISS, VMAX), et entre (LONG, LARG, POIDS). Nous observons également que le niveau global des corrélations est élevé.

Le « heatmap » tient (un peu) du gadget sur (p = 6) variables. Mais sur une plus forte dimensionnalité, organisé judicieusement à l'aide de l'ACP par exemple, ce type de graphique peut se révéler décisif pour identifier rapidement les liaisons et/ou groupes de liaisons importantes qui structurent les données.

1.1.3.3 Principe de l'analyse en composantes principales (2)

Ce nouveau prisme nous permet de poser autrement le problème d'analyse en composantes principales. Elle a pour objectif toujours d'élaborer une succession de facteurs. Le premier est construit de manière à maximiser la somme des carrés des corrélations avec les variables :

$$\lambda_1 = \sum_{j=1}^{p} r_j^2(F_1)$$

Où $r_j(F_1)$ indique la corrélation de la variable n°j avec le facteur (F1).

La seconde composante (F2) répond à la même spécification, mais travaille sur la partie résiduelle, non-expliquée par la précédente. Elle est donc orthogonale à (F1). La troisième composante, etc.

Les (λ_k) sont exactement les mêmes que lors de l'analyse des proximités entre les individus. Elles correspondent à l'inertie restituée par les facteurs (F_k). Comme ces derniers sont deux à deux orthogonaux, les (λ_k) s'additionnent d'un facteur à l'autre. La somme totale est égale l'inertie globale. Pour une ACP normée, lorsque nous travaillons à partir de la matrice des corrélations, elle est égale à p, le nombre de descripteurs.

Nous sollicitons de nouveaux la classe PCA de « scikit-learn » pour réaliser une ACP sur les données centrées et réduites c.-à-d. une ACP normée. Nous produisons les coordonnées factorielles des individus. Nous calculons alors les (λ_k) via la somme des carrés des corrélations entre les variables et les facteurs. La somme des (λ_k) devrait être égale à (p). Nous verrons bien.

```
#acp sur Z (p = 6 variables)
acp = PCA()

#coordonnées factorielles
coord = acp.fit_transform(Z)

#afficher les nouvelles coordonnées des premiers véhicules
print(pandas.DataFrame(coord,index=Dbis.index).head())
```

```
                        0         1         2         3         4         5
Modele
Toyota Corolla  -2.138924  1.785681  0.571862  0.201927 -0.301357  0.053921
Citroen GS Club  1.561459 -1.527040  1.315271 -0.211352  0.148516 -0.327238
Lada 1300       -1.119385 -0.674505  0.456588 -0.167626 -0.375364  0.271803
Simca 1300      -2.573742  0.112884  0.148570 -0.017343  0.226853  0.263476
Lancia Beta      0.427855  0.695567 -0.193286 -0.627754  0.263505 -0.037153
```

```
#nombre de variables
p = Z.shape[1]
print(p)
```

```
6
```

```
#corrélation des facteurs avec les variables
Mlambda = numpy.corrcoef(x=coord,y=Z,rowvar=False)[:p,p:]

#affichage des corrélations : lignes = facteurs, colonnes = variables
print(Mlambda)
```

```
[[ 0.89  0.89  0.89  0.81  0.91  0.75]
 [ 0.11  0.38 -0.38 -0.41 -0.22  0.57]
 [-0.22 -0.11  0.04  0.37 -0.3   0.3 ]
 [ 0.37 -0.17 -0.13  0.1  -0.14 -0.03]
 [ 0.05 -0.09  0.22 -0.15 -0.09  0.06]
 [-0.01 -0.13 -0.04 -0.02  0.12  0.1 ]]
```

```
#somme des corrélations^2 entre les variables et les facteurs
#on doit obtenir les lambda_k
```

```
slambda = numpy.sum(Mlambda**2,axis=1)
print(slambda)
```

```
[4.42 0.86 0.37 0.21 0.09 0.04]
```

```
#dont la somme totale (des lambda) = inertie = p puisque ACP normée
print(numpy.sum(slambda))
```

```
6.000000000000002
```

La somme des (λ_k) est égale au nombre de variables (p = 6). Dans notre exemple, nous observons que ($\lambda_1 = 4.42$). Elle explique pour (4.42 / p = 73.68%) de l'information disponible.

Bien sûr, si nous souhaitons faire le pont avec l'approche sous l'angle des proximités entre les individus (section 1.1.2.4), nous obtenons exactement les mêmes (λ_k) en calculant les variances des facteurs.

```
#faire le lien avec le premier prisme
#variance des coordonnées factorielles
print(numpy.var(coord,axis=0,ddof=0))
```

```
[4.42 0.86 0.37 0.21 0.09 0.04]
```

1.1.3.4 Préservation des corrélations

Comme pour les distances entre individus, nous pouvons approcher les corrélations entre les variables dans un espace factoriel de dimension réduite. Si $r(V_1, V_2)$ est la corrélation brute entre les variables (V_1,V_2), la corrélation approchée sur les (q) premiers facteurs s'écrit :

$$r_{\{F_1,\dots,F_q\}}(V_1, V_2) = \sum_{k=1}^{q} r_{V_1}(F_k) \times r_{V_2}(F_k)$$

Dans le code Python ci-dessus, la matrice (R) représente les corrélations brutes entre les variables, (Mlambda) les corrélations des variables avec les facteurs. Pour quelques exemples, voyons de quelle manière les corrélations peuvent être approximées avec la première ou les deux premières composantes.

(CYL, POIDS). Pour le cas de la corrélation entre CYL et POIDS.

```
#liste des variables
print(D.columns)
```

```
Index(['CYL', 'PUISS', 'LONG', 'LARG', 'POIDS', 'VMAX'], dtype='object')
```

```
#corrélation entre CYL et POIDS
print(R[0,4])

0.7889520283865826

#corrélation approchée sur le premier facteur
print(Mlambda[0,0]*Mlambda[0,4])

0.8087519932967818
```

L'approximation est déjà de bonne facture sur la première composante. Elle ne peut être meilleure si nous rajoutons la seconde.

```
#corrélation approchée sur les 2 premiers facteurs
print(Mlambda[0,0]*Mlambda[0,4]+Mlambda[1,0]*Mlambda[1,4])

0.7829518411637988
```

Et c'est bien le cas.

(POIDS, VMAX) Pour le cas de la corrélation entre POIDS et VMAX maintenant.

```
#corrélation entre POIDS et VMAX
print(R[4,5])

0.4775955992662051

#corrélation approchée sur le premier facteur
print(Mlambda[0,4]*Mlambda[0,5])

0.6831543614480379
```

L'approximation est mauvaise pour ce cas. Rajoutons de la précision avec la seconde composante.

```
#corrélation approchée sur les 2 premiers facteurs
print(Mlambda[0,4]*Mlambda[0,5]+Mlambda[1,4]*Mlambda[1,5])

0.55438062524932
```

Il y a une amélioration certes, mais nous sommes loin du compte encore.

Conclusion : Comme pour les distances entre les individus, la précision de l'approximation des corrélations dépend : (1) du pouvoir de restitution des facteurs utilisés (pourcentage d'inertie porté par les facteurs) ; (2) de la qualité de représentation des variables sur ces facteurs, qu'il faudra pouvoir caractériser (section 1.3).

1.2 Organisations des calculs

Nous savons maintenant ce que nous souhaitons obtenir. Nous décryptons dans cette section comment y parvenir à partir de nos données en mettant (un peu) les mains dans le cambouis des calculs matriciels. Comprendre les mécanismes internes nous donnera une meilleure visibilité sur la nature de la méthode et la teneur des résultats.

Récapitulons. A partir d'un ensemble de données dont la quantité d'information totale est comptabilisée par l'inertie I_p, l'objectif de l'ACP est de construire des facteurs :

$$\begin{cases} F_1 = a_{11}\, z_1 + a_{21} z_2 + \cdots + a_{p1}\, z_p \\ \cdots \\ F_k = a_{1k}\, z_1 + a_{2k} z_2 + \cdots + a_{pk}\, z_p \\ \cdots \\ F_p = a_{1p}\, z_1 + a_{2p} z_2 + \cdots + a_{pp}\, z_p \end{cases}$$

- La variable (z_j) représente la variable originelle (x_j) centrée (ACP non-normée) ou centrée et réduite (ACP normée) ;
- Le degré de représentativité des facteurs est matérialisé par la variance expliquée (λ_k) ;
- Le facteur (F_k) est construit à partir de l'information résiduelle des (k-1) précédents :
    - Les facteurs sont par conséquent d'importance décroissante ;
    - Et ils sont deux à deux orthogonaux ;
- En pratique, on se contente de ne retenir que sur les (q) premiers, avec ($q \ll p$), qui portent une fraction suffisamment importante de l'information disponible. Nous pouvons ainsi négliger les (p-q) facteurs restants. Cette forme de « compression » fait tout l'intérêt de l'ACP. Si l'on considère tous les facteurs (q = p), nous disposons de toute l'information initialement disponible.
- Dans l'espace factoriel défini par les (q) premières composantes, nous devons pouvoir reconstituer de manière satisfaisante les proximités entre les individus et les liaisons entre les variables.

Comment obtenir les coefficients (a_{jk}) à partir des données ?

1.2.1 Calculs (1) – Diagonalisation de la matrice des corrélations

Pourquoi la diagonalisation de la matrice des corrélations ? La première approche consiste à considérer le problème de l'ACP tel qu'il est posé dans l'analyse des relations des variables avec les facteurs (section 1.1.3.3). Si l'on s'en tient au premier facteur, on cherche à estimer le vecteur $a = (a_1, \dots, a_p)$ telle que la somme des carrés des corrélations des variables avec le facteur soit maximale.

En termes matriciels, nous pouvons l'écrire : $\max_a a^T R\, a$

Dans le même temps, on souhaite que le vecteur (a) soit normé : $\|a\| = a^T a = 1$

Nous avons donc un problème d'optimisation sous contrainte :

$$\begin{cases} \max_a a^T R a \\ s.c.: \ a^T a = 1 \end{cases}$$

Nous passons par le lagrangien pour le résoudre :

$$L = a^T R\, a - \lambda\,(a^T a - 1)$$

En calculant la dérivée partielle par rapport à (a), nous obtenons deux résultats importants :

1. $(R\, a = \lambda\, a)$, le vecteur solution (a) est le premier vecteur propre de la matrice des corrélations R, (λ) est la valeur propre associée.
2. $(\lambda = a^T R\, a)$, (λ) correspond à la valeur propre.

Exemple de calcul sous Python. Et, de manière générale, les vecteurs et valeurs propres de la matrice R (en ACP normée) constituent les solutions de l'ACP : les vecteurs propres sont deux à deux orthogonaux, ils fournissent directement les coefficients (a_{jk}) définissant les composantes principales (F_k) ; les valeurs propres (λ_k) traduisent la variance restituée par les facteurs.

La librairie « Numpy » regorge de fonctions qui font notre bonheur. Dans ce qui suit, nous affichons la matrice des corrélations pour rappel, nous la diagonalisons, nous procédons à quelques vérifications enfin.

```
#rappel matrice des corrélations
print(R)
```

```
[[1.   0.8  0.7  0.63 0.79 0.66]
 [0.8  1.   0.64 0.52 0.77 0.84]
 [0.7  0.64 1.   0.85 0.87 0.48]
 [0.63 0.52 0.85 1.   0.72 0.47]
 [0.79 0.77 0.87 0.72 1.   0.48]
 [0.66 0.84 0.48 0.47 0.48 1.  ]]
```

```
#calcul des valeurs et vecteurs propres
sol = numpy.linalg.eig(R)
print(sol)

#valeurs propres

(array([4.42, 0.86, 0.37, 0.21, 0.09, 0.04]),

#vecteurs propres – organisés en colonnes

 array([[-0.42, -0.12, -0.35,  0.81,  0.15, -0.06],
        [-0.42, -0.42, -0.18, -0.36, -0.29, -0.63],
        [-0.42,  0.41,  0.07, -0.28,  0.73, -0.19],
        [-0.39,  0.45,  0.6 ,  0.21, -0.48, -0.11],
        [-0.43,  0.24, -0.48, -0.3 , -0.3 ,  0.58],
        [-0.36, -0.62,  0.49, -0.07,  0.19,  0.46]]))
```

– Si l'on souhaite accéder exclusivement aux valeurs propres.

```
#valeurs propres
print(sol[0])

[4.42 0.86 0.37 0.21 0.09 0.04]
```

– Au premier vecteur propre.

```
#vecteurs propres pour le premier facteur
print(sol[1][:,0])

[-0.42 -0.42 -0.42 -0.39 -0.43 -0.36]
```

– Vérifions sa norme : elle est égale à 1 (aux erreurs de troncature près)

```
#vérifions sa norme
print(numpy.linalg.norm(sol[1][:,0]))

0.9999999999999999
```

– Est-ce que le repère factoriel est orthonormal ? Oui.

```
#base orthonormale ?
print(numpy.dot(numpy.transpose(sol[1]),sol[1]))

[[ 1. -0.  0.  0. -0. -0.]
 [-0.  1.  0.  0.  0.  0.]
 [ 0.  0.  1. -0. -0. -0.]
```

```
 [ 0.  0. -0.  1. -0.  0.]
 [-0.  0. -0. -0.  1.  0.]
 [-0.  0. -0.  0.  0.  1.]]
```

Coordonnées factorielles des individus. Pour calculer les coordonnées factorielles des individus, il suffit d'appliquer les coefficients estimés (les vecteurs propres) sur les variables centrées et réduites.

$$F_{ik} = \sum_{j=1}^{p} a_{jk} \times z_{ij}$$

Un simple produit matriciel fait l'affaire. Pour le premier facteur :

```
#coordonnées factorielles sur le premier facteur (n°0)
#produit matriciel
c1 = numpy.dot(Z,sol[1][:,0])

#affichage avec les identifiants des véhicules
print(pandas.DataFrame(c1,columns=['F_1'],index=D.index))
```

```
                     F_1
Modele
Alfasud TI       2.138924
Audi 100        -1.561459
Simca 1300       1.119385
Citroen GS Club  2.573742
Fiat 132        -0.427855
Lancia Beta      0.304238
Peugeot 504     -0.683928
Renault 16 TL    1.948493
Renault 30      -4.409735
Toyota Corolla   3.985782
Alfetta 1.66    -0.437658
Princess 1800   -1.018175
Datsun 200L     -2.941080
Taunus 2000     -1.314880
Rancho           0.691111
Mazda 9295      -0.385709
Opel Rekord     -2.289768
Lada 1300        2.708574
```

Présentation des résultats – Corrélation des variables avec les axes. Dans les logiciels, les (λ_k) apparaissent sous l'appellation « valeurs propres », on sait pourquoi. On utilise également le terme « variance expliquée ». Les vecteurs propres apparaissent rarement sous leur forme native en revanche. On préfère les présenter sous l'angle de la corrélation avec les variables où :

$$r_j(F_k) = \sqrt{\lambda_k} \times a_{jk}$$

Pour les données « Autos », le calcul est très simple.

```
#corrélation des variables avec le premier facteur
r_1 = numpy.sqrt(sol[0][0])*sol[1][:,0]

#affichage des corrélations
print(pandas.DataFrame(r_1,index=D.columns))

              0
CYL   -0.893464
PUISS -0.886858
LONG  -0.886155
LARG  -0.813536
POIDS -0.905187
VMAX  -0.754710
```

1.2.2 Calculs (2) – Décomposition en valeurs singulières

L'approche alternative pour l'ACP est de passer par la décomposition en valeurs singulières de la matrice des données centrées et réduites. Elle présente un double avantage par rapport à la précédente : elle ne nécessite pas le calcul de la matrice des corrélations, cela peut être décisif si nous traitons des données à très forte dimensionnalité ; elle montre bien le caractère dual de l'analyse, pédagogiquement je trouve cela intéressant.

Décomposition en valeurs singulières d'une matrice. La décomposition en valeurs singulières (SVD – singular value decomposition, en anglais) est une procédure de factorisation des matrices. Schématiquement, il s'agit de décomposer la matrice Z :

$$Z = U\,\Delta\,V^T$$

Avec :

$$\begin{cases} Z\,v_k = \delta_k\,u_k \\ Z^T u_k = \delta_k\,v_k \end{cases}$$

Sous Python, nous faisons appel à svd() de la librairie « Numpy ». Nous disposons de 3 éléments en sortie de la fonction.

```
#décomposition en valeurs singulières
Us, delta, Vs = numpy.linalg.svd(Z)
```

Variance restituée par les axes. **Δ** est une matrice diagonale dite des valeurs singulières (δ_k). On retrouve très facilement les variances restituées par les axes avec

$$\lambda_k = \frac{\delta_k^2}{n}$$

```
#vecteur des valeurs singulières
print(delta)

[8.92 3.93 2.59 1.96 1.29 0.88]

#et pour obtenir les lambda_k
print((delta**2)/n)

[4.42 0.86 0.37 0.21 0.09 0.04]
```

Coordonnées des variables. **V** est une matrice de taille (p, p), constituée d'un ensemble de vecteurs orthonormés dits « d'entrée ». Concrètement, nous dirons surtout qu'elle correspond à la matrice des vecteurs propres de (Z^TZ). Ainsi, nous disposons directement des résultats (a_{jk}) issus de la diagonalisation de la matrice des corrélations.

```
#matrice singulière à droite (d'entrée)
print(numpy.transpose(Vs))

[[ 0.42 -0.12  0.35 -0.81 -0.15  0.06]
 [ 0.42 -0.42  0.18  0.36  0.29  0.63]
 [ 0.42  0.41 -0.07  0.28 -0.73  0.19]
 [ 0.39  0.45 -0.6  -0.21  0.48  0.11]
 [ 0.43  0.24  0.48  0.3   0.3  -0.58]
 [ 0.36 -0.62 -0.49  0.07 -0.19 -0.46]]

#vérification -- vecteurs propres de (Z'Z) -- OK
print(numpy.linalg.eig(numpy.dot(numpy.transpose(Z),Z))[1])

[[-0.42 -0.12 -0.35  0.81  0.15 -0.06]
 [-0.42 -0.42 -0.18 -0.36 -0.29 -0.63]
 [-0.42  0.41  0.07 -0.28  0.73 -0.19]
 [-0.39  0.45  0.6   0.21 -0.48 -0.11]
 [-0.43  0.24 -0.48 -0.3  -0.3   0.58]
 [-0.36 -0.62  0.49 -0.07  0.19  0.46]]
```

Coordonnées des individus. **U** est de taille (n, n), constituée de vecteurs orthogonaux dits de « sortie ». Elle correspond à la matrice des vecteurs propres de (ZZ^T). Concrètement, en nous en tenant aux (p) premières colonnes, elle nous permet surtout de produire directement les coordonnées factorielles des individus :

$$F_{ik} = \delta_k \times u_{ik}$$

```
#matrice singulière à gauche (de sortie)
print(Us)

[[-0.24 -0.45 -0.22 -0.1   0.23 -0.06  0.31  0.21  0.17 -0.23 -0.2   0.14
   0.4  -0.03 -0.04  0.24  0.17 -0.3 ]
 [ 0.18  0.39 -0.51  0.11 -0.11  0.37  0.09  0.25  0.42 -0.06 -0.11  0.1

... etc. tout afficher n'a pas d'intérêt...

#coordonnées factorielles - 1er facteur
print(pandas.DataFrame(Us[:,0]*delta[0],columns=['F_1'],index=D.index))

                     F_1
Modele
Alfasud TI      -2.138924
Audi 100         1.561459
Simca 1300      -1.119385
Citroen GS Club -2.573742
Fiat 132         0.427855
Lancia Beta     -0.304238
Peugeot 504      0.683928
Renault 16 TL   -1.948493
Renault 30       4.409735
Toyota Corolla  -3.985782
Alfetta 1.66     0.437658
Princess 1800    1.018175
Datsun 200L      2.941080
Taunus 2000      1.314880
Rancho          -0.691111
Mazda 9295       0.385709
Opel Rekord      2.289768
Lada 1300       -2.708574
```

Remarque « diagonalisation vs. décomposition en valeurs singulières » : Il faut lire la documentation pour identifier l'algorithme utilisé dans les logiciels. Dans un exemple de text mining, avec de nombreux descripteurs, mes étudiants étaient très étonnés de voir planter princomp() du logiciel R, alors que prcomp() de la même libraire fonctionnait sans problème. Je leur ai demandé de lire le fichier d'aide pour comprendre. Voici ce qui est dit dans princomp() : « The calculation is done using eigen on the correlation or covariance matrix, as determined by cor. This is done for compatibility with the S-PLUS result. A preferred method of calculation is to use svd on x, as is done in prcomp. ». Et dans prcomp() : « The calculation is done by a singular value decomposition of the (centered and possibly scaled) data matrix, not by using eigen on the covariance matrix. This is generally the preferred method for numerical accuracy. The print method for these objects prints the results in a nice format and the plot method produces a scree plot. ». Passer par la décomposition en valeurs singulières fournit des résultats

numériquement plus précis dixit la documentation. Je sais par expérience que la stratégie est également plus efficace sur les très grandes bases.

1.3 Pratique de l'ACP avec « fanalysis » sous Python

Dans cette section, nous déroulons sous Python une étude type d'analyse en composantes principales sur les données « Autos ». Les thèmes abordés ne sont pas toujours traités dans cet ordre dans la pratique de l'ACP, mais ils sont plus ou moins incontournables pour exploiter pleinement les informations portées par les données. Les mêmes traitements menés à l'aide d'autres logiciels sont décrits dans des tutoriels dédiés : sous R (TUTO 4) ; sous Excel (TUTO 5) ; sous TANAGRA (TUTO 6). Il est toujours intéressant de faire le parallèle entre les différents outils et comparer le mode de présentation des résultats.

Nous utilisons le package « fanalysis » pour Python développé par Olivier Garcia, un de mes anciens étudiants du Master SISE (2000). Une courte description de ses fonctionnalités est disponible sur mon blog (« Analyses factorielles sous Python avec fanalysis », juin 2018). L'outil a une double finalité : il permet de mener une étude descriptive répondant à la pratique usuelle des méthodes factorielles ; il permet de les utiliser en tant que techniques de prétraitement précédant les algorithmes de machine learning dans des mécanismes de pipeline (« Pipeline sous Python – La méthode DISQUAL », juin 2018).

Des études plus réalistes sur des fichier plus importants (en taille) sont accessibles sur mon blog. Je conseille en particulier le tutoriel « ACP avec TANAGRA – Nouveaux outils » (TUTO 2, juin 2012) qui présente à peu près la même trame – sur un autre jeu de données – en faisant le parallèle avec les sorties de la PROC FACTOR du logiciel SAS.

Pour des raisons de lisibilité, nous recommençons à zéro dans cette section, avec le chargement des données et la préparation des structures. Nous suivons ensuite la démarche d'analyse proposée par l'auteur de la librairie (« Package fanalysis – Analyse en Composantes principales »).

Importation des données. Aucune difficulté intrinsèque ici, nous avons toujours (n = 18) observations et (p = 6) variables.

```
#chargement des données - index_col = 0 : indiquer que la colonne n°0 est un label
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="DATA_ACP_ACTIF",index_col=0)

#affichage des caractéristiques
print(D.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18 entries, Alfasud TI to Lada 1300
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   CYL     18 non-null     int64
 1   PUISS   18 non-null     int64
 2   LONG    18 non-null     int64
 3   LARG    18 non-null     int64
 4   POIDS   18 non-null     int64
 5   VMAX    18 non-null     int64
dtypes: int64(6)
memory usage: 1008.0+ bytes
```

```
#nombre de variables
p = D.shape[1]

#nombre d'observations
n = D.shape[0]

#matrice des X
X = D.values
```

ACP avec « fanalysis ». Pour lancer l'ACP nous importons la classe de calcul, nous instancions l'objet de calcul et nous lançons l'analyse.

```
#importer la classe de calcul
from fanalysis.pca import PCA

#instancier l'objet de calcul
#std_unit = True ➔ ACP normée
acp = PCA(std_unit=True,row_labels=D.index,col_labels=D.columns)

#lancer les calculs sur les données
acp.fit(X)

#propriétés de l'objet généré
print(dir(acp))
```

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format
__', '__ge__', '__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__
', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__
reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setstate__', '__sizeof__
', '__str__', '__subclasshook__', '__weakref__', '_compute_stats', '_compute_svd',
'_get_param_names', '_get_tags', '_more_tags', 'col_contrib_', 'col_coord_', 'col_
```

```
cor_', 'col_cos2_', 'col_labels', 'col_labels_', 'col_labels_short_', 'col_topanda
s', 'correlation_circle', 'eig_', 'eigen_vectors_', 'fit', 'fit_transform', 'get_p
arams', 'mapping', 'mapping_col', 'mapping_row', 'means_', 'model_', 'n_components
', 'n_components_', 'plot_col_contrib', 'plot_col_cos2', 'plot_eigenvalues', 'plot
_row_contrib', 'plot_row_cos2', 'row_contrib_', 'row_coord_', 'row_cos2_', 'row_la
bels', 'row_labels_', 'row_topandas', 'set_params', 'ss_col_coord_', 'stats', 'std
_', 'std_unit', 'transform']
```

L'objet possède une série de propriétés et de méthodes que nous exploiterons à l'envi tout au long de cette section.

1.3.1 Détermination du nombre de facteurs

Le choix du nombre de facteurs à retenir est très important en ACP (TUTO 3). L'enjeu est de distinguer d'une part l'information pertinente (le « signal ») véhiculée par les axes que l'on choisit de retenir ; et d'autre part, l'information résiduelle – le « bruit » issu des fluctuations d'échantillonnage – traduite par les derniers facteurs que l'on choisit de négliger. Et c'est là que le bât blesse. On nous dit généralement qu'il faut conserver les facteurs intéressants, pourvu qu'ils soient interprétables. Difficile d'être plus approximatif. Pour un expert, ce n'est pas vraiment un problème. Attention, l'expertise ne se limite pas à la méthode. Elle englobe une bonne connaissance du domaine (savoir ce qui est possible d'obtenir ou pas) et des données manipulées (pouvoir détecter rapidement les anomalies ou les évidences). Bref, il dispose de tous les garde-fous nécessaires pour produire des résultats valables.

Pour le néophyte, l'à peu près n'est pas gérable. C'est un peu le cas de mes étudiants. Ils connaissent bien les statistiques. Ils sont en train d'apprendre les techniques factorielles. Mais pour ce qui est des applications pratiques sur des fichiers de données, ils ne sont ni médecins, ni spécialistes du marketing, etc. L'expertise métier leur fait défaut pour encadrer les résultats. Ils ont besoin de repères numériques forts lorsqu'ils mènent une analyse. Et je me suis rendu compte que la très grande majorité des étudiants se contentaient très souvent de la règle de Kaiser-Guttman (valeur propre > 1 en ACP normée), et parfois de la règle du coude associée au « scree plot » mettant en lumière la décroissance des valeurs propres (« éboulis » des valeurs propres). Pourtant d'autres règles existent. Elles sont malheureusement peu connues, peu diffusées, et de ce fait peu utilisées.

Dans cette section, nous passons en revue quelques solutions simples à mettre en œuvre pour identifier le nombre adéquat de facteurs de l'ACP. Les techniques de rééchantillonnage qui impliquent plusieurs passages sur les données feront l'objet d'un développement spécifique dans un autre chapitre (section 2.1).

Ces approches « simples » s'appuient essentiellement sur l'étude des valeurs propres. Rappelons que la valeur propre correspond à la fraction d'inertie que la composante retranscrit. Plus elle est élevée, plus le facteur est important dans la lecture des résultats. L'enjeu justement est de déceler à partir de quel stade l'information restituée peut être considérée négligeable. L'affaire n'est pas facile. En effet, plusieurs éléments entrent en ligne de compte (Jackson, 1993) : le nombre d'observations « **n** » ; le nombre de variables « **p** » de l'analyse ; le ratio « **n**:**p** »entre le nombre d'observations et le nombre de variables ; le degré de liaison (la corrélation) entre les variables ; l'existence éventuelle de blocs de variables corrélées dans le tableau de données.

Le ratio « n:p » est particulièrement important. Il détermine la stabilité des résultats. Certaines références affirment qu'une ACP n'est vraiment viable que s'il est supérieur à 3 (Grossman et al., 1991). Nous avons juste (18/6 = 3) dans notre fichier. Ouf ! A priori, nous pouvons travailler en confiance.

1.3.1.1 Procédures graphiques

La règle de Cattell (1966, 1977) repose sur l'étude de la courbe de décroissance des valeurs propres (λ_k) pour identifier le bon nombre de facteurs. Elle est valable pour les ACP normées et non-normées. L'idée est de détecter les « coudes » (les « cassures ») signalant un changement de structure. Cette approche est intéressante parce qu'elle est nuancée. Elle permet de dépasser l'arbitraire purement numérique. Mais elle est compliquée à mettre en œuvre parce qu'elle est justement soumise à l'arbitraire. La détection n'est pas toujours évidente. Il faut répondre à plusieurs questions : Où est situé le coude ? Est-ce qu'il est unique ? Est-ce que nous l'incluons ou pas dans la sélection ?

En règle générale, le coude est très marqué lorsque nous traitons des variables fortement corrélées. Lorsqu'elles le sont faiblement ou lorsqu'il y a des blocs de variables corrélées, plutôt

qu'une solution unique « évidente », nous devons envisager plusieurs scénarios. Concernant l'intégration du coude dans la sélection, Cattel lui-même a varié. Dans un premier temps (Cattell, 1966), il conseillait de ne sélectionner que les facteurs qui sont avant le coude ; puis, dans un second temps (1977), il préconise finalement de l'intégrer. Tout dépend de la valeur associée au coude en réalité. Si elle est faible, on peut négliger le facteur. Nous devons le sélectionner en revanche si elle est élevée.

Sur les données « Autos », la propriété « .eig_ » fournit les valeurs propres de différentes manières : valeur brute, en proportion d'inertie expliquée, en proportion cumulée.

```
#valeurs propres
print(acp.eig_)
```

```
[[4.42085806e+00 8.56062289e-01 3.73066077e-01 2.13922089e-01
  9.28012120e-02 4.32902727e-02]
 [7.36809677e+01 1.42677048e+01 6.21776796e+00 3.56536815e+00
  1.54668687e+00 7.21504545e-01]
 [7.36809677e+01 8.79486725e+01 9.41664404e+01 9.77318086e+01
  9.92784955e+01 1.00000000e+02]]
```

Nous construisons la courbe, que l'on appelle aussi « Eboulis des valeurs propres » ou « Scree plot » selon les références. J'ai rajouté une ligne pointillée rouge dont nous négligeons la lecture pour l'instant (Figure 8).

```
%matplotlib inline

#librairie graphique
import matplotlib.pyplot as plt

#préparer le graphique
fig, ax = plt.subplots(figsize=(5,5))
ax.plot(range(1,p+1),acp.eig_[0],".-")
ax.set_xlabel("Nb. facteurs")
ax.set_ylabel("Val. propres")
plt.title("Eboulis des valeurs propres")

#rajout du seuil du Kaiser
ax.plot([1,6],[1,1],"r--",linewidth=1)

plt.show()
```

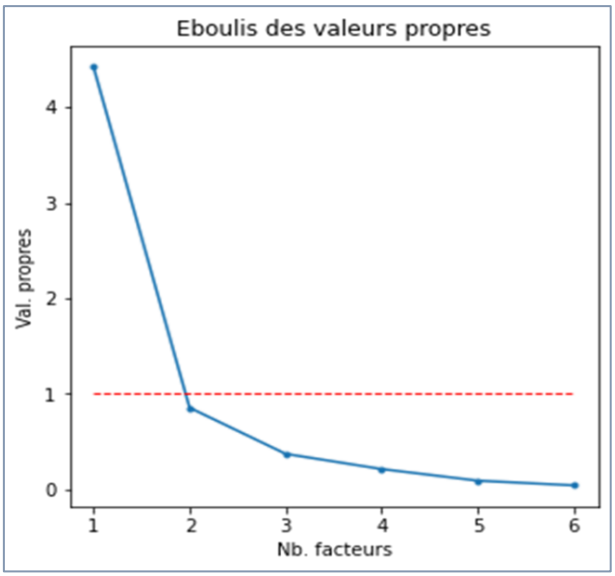


Figure 8 – Scree plot – Données "Autos"

Il y a un coude évident ici pour (k = 2). Est-ce qu'il faut l'inclure ou pas dans la sélection c.-à-d. est-ce qu'il faut prendre (q = 1) ou (q = 2) facteurs finalement ? Pour les graphiques, (q = 2) est incontournable. Mais de manière générale, la décision repose pour beaucoup sur la proportion d'inertie portée par le facteur concerné. Ici, elle semble assez élevée (14.27%), on serait tenté de la sélectionner nonobstant les contraintes de représentations graphiques.

Pour préciser la lecture, il semble intéressant de compléter le « scree plot » par un second graphique décrivant l'évolution de l'inertie expliquée par les (q) premiers axes. Il s'agit ni plus ni moins de la courbe des valeurs propres cumulées en pourcentage. Un « coude » devrait y être visible également. Mais la partie subséquente doit présenter une pente moindre, indiquant ainsi un apport négligeable des facteurs restants.

Nous réalisons ce second graphique pour les données « Autos » (Figure 9).

```
#proportion de variance expliquée
fig, ax = plt.subplots(figsize=(5,5))
ax.plot(range(0,p+1),numpy.append(0,acp.eig_[2]),".-")
ax.set_xlabel("Nb. facteurs")
ax.set_ylabel("% de var. expliquée")
plt.title("Variance expliquée")

plt.show()
```

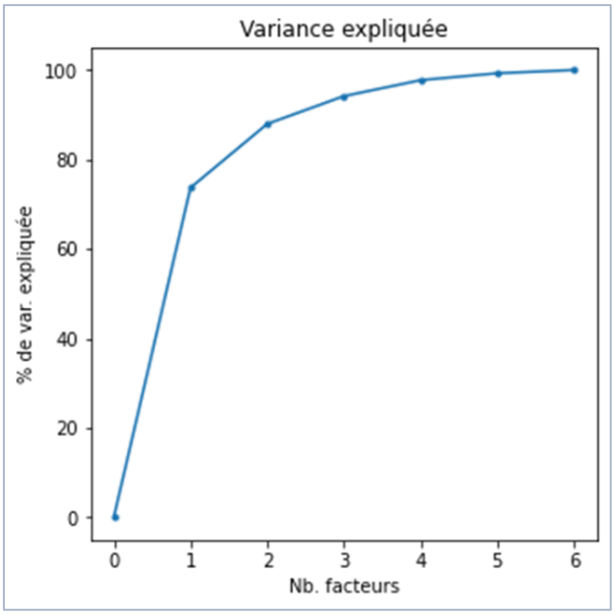


Figure 9 – Courbe de variance expliquée – Données "Autos"

Lorsque l'on passe de (q = 0) à (q = 1) facteur, le gain d'information est important, la pente de la courbe est forte. Il y a une cassure à (q = 1). Pour le passage de (q = 1) à (q = 2) facteurs, le gain est moindre certes, mais la pente reste conséquente. Il y a une seconde cassure en (q = 2), et par la suite, la pente est de plus en plus horizontale.

A l'évidence, le choix se joue entre (q = 1) ou (q = 2). Ce graphique de l'évolution de la variance expliquée en fonction du nombre de composantes combiné au « scree plot » se révèle souvent décisif dans l'appréciation du nombre de facteurs à retenir.

1.3.1.2 Proportion d'inertie expliquée

On est parfois tenté d'utiliser explicitement la part de variance expliquée pour déterminer le nombre de facteurs. La règle serait alors : « sélectionner suffisamment d'axes pour expliquer au moins x% de l'inertie totale ». Cette stratégie est descendue en flammes par toutes les références que j'ai pu consulter. Pour la simple raison qu'elle ne tient pas compte des corrélations entre les variables. Dans notre exemple, mettons que l'on souhaite capter 95% de l'information disponible, nous serions amenés à retenir (q = 3) facteurs. Tout en sachant pertinemment que la variabilité est très faible à partir du 3ème.

Néanmoins, après coup, après avoir choisi le nombre d'axes à l'aide d'une des approches décrites dans ce document, il est intéressant de pouvoir situer la quantité d'information – à l'aide de la fraction d'inertie expliquée - que restituent les facteurs sélectionnés.

1.3.1.3 Règles de Kaiser simples et améliorées

Règle de Kaiser-Guttman. La règle de Kaiser-Guttamn repose sur une idée simple. Dans une ACP normée, la somme des valeurs propres étant égale au nombre de variables, leur moyenne vaut 1. Nous considérons par conséquent qu'un axe est intéressant si sa valeur propre est supérieure 1.

Il existe d'autres manières de considérer ce seuil : un axe est intéressant s'il contribue plus qu'une des variables prise individuellement ; ou encore, si les variables étaient deux à deux orthogonales, les valeurs propres issues de l'analyse seraient toutes égales à 1.

Pour les données « Autos », le seuil a été rajoutée dans le « scree plot » (Figure 8, ligne rouge pointillée). Cette règle de sélection sert de repère à tous les praticiens de l'analyse en composantes principales. Elle a le mérite de la popularité. Mais elle a pour inconvénient de ne tenir compte en aucune manière les caractéristiques des données, notamment ses dimensions (nombre d'observations et de variables).

Règle de Karlis-Saporta-Spinaki. On pense généralement que le seuil 1 est trop permissif c.-à-d. nous retenons plus d'axes factoriels qu'il n'en faut. Il n'est réellement fondé que si les variables sont fortement corrélées, autrement il faudrait le relever. Une règle plus restrictive consiste à le définir comme suit : moyenne des valeurs propres + 2 fois leur écart-type (Saporta, 2006 ; page 172). Elle rappelle la définition de la valeur critique d'un test unilatéral de conformité à 5%, où la statistique suit asymptotiquement une loi normale.

La règle d'acceptation devient maintenant :

$$\lambda > \mathbf{1} + 2\sqrt{\frac{p-1}{n-1}}$$

Par rapport à la règle de Kaiser, elle est plus restrictive. Cela va dans le sens souhaité. Notons également qu'elle dépend du ratio « n:p », déterminant dans la qualité des résultats de l'ACP. Nous serons d'autant plus exigeants – nous serons enclins à accepter moins de facteurs - que le nombre de variables « p » est élevé par rapport aux observations disponibles « n ».

Pour nos données, le seuil devient :

$$1 + 2\sqrt{\frac{p-1}{n-1}} = 1 + 2\sqrt{\frac{6-1}{18-1}} = 2.08$$

```
#librairie pour calc. math
import math

#seuil de Karlis-Saporta-Spinaki
kss = 1+2*math.sqrt((p-1)/(n-1))
print(kss)
```

```
2.084652289093281
```

Seul le premier facteur échapperait au couperet.

1.3.1.4 Test de sphéricité de Bartlett

Bien qu'il existe une variante qui peut être utilisée pour identifier le nombre de facteurs pertinents (TUTO 3, section 5.2), le test de Bartlett sert avant tout à détecter l'existence de relations « intéressantes » entre les variables (TUTO 7). En effet, on peut considérer l'ACP comme une compression de l'information. Elle n'est possible que si les données présentent une certaine redondance. Si les variables sont deux à deux orthogonales, a fortiori si elles sont deux à deux indépendantes, il n'y a pas de compression possible, le nombre adéquat de facteurs à retenir est égal au nombre de variables. Dans ce dernier cas, la matrice des corrélations est égale à la matrice identité.

Formellement, l'hypothèse nulle (Ho) du test de sphéricité de Bartlett correspond à : « toutes les valeurs propres sont identiques, elles sont égales à 1 ». Si on rejette Ho, on sait qu'il existe au moins un facteur pertinent dans l'ACP. Nous ne pouvons pas spécifier leur nombre toutefois.

La statistique de test est basée sur le déterminant du coefficient de corrélation R, elle s'écrit

$$C = -\left(n - 1 - \frac{2\,p + 5}{6}\right) \times \ln|R|$$

Où R est la matrice des corrélations, |R| son déterminant.

Sous Ho, elle suit une loi du χ^2 à [p x (p – 1) / 2] degrés de liberté.

|R| est égal au produit des valeurs propres. Si les variables sont deux à deux orthogonales, il est égal à 1, et la statistique de test C est égale à 0. Il s'agit d'apprécier dans quelle mesure nous nous écartons de cette situation de référence.

Il n'est pas nécessaire de calculer explicitement la matrice de corrélation pour réaliser le test. La formule peut être réécrite en exploitant les valeurs propres fournies par l'objet ACP :

$$C = -\left(n - 1 - \frac{2\,p + 5}{6}\right) \times \sum_{k=1}^{p} \ln \lambda_k$$

Pour nos données :

```
#librairie pour calculs matriciels
import numpy

#librairie pour calculs statistiques
import scipy.stats as stats

#stat. de test Bartlett
C = -(n-1-(2*p+5)/6) * numpy.sum(numpy.log(acp.eig_[0]))
print(C)
```

```
95.11987830353813
```

```
#degrés de liberté
ddl = p*(p-1)/2

#p-value
print(1-stats.chi2.cdf(C,df=ddl))
```

```
1.0891287871572786e-13
```

Manifestement, il y a de l'information exploitable à revendre dans les données « Autos ».

Il faut quand même prendre avec prudence ce test. Il a tendance à considérer toute configuration significative à mesure que la taille de l'échantillon « n » augmente. Ce n'est guère étonnant, « n » intervient dans la statistique mais pas dans le calcul des degrés de liberté. De fait, on

recommande (Dugard et al., 2010) de ne l'utiliser que lorsque le ratio « n:p » (nombre d'observations sur nombre de variables) est (approximativement) inférieur à 5 c.-à-d. lorsque nous avons moins de 5 observations par variable.

1.3.1.5 Test des « bâtons brisés »

Ce test est dû à Frontier (1976) et Legendre-Legendre (1983). Il repose sur l'idée que si l'inertie totale était dispatchée aléatoirement sur les axes, la distribution des valeurs propres suivrait la loi des « bâtons brisés » (broken-stick). Elle a été tabulée semble-t-il. Mais il est très facile de calculer la valeur critique pour le choix de « k » facteurs pour un test à 5%. Elle s'écrit :

$$b_k = \sum_{i=k}^{p} \frac{1}{i}$$

Pour tester le 1er facteur pour les données « Autos » où (p = 6), la valeur seuil est :

$$b_1 = \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{6} = 2.45$$

Pour le 2nd facteur,

$$b_2 = \frac{1}{2} + \cdots + \frac{1}{6} = 1.45$$

Nous pouvons ainsi reconstruire le « Scree plot » en y faisant figurer les seuils définis par le test des bâtons brisés.

```
#seuils pour test des bâtons brisés
b = numpy.flip(numpy.cumsum(1/numpy.arange(p,0,-1)))
print(b)
```

```
[2.45       1.45       0.95       0.61666667 0.36666667 0.16666667]
```

```
#vérification des v.p. qui passent
fig, ax = plt.subplots(figsize=(5,5))
ax.plot(range(1,p+1),acp.eig_[0],".-")
ax.set_xlabel("Nb. facteurs")
ax.set_ylabel("Val. propres")
plt.title("Val. propres et seuils des bâtons brisés")

#rajout du seuil des bâtons brisés
ax.plot(range(1,p+1),b,"r--",linewidth=1)
```

```
plt.show()
```

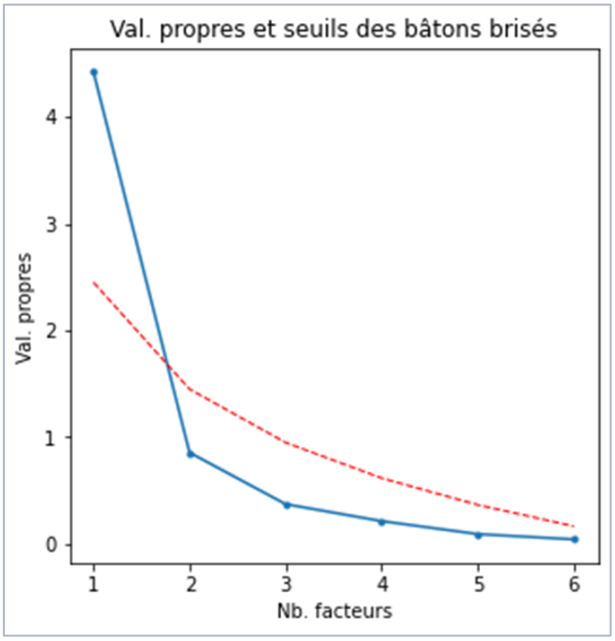


Figure 10 – Valeurs propres et seuils des bâtons brisés – Données "Autos"

Ainsi, la procédure des bâtons brisés nous annonce que seul le 1er facteur est réellement pertinent pour décrire les données.

Le test des bâtons brisés a plutôt bonne réputation. On lui reproche simplement de ne tenir compte ni de « n » (le nombre d'observations) ni du ratio « n:p » dans la définition des valeurs critiques (TUTO 3). Elle n'est valable que pour l'ACP normée.

1.3.1.6 Conclusion sur la détection du nombre de facteurs

Il n'y a pas de solution miracle pour l'identification du nombre de facteurs. Mis à part le test de Bartlett qui répond à un cahier de charges très spécifique, les différentes techniques présentées se valent et apportent des points de vue complémentaires dans notre pratique de l'ACP. Il n'y a pas de vérités absolues en statistique exploratoire. Il y a seulement des pistes que nous devons étudier attentivement en le reliant aux caractéristiques (objectifs, contexte) de notre étude et de nos données. L'expertise métier est essentielle pour valider les résultats. Il y a aussi une part d'intuition qui tient à l'expérience du data scientist. Sur les données « Autos », il faut en réalité sélectionner (q = 2) composantes… qui deviennent évidentes : soit après rotation des facteurs (section 2.2), soit en travaillant sur les corrélations partielles (section 2.4).

1.3.2 Représentation des variables et aides à l'interprétation

Les composantes représentent les « dimensions » que recèlent les données que nous analysons. Pour comprendre leur nature, il faut pouvoir les mettre en relation avec les variables qui pèsent plus ou moins dans leur construction.

Corrélation. La corrélation des variables avec les composantes – qui correspond aussi à leurs coordonnées - est un outil privilégié pour l'interprétation du repère factoriel. Il n'est pas nécessaire de les calculer ex-post, nous pouvons les déduire directement des valeurs et vecteurs propres :

$$r_j(F_k) = \sqrt{\lambda_k} \times a_{jk}$$

Nous affichons les corrélations sur les (q = 2) premières composantes de l'ACP sur « Autos ».

```
#corrélation des variables avec les facteurs
print(pandas.DataFrame(acp.col_coord_[:,:2],index=D.columns))
```

```
              0         1
CYL    0.893464 -0.114906
PUISS  0.886858 -0.384689
LONG   0.886155  0.381029
LARG   0.813536  0.412736
POIDS  0.905187  0.224532
VMAX   0.754710 -0.573519
```

Nous observons :

- Que toutes les variables déterminent plus ou moins le 1er facteur, avec des corrélations élevées (toutes positives ici). On parle d'effet taille dans ce type de situation. Une des variables pèse sur l'ensemble des autres et masque les différences. Nous en reparlerons plus loin (section 2.4). En l'état, on peut y voir une opposition entre les citadines (« petites » voitures à faible encombrement avec un moteur réduit) et les routières (« grosses » voitures à forte cylindrée). Le facteur traduit la « gamme » des véhicules.
- Pour le 2nd facteur, vu la corrélation avec VMAX, on peut le voir comme reflétant la « sportivité » des véhicules. On pourrait être amené à penser qu'il y a une opposition entre (vitesse maximum) d'une part, (largeur) d'autre part. En réalité, il faut être circonspect parce que la lecture dépend de la position sur le premier facteur. En investiguant deux

secondes, nous constatons que c'est une fausse impression que l'on ne retrouve absolument dans les nuages de points des variables prises deux à deux (Figure 3).

Pour une meilleure lisibilité, on privilégie une vue synthétique avec le « **cercle des corrélations** ». Ce sont les « directions » qui importent, d'où les flèches dans la représentation graphique. Elles permettront de comprendre les positions relatives des observations dans la « carte des individus » (section 1.3.3).

```
#cercle des corrélations
acp.correlation_circle(num_x_axis=1,num_y_axis=2)
```

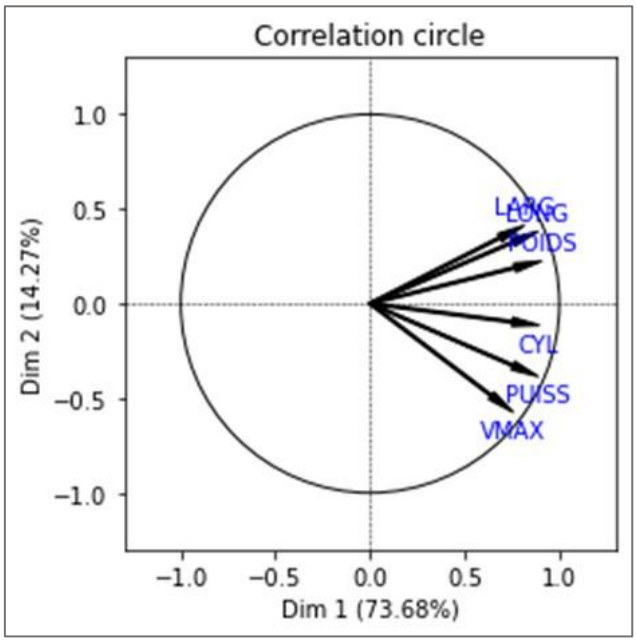


Figure 11 – Cercle des corrélations – Données "Autos"

Clairement ici, la cylindrée joue un rôle central (c'est le cas de le dire) en étant corrélée avec les autres variables.

Remarque : Nous sommes dans une situation simple dans notre exemple. Les (q = 2) premiers facteurs suffisent pour décrire les données. Sinon, il faudrait considérer les représentations factorielles par paires de composantes. La gymnastique est autrement plus complexe.

Cosinus². Le COS2 désigne la qualité de la représentation des variables (x_j) sur une composante. Elle correspond au carré du coefficient de corrélation, elle est définie entre 0 et 1 :

$$COS_{jk}^2 = r_j^2(F_k)$$

Il présente l'avantage de s'additionner d'un facteur à l'autre. Pour indiquer l'information de la variable (x_j) restituée sur les (q) premiers facteurs, nous faisons :

$$\sum_{k=1}^{q} COS_{jk}^2$$

Et lorsque nous considérons l'ensemble des (p) facteurs, nous avons capté toute l'information portée par une variable :

$$\sum_{k=1}^{p} COS_{jk}^2 = 1$$

Dans l'exemple « Autos », mis à part VMAX et LARG, toutes les variables sont bien représentées dès le 1er facteur. Nous avons épuisé quasiment toute l'information portée par chaque variable dès le 2nd facteur puisque, dans le pire des cas, nous avons au moins restitué **81%** de l'information disponible (CYL).

```
#cos2 des variables sur les 2 facteurs
print(pandas.DataFrame(acp.col_cos2_[:,:2],index=D.columns))

              0         1
CYL    0.798277  0.013203
PUISS  0.786517  0.147986
LONG   0.785270  0.145183
LARG   0.661841  0.170351
POIDS  0.819364  0.050415
VMAX   0.569588  0.328925

#cos2 cumulé sur les 2 premiers facteurs
print(pandas.DataFrame(numpy.cumsum(acp.col_cos2_[:,:2],axis=1),index=D.columns))

              0         1
CYL    0.798277  0.811481
PUISS  0.786517  0.934503
LONG   0.785270  0.930453
LARG   0.661841  0.832192
POIDS  0.819364  0.869779
VMAX   0.569588  0.898512
```

Contributions. Les contributions matérialisent l'influence d'une variable dans la définition de chaque composante. Elles sont basées également sur le carré des corrélations, mais normalisées afin que leurs sommes sur chaque composante soit égale à 1.

Sachant que :

$$\sum_{j=1}^{p} r_j^2(F_k) = \lambda_k$$

La contribution de la variable (x_j) au facteur (F_k) s'écrit :

$$CTR_{jk} = \frac{r_j^2(F_k)}{\lambda_k}$$

Voici les contributions aux deux premiers facteurs des données « Autos », exprimées en pourcentage. La somme en colonne fait bien 100%.

```
#contributions des variables sur les 2 premiers facteurs (en %)
print(pandas.DataFrame(acp.col_contrib_[:,:2],index=D.columns))
```

```
               0          1
CYL    18.057062   1.542342
PUISS  17.791052  17.286793
LONG   17.762847  16.959384
LARG   14.970882  19.899361
POIDS  18.534057   5.889155
VMAX   12.884099  38.422964
```

Pas de surprise ici, les variables les plus corrélées avec les facteurs sont celles qui pèsent le plus dans leur définition.

Remarque 1 : En analyse en composantes principales, les COS2 et CTR apportent peu dans l'interprétation des résultats par rapport à la corrélation des variables aux facteurs. En effet, cette dernière étant bornée entre -1 et +1, elle permet déjà des comparaisons directes.

Remarque 2 : Lors de la reconstitution des corrélations dans l'espace factoriel (section 1.1.3.4), l'approximation sera d'autant meilleure que les COS2 des variables traitées seront élevés.

1.3.3 Représentation des individus et aides à l'interprétation

Dans la culture francophone de l'analyse des données, la popularité de l'analyse en composantes principales repose pour beaucoup sur les cartes des individus. Je trouve pour ma part assez addictif de pouvoir positionner des objets dans un repère, pouvoir comprendre les proximités et les oppositions en les reliant à leurs (multiples) caractéristiques, surtout si l'on dispose par ailleurs de quelques connaissances du domaine. Pour l'exemple des données « Autos », le féru des automobiles des années 70 que je suis comprend intuitivement pourquoi les voitures

italiennes (de ces années-là !) sont regroupées dans une zone spécifique du plan factoriel, pourquoi la Renault 30 est si différente, etc. Pouvoir associer ces connaissances a priori à des descriptions objectives est jouissif.

Coordonnées des individus. Les coordonnées des individus peuvent être obtenues : soit en appliquant les vecteurs propres de la matrice des corrélations aux valeurs centrées et réduites (Z) (section 1.2.1) ; soit en pondérant les vecteurs singuliers à gauche U (left-singular vectors) par les valeurs singulières lors de la décomposition en valeurs singulières de la matrice Z (section 1.2.2). Elles correspondent au champ « .**row_coord_** » de notre objet ACP.

```
#coordonnées factorielles des individus dans le plan
print(pandas.DataFrame(acp.row_coord_[:,:2],index=D.index))
```

```
                        0         1
Modele
Alfasud TI      -2.138924 -1.785681
Audi 100         1.561459  1.527040
Simca 1300      -1.119385  0.674505
Citroen GS Club -2.573742 -0.112884
Fiat 132         0.427855 -0.695567
Lancia Beta     -0.304238  0.196149
Peugeot 504      0.683928  0.933057
Renault 16 TL   -1.948493  0.980448
Renault 30       4.409735 -1.063633
Toyota Corolla  -3.985782 -0.236240
Alfetta 1.66     0.437658 -1.912448
Princess 1800    1.018175  0.841712
Datsun 200L      2.941080  0.559175
Taunus 2000      1.314880 -0.486522
Rancho          -0.691111  0.897721
Mazda 9295       0.385709 -0.356185
Opel Rekord      2.289768 -0.104345
Lada 1300       -2.708574  0.143699
```

Nous nous empressons de réaliser la représentation graphique des individus dans le plan factoriel. Une fonction dédiée réalise le travail pour nous.

```
#carte des individus
acp.mapping_row(num_x_axis=1,num_y_axis=2,figsize=(10,10))
```

Attention ! Les librairies graphiques procèdent quasiment toujours à une mise à l'échelle automatique pour que les points « remplissent » au mieux le canevas de présentation. De fait, la lecture doit tenir compte des proportions de variance expliquées en abscisse (73.68%) et en ordonnée (14.27%) (Figure 12) pour éviter les conclusions intempestives.

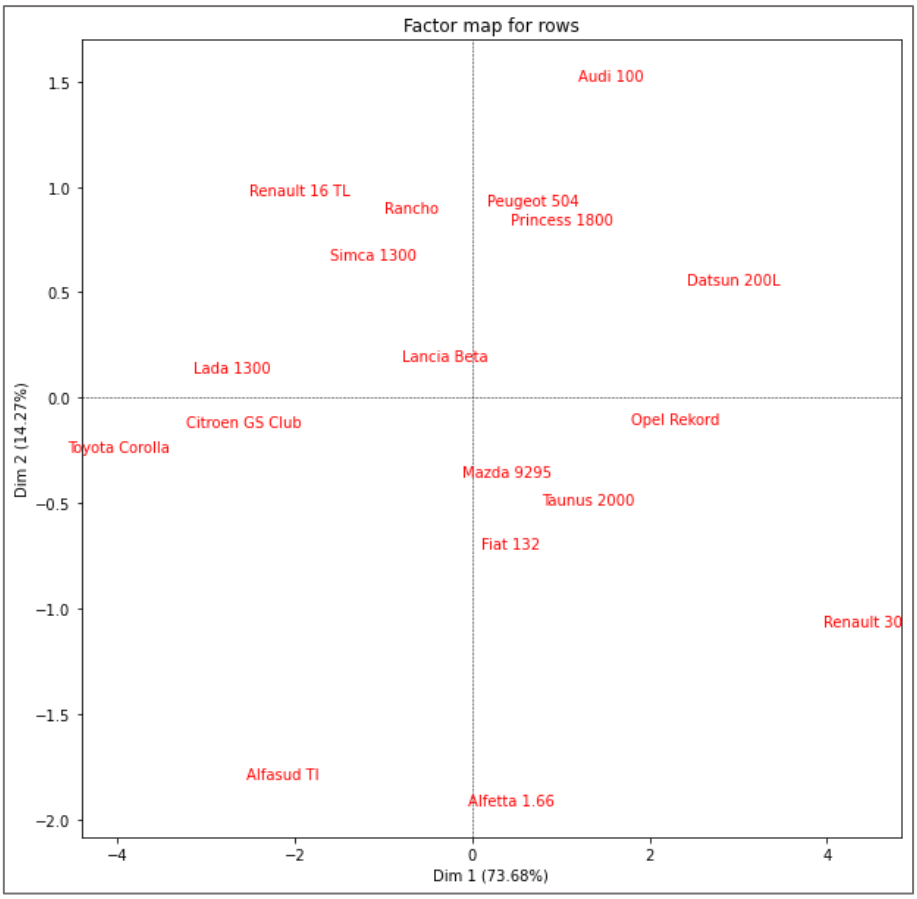


Figure 12 – Carte des individus – Version 1 – Données "Autos"

Sans ces indications, nous serions tentés de dire que :

a. l'écart entre la Simca 1300 et la Lancia Beta se joue plus sur l'ordonnée (Dim 2) que sur l'abscisse (Dim 1) ;
b. la Peugeot 504 est à égale distance de l'Audi 100 et de la Datsun 200L.

Or, ces deux assertions s'avèrent fausses si l'on procède aux calculs à partir des coordonnées factorielles. En effet (les coordonnées sont visibles dans la sortie Python ci-dessus) :

a. $d^2(Simca, Lancia) = \big(-1.12 - (-0.30)\big)^2 + (0.67 - 0.20)^2 = 0.66 + 0.23 = 0.89$; l'écart se fait plus sur le 1er facteur que sur le 2nd en réalité.
b. $d^2(Audi, 504) = (1.56 - 0.68)^2 + (1.53 - 0.93)^2 = 1.12$ et $d^2(Datsun, 504) = (2.94 - 0.68)^2 + (0.56 - 0.93)^2 = 5.23$; la Datsun s'avère nettement plus éloignée de la 504 que ne l'est l'Audi 100, on le perçoit visuellement uniquement si l'on a intégré que la dispersion est plus forte sur le 1er facteur.

C'est pour cette raison que je préconise systématiquement de fixer la même échelle en abscisse et en ordonnée des graphiques factoriels. Les positions relatives sont alors ramenées à de plus justes proportions, et reflètent plus fidèlement les écarts et leur nature (Figure 13).

```
#graphique avec les échelles carrées
fig, ax = plt.subplots(figsize=(10,10))
ax.plot(acp.row_coord_[:,0],acp.row_coord_[:,1],'wo')
ax.axis([-5,+5,-5,+5])
ax.plot([-5,+5],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-5,+5],color='silver',linestyle='--')
ax.set_xlabel("Dim.1 (73.68%)")
ax.set_ylabel("Dim.2 (14.27%)")
plt.title("Carte des individus")

for i in range(n):
    ax.text(acp.row_coord_[i,0],acp.row_coord_[i,1],D.index[i])

plt.show()
```

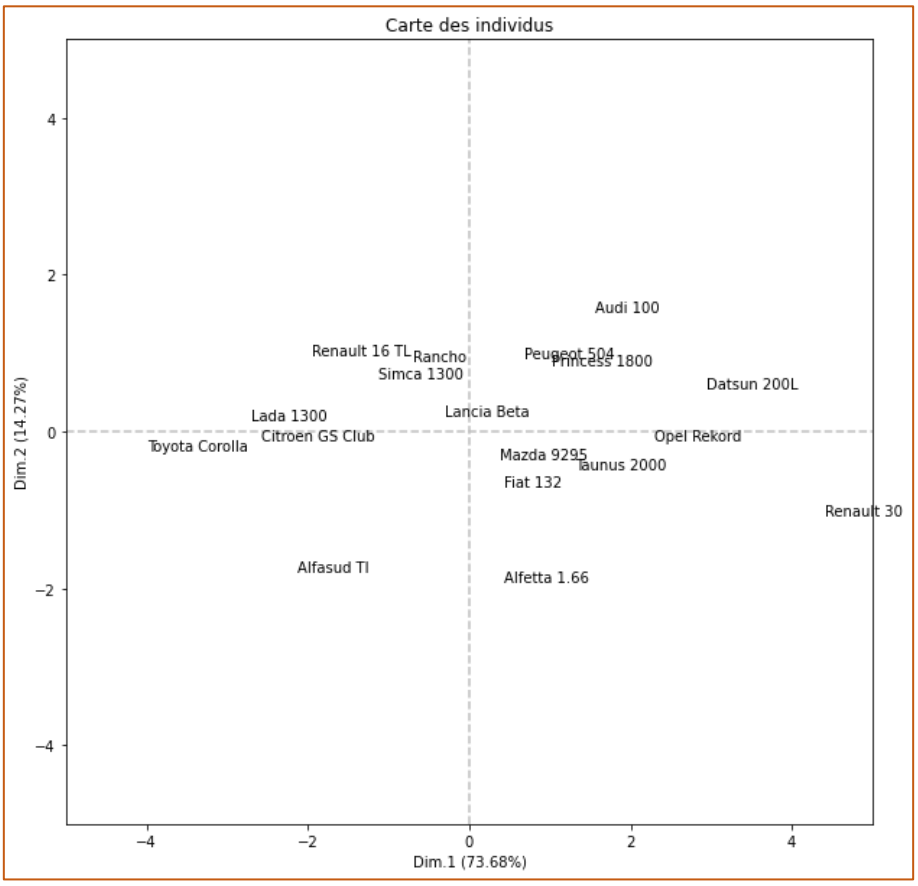


Figure 13 – Carte des individus – Version 2 – Données "Autos"

Cosinus² (COS2). Les cosinus expriment la qualité de représentation des individus. Ils sont basés sur le carré des coordonnées mais normalisées d'une manière spécifique. Pour les comprendre, intéressons-nous tout d'abord au concept de distance à l'origine.

Rappelons que l'inertie totale du nuage de points s'écrit :

$$I_p = \frac{1}{n}\sum_{i=1}^{n} d^2(i,G)$$

Nous travaillons sur (Z) qui est centrée, et éventuellement réduite, le barycentre et l'origine sont confondus, l'inertie devient :

$$I_p = \frac{1}{n}\sum_{i=1}^{n} d_i^2 = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p} z_{ij}^2$$

La quantité (d_i^2) indique la part de l'individu dans l'inertie totale, dans l'espace des variables centrées et éventuellement réduites (z_j) : c'est le carré de la distance à l'origine.

Notons une équivalence très importante : nous pouvons reconstituer cette distance dans l'espace factoriel si nous prenons tous les (p) facteurs.

$$d_i^2 = \sum_{k=1}^{p} F_{ik}^2$$

Le cosinus^2 de l'individu n°i sur le facteur (F_k), qui désigne la qualité de représentation de l'individu, est défini par le rapport :

$$COS_{ik}^2 = \frac{F_{ik}^2}{d_i^2}$$

Un individu est parfaitement représenté lorsque nous prenons en compte l'ensemble des (p) composantes. Il vient la naturellement,

$$\sum_{k=1}^{p} COS_{ik}^2 = 1$$

En pratique, on s'intéressera à la somme cumulée sur les (q) premiers facteurs pour savoir si un individu est bien représenté dans l'espace factoriel que l'on s'est choisi.

$$COS^2_{i(q)} = \sum_{k=1}^{q} COS^2_{ik}$$

Pour les COS2, je conseille habituellement aux étudiants de regarder en priorité les individus qui présentent une valeur anormalement faible (par rapport aux autres) sur le cumul des (q) facteurs analysés. Cela indique qu'ils présentent des particularités qui les distinguent de la « tendance » générale. Il est toujours intéressant de savoir pourquoi.

Dans le code Python qui suit, nous calculons tout d'abord la distance à l'origine de chaque individu de deux manières : dans l'espace des variables centrées et réduites, dans l'espace factoriel incluant tous les facteurs.

```
#données cenrrées et réduites
Z = (D.values - acp.means_) / acp.std_

#distance à l'origine des individus dans le repère initial
disto = numpy.apply_along_axis(arr=Z,axis=1,func1d=lambda x: numpy.sum(x**2))

#distance à l'origine à partir des coordonnées factorielles
distoBis = numpy.sum(acp.row_coord_**2,axis=1)

#affichage de contrôle
print(pandas.DataFrame(numpy.transpose(numpy.array([disto,distoBis])),index=D.index))
```

```
                         0          1
Modele
Alfasud TI        8.225176   8.225176
Audi 100          6.673755   6.673755
Simca 1300        2.159327   2.159327
Citroen GS Club   6.780145   6.780145
Fiat 132          1.169124   1.169124
Lancia Beta       1.134950   1.134950
Peugeot 504       1.512793   1.512793
Renault 16 TL     5.636826   5.636826
Renault 30       21.789657  21.789657
Toyota Corolla   16.290143  16.290143
Alfetta 1.66      4.456770   4.456770
Princess 1800     1.952513   1.952513
Datsun 200L      11.112624  11.112624
Taunus 2000       2.452986   2.452986
Rancho            1.963373   1.963373
Mazda 9295        0.684521   0.684521
Opel Rekord       6.083119   6.083119
Lada 1300         7.922198   7.922198
```

Les résultats sont cohérents, c'est toujours rassurant. Au-delà du COS2 qui nous calculerons tout de suite après, nous remarquons le rôle important de la Renault 30, la Toyota Corolla et

Datsun 200L. C'est sans surprise que nous les retrouvons aux extrémités de la première composante principale.

Nous formons les COS2 et nous comparons avec ceux fournis directement par la classe PCA.

```
#cos2 des individus
lig_cos2 = numpy.apply_along_axis(arr=acp.row_coord_[:,:2],axis=0,func1d=lambda x: (x**2)/disto)
print(pandas.DataFrame(lig_cos2,index=D.index))
```

```
                        0         1
Modele
Alfasud TI       0.556218  0.387670
Audi 100         0.365334  0.349406
Simca 1300       0.580284  0.210694
Citroen GS Club  0.976992  0.001879
Fiat 132         0.156579  0.413826
Lancia Beta      0.081555  0.033900
Peugeot 504      0.309202  0.575488
Renault 16 TL    0.673539  0.170535
Renault 30       0.892431  0.051920
Toyota Corolla   0.975219  0.003426
Alfetta 1.66     0.042978  0.820652
Princess 1800    0.530947  0.362855
Datsun 200L      0.778390  0.028137
Taunus 2000      0.704819  0.096496
Rancho           0.243273  0.410469
Mazda 9295       0.217336  0.185337
Opel Rekord      0.861900  0.001790
Lada 1300        0.926052  0.002607
```

```
#affichage de contrôle : COS2 de l'outil PCA
print(pandas.DataFrame(acp.row_cos2_[:,:2],index=D.index))
```

```
                        0         1
Modele
Alfasud TI       0.556218  0.387670
Audi 100         0.365334  0.349406
Simca 1300       0.580284  0.210694
Citroen GS Club  0.976992  0.001879
Fiat 132         0.156579  0.413826
Lancia Beta      0.081555  0.033900
Peugeot 504      0.309202  0.575488
Renault 16 TL    0.673539  0.170535
Renault 30       0.892431  0.051920
Toyota Corolla   0.975219  0.003426
Alfetta 1.66     0.042978  0.820652
Princess 1800    0.530947  0.362855
Datsun 200L      0.778390  0.028137
Taunus 2000      0.704819  0.096496
Rancho           0.243273  0.410469
Mazda 9295       0.217336  0.185337
Opel Rekord      0.861900  0.001790
Lada 1300        0.926052  0.002607
```

Le cumul sur les deux premiers facteurs est plus intéressant puisque nous avons fait le choix de travailler dans le premier plan factoriel.

```
#cumul sur les 2 premiers facteurs
print(pandas.DataFrame(numpy.cumsum(acp.row_cos2_[:,:2],axis=1),index=D.index))
```

```
                         0         1
Modele
Alfasud TI        0.556218  0.943889
Audi 100          0.365334  0.714741
Simca 1300        0.580284  0.790978
Citroen GS Club   0.976992  0.978871
Fiat 132          0.156579  0.570405
Lancia Beta       0.081555  0.115454
Peugeot 504       0.309202  0.884690
Renault 16 TL     0.673539  0.844075
Renault 30        0.892431  0.944351
Toyota Corolla    0.975219  0.978645
Alfetta 1.66      0.042978  0.863630
Princess 1800     0.530947  0.893802
Datsun 200L       0.778390  0.806527
Taunus 2000       0.704819  0.801315
Rancho            0.243273  0.653742
Mazda 9295        0.217336  0.402674
Opel Rekord       0.861900  0.863690
Lada 1300         0.926052  0.928659
```

La Lancia Beta ne suit pas la tendance générale visiblement, seule 11% de l'information qu'elle véhicule (hum !) est captée par les (q=2) premiers facteurs. En revenant au tableau de données, on se rend compte qu'elle est assez particulière par rapport aux autres : son petit moteur (cylindrée) n'est pas vraiment en phase avec ce que l'on attendrait en termes de puissance, vitesse maximale, où on la situerait plutôt parmi les berlines moyennes.

Contributions (CTR). Les contributions sont également basées sur le carré des coordonnées factorielles, mais normalisées par la variance restituée (au facteur $\frac{1}{n}$ près) pour que leur somme par composante soit égale à 1. Elles expriment l'importance relative des individus dans la construction des facteurs.

$$CTR_{ik} = \frac{F_{ik}^2}{n \times \lambda_k}$$

Avec

$$\sum_{i=1}^{n} CTR_{ik} = 1$$

Dans le cas des contributions, notre œil devrait être attiré par les valeurs fortes. Elles indiquent les individus déterminants dans la lecture des composantes. Ils permettent d'asseoir les interprétations. On devra se méfier néanmoins des influences anormalement élevées, elles sont plutôt symptomatiques d'observations atypiques susceptibles de fausser les résultats.

Nous calculons les contributions sur les 2 premiers facteurs (en pourcentage), puis nous comparons de nouveau les résultats avec ceux de l'outil PCA. Tout va pour le mieux.

```
#contributions sur les 2 premiers facteurs
lig_ctr = (acp.row_coord_[:,:2]**2)/(n*acp.eig_[0][:2])*100
print(pandas.DataFrame(lig_ctr,index=D.index))
```

```
                         0          1
Modele
Alfasud TI        5.749254  20.693307
Audi 100          3.063951  15.132933
Simca 1300        1.574636   2.952519
Citroen GS Club   8.324360   0.082697
Fiat 132          0.230046   3.139789
Lancia Beta       0.116318   0.249686
Peugeot 504       0.587817   5.649867
Renault 16 TL     4.771099   6.238372
Renault 30       24.436884   7.341856
Toyota Corolla   19.964025   0.362185
Alfetta 1.66      0.240708  23.735669
Princess 1800     1.302765   4.597791
Datsun 200L      10.870129   2.029163
Taunus 2000       2.172668   1.536130
Rancho            0.600229   5.230043
Mazda 9295        0.186956   0.823327
Opel Rekord       6.588764   0.070658
Lada 1300         9.219391   0.134007
```

```
#vérification avec l'outil PCA
print(pandas.DataFrame(acp.row_contrib_[:,:2],index=D.index))
```

```
                         0          1
Modele
Alfasud TI        5.749254  20.693307
Audi 100          3.063951  15.132933
Simca 1300        1.574636   2.952519
Citroen GS Club   8.324360   0.082697
Fiat 132          0.230046   3.139789
Lancia Beta       0.116318   0.249686
Peugeot 504       0.587817   5.649867
Renault 16 TL     4.771099   6.238372
Renault 30       24.436884   7.341856
Toyota Corolla   19.964025   0.362185
Alfetta 1.66      0.240708  23.735669
Princess 1800     1.302765   4.597791
Datsun 200L      10.870129   2.029163
Taunus 2000       2.172668   1.536130
Rancho            0.600229   5.230043
Mazda 9295        0.186956   0.823327
```

```
Opel Rekord         6.588764    0.070658
Lada 1300           9.219391    0.134007
```

L'outil propose un affichage graphique des (nb_values) contributions les plus importantes selon les facteurs (num_axis) (Figure 14). Solution pratique lorsque la taille du jeu de données est importante. Dans une étude réelle, il faudrait se poser des questions sur le rôle de la Renault 30. C'est la seule routière du lot finalement, la seule équipée d'un moteur V6 de surcroît.

```
#affichage graphique - 1er facteur
acp.plot_row_contrib(num_axis=1,nb_values=5)
```

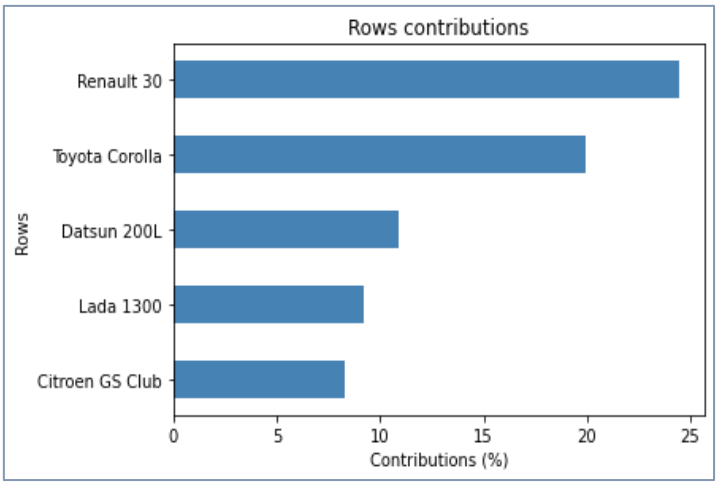


Figure 14 – 5 plus fortes contributions au premier facteur de l'ACP – Données "Autos"

Remarque : Nous comprenons aisément au regard des formules que les individus situés aux extrémités des facteurs sont les mieux représentés (COS2) et sont les plus influents (CTR).

1.3.4 Relation de transition

Puisque l'analyse en composantes principales est duale par nature, il existe une relation entre les coordonnées factorielles des individus (F_{ik}) et celles des variables [les corrélations, $r_j(F_k)$] via les formules dites de transition.

Pour obtenir les coordonnées de l'individu n°i sur le facteur (F_k) à partir des coordonnées des variables, la transition s'écrit :

$$F_{ik} = \frac{1}{\sqrt{\lambda_k}} \sum_{j=1}^{p} z_{ij} \times r_j(F_k)$$

Nous vérifions cela très facilement sous Python. Les coordonnées ainsi calculées pour le 1er facteur correspondent parfaitement aux coordonnées des individus fournies par l'outil PCA.

```
#coordonnées des individus à partir des variables - Facteur 1
f1 = 1/math.sqrt(acp.eig_[0][0]) * numpy.dot(Z,acp.col_coord_[:,0])

#affichage + vérification avec coordonnées de PCA
print(pandas.DataFrame(numpy.transpose(numpy.array([f1,acp.row_coord_[:,0]])),index=D.index))
```

```
                      0         1
Modele
Alfasud TI      -2.138924 -2.138924
Audi 100         1.561459  1.561459
Simca 1300      -1.119385 -1.119385
Citroen GS Club -2.573742 -2.573742
Fiat 132         0.427855  0.427855
Lancia Beta     -0.304238 -0.304238
Peugeot 504      0.683928  0.683928
Renault 16 TL   -1.948493 -1.948493
Renault 30       4.409735  4.409735
Toyota Corolla  -3.985782 -3.985782
Alfetta 1.66     0.437658  0.437658
Princess 1800    1.018175  1.018175
Datsun 200L      2.941080  2.941080
Taunus 2000      1.314880  1.314880
Rancho          -0.691111 -0.691111
Mazda 9295       0.385709  0.385709
Opel Rekord      2.289768  2.289768
Lada 1300       -2.708574 -2.708574
```

Inversement, il est possible d'obtenir les coordonnées des variables $r_j(F_k)$ à partir de celles des individus (F_{ik}) :

$$r_j(F_k) = \frac{1}{n\sqrt{\lambda_k}} \sum_{i=1}^{n} z_{ij} \times F_{ik}$$

De nouveau sous Python, toujours pour le 1er facteur,...

```
#coordonnées des variables à partir des individus
r1 = 1/(n*math.sqrt(acp.eig_[0][0])) * numpy.dot(numpy.transpose(Z),acp.row_coord_[:,0])

#affichage + vérification avec corrélations de PCA
print(pandas.DataFrame(numpy.transpose(numpy.array([r1,acp.col_coord_[:,0]])),index=D.columns))
```

```
              0         1
CYL    0.893464  0.893464
PUISS  0.886858  0.886858
LONG   0.886155  0.886155
LARG   0.813536  0.813536
POIDS  0.905187  0.905187
VMAX   0.754710  0.754710
```

Ces relations légitiment la lecture conjointe des cartes des variables et des individus c.-à-d. pour expliquer les positions relatives entre les individus, on s'intéresse aux directions exprimées par les corrélations des variables avec les axes. Par exemple, la Renault 30 et la Toyota Corolla s'opposent sur le premier axe (Figure 13) à cause du différentiel de cylindrée (Figure 11).

Biplot. Pour aller plus loin, certains outils proposent un graphique « biplot », assez séduisant au demeurant, pour représenter simultanément les individus et les variables. Il ne doit pas nous induire en erreur. Pour les individus, nous raisonnons en positions relatives dans le repère ; pour les variables, nous raisonnons en directions. Mais les proximités entre individus et variables n'ont aucun sens. Une fois cette précaution assimilée, nous représentons ci-dessous le graphique « biplot » produit par la fonction princomp() du logiciel R (Figure 15).

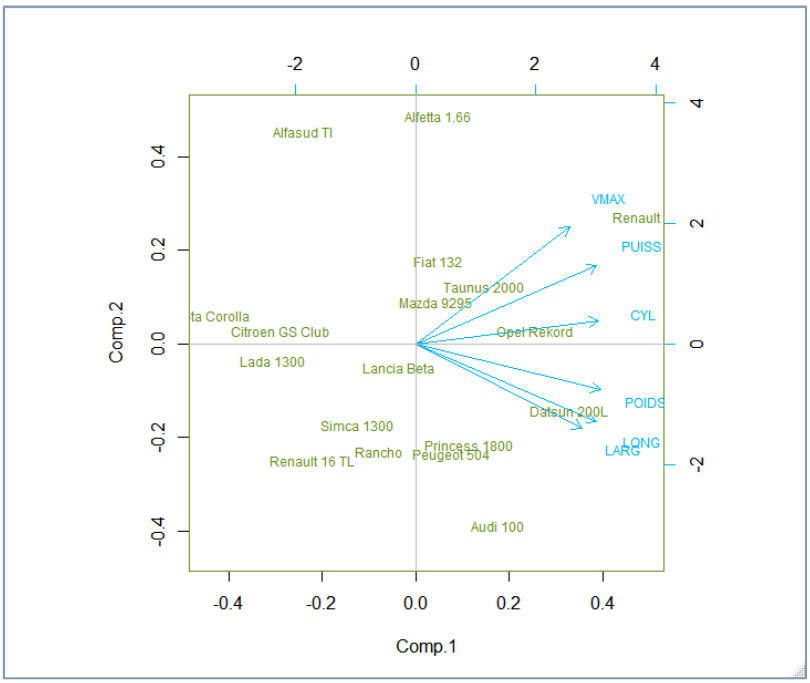


Figure 15 – Représentation "biplot" – Données "Autos"

Annoncer par exemple à la lecture de ce graphique que la Renault 30 serait puissante (proche de PUISS) mais légère (éloigné de POIDS) est erroné. Dire en revanche que le 1er facteur est défini positivement par l'ensemble des variables, et que la Renault 30, située à son extrémité-est, réunit les caractéristiques de (POIDS, VMAX, etc.) élevées, est correct.

1.4 Reconstitution des données

L'ACP préserve au mieux les distances entre individus et les corrélations entre les variables dans un espace de représentation de dimension réduite. Elle permet également de reconstituer – de façon approximative – les valeurs initiales des données en se focalisant sur les formes essentielles captées sur les composantes. Ce faisant, elle évacue le bruit résiduel contenu dans les données. Cette fonctionnalité est souvent mise en avant dans le traitement d'images, lors des opérations de compression avec perte, que l'on essaie de contrôler avec la détermination de la taille de la dimensionnalité à retenir. Elle est moins mise en avant sur les données tabulaires usuelles. Pourtant, je lui vois au moins une propriété très intéressante, un peu à la manière des auto-encodeurs (« Auto-encodeur avec Keras sous Python », novembre 2019), elle contribue à l'identification des observations atypiques par comparaison entre les valeurs observées et « estimées », celles dont les caractéristiques diffèrent de la tendance générale.

Revenons sur quelques formules importantes pour décrypter le mécanisme. Les coordonnées factorielles des individus est obtenu par le produit matriciel :

$$F_{(p)} = Z\ a_{(p)}$$

Où (Z) est la matrice des données centrées et réduites, de dimension (n, p) ; $F_{(p)}$ les coordonnées factorielles sur l'ensemble des (p) facteurs (n, p) ; a(p) est la matrice des vecteurs propres (p,p), elle est orthogonale, son inverse est égale à sa transposée.

Il est très facile de calculer les coordonnées dans l'espace originel à partir des coordonnées factorielles, si l'on prend tous les (p) facteurs :

$$Z = F_{(p)}\ a_{(p)}^T$$

Nous avons une approximation si l'on ne prend que les (q < p) premiers facteurs :

$$\hat{Z} = F_{(q)}\ a_{(q)}^T$$

Sachant que les variables ont été centrées et réduits avec la formule :

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{\sigma_j}$$

Les valeurs estimées « déstandardisées » s'écrivent :

$$\hat{x}_{ij} = \hat{z}_{ij} \times \sigma_j + \bar{x}_j$$

Nous souhaitons reconstituer les données initiales à partir des (q = 2) premiers facteurs pour les données « Autos ». L'objet PCA nous fournit tous les éléments de calculs. Nous affichons pour contrôle les vecteurs propres (a_{jk}). Si elles ne sont pas directement disponibles, n'oublions pas que nous pouvons les obtenir à partir des corrélations des variables aux facteurs [$a_{jk} = \frac{r_j(F_k)}{\sqrt{\lambda_k}}$]. Puis nous calculons l'approximation des valeurs centrées et réduites.

```
#vecteurs propres (q = 2) premiers facteurs
print(acp.eigen_vectors_[:,:2])
```

```
[[ 0.42493602 -0.12419108]
 [ 0.42179441 -0.41577389]
 [ 0.42145993  0.41181773]
 [ 0.38692224  0.446087  ]
 [ 0.43051198  0.24267581]
 [ 0.35894427 -0.6198626 ]]
```

```
#valeurs centrées et réduites reconstituées
ZC = numpy.dot(acp.row_coord_[:,:2],numpy.transpose(acp.eigen_vectors_[:,:2]))
print(ZC)
```

```
[[-0.68714004 -0.15974647 -1.6368457  -1.6241662  -1.35417384  0.3391225 ]
 [ 0.47387522  0.02371103  1.28695448  1.28535587  1.04280237 -0.38607854]
 [-0.55943466 -0.75259209 -0.19400286 -0.13222707 -0.31822269 -0.81989747]
 [-1.07965636 -1.03865558 -1.1312167  -1.04619406 -1.1354209  -0.85385718]
 [ 0.26819434  0.46966566 -0.10612313 -0.14473684  0.01539943  0.58473234]
 [-0.15364146 -0.20987928 -0.04744643 -0.03021689 -0.08337739 -0.23078966]
 [ 0.17474853 -0.09946343  0.67249777  0.68085164  0.52086972 -0.33287478]
 [-0.94974758 -1.22950797 -0.41744566 -0.31654998 -0.60091837 -1.30714333]
 [ 2.00594882  2.30223229  1.42050343  1.23175145  1.64032553  2.2421554 ]
 [-1.66436358 -1.58295814 -1.77713557 -1.64757164 -1.77325693 -1.28423718]
 [ 0.42348573  0.97974779 -0.60312467 -0.68377858 -0.27568782  1.34254999]
 [ 0.32812626  0.07949877  0.77575209  0.76943152  0.64259987 -0.15627763]
 [ 1.18032648  1.00804104  1.46982552  1.3874099   1.40186847  0.70907252]
 [ 0.61916179  0.75689253  0.35381082  0.29172512  0.44800453  0.77354586]
 [-0.40516711 -0.66475603  0.07842183  0.13305548 -0.07967646 -0.80453439]
 [ 0.20813654  0.31078209  0.01587771 -0.00964997  0.07961491  0.35923348]
 [ 0.98596369  1.00919522  0.92207459  0.83941547  0.96045074  0.88657854]
 [-1.16881662 -1.20220742 -1.08237754 -0.98390521 -1.13120118 -1.06130051]]
```

A partir des moyennes et écarts-type stockées par l'objet lors de la réalisation de l'ACP, nous formons les valeurs reconstituées, puis nous affichons les valeurs initiales pour comparaisons.

```
#moyennes
print(acp.means_)
```

```
[[1631.66666667   84.61111111  433.5         166.66666667 1078.83333333
   158.27777778]]
```

```
#écarts-type
print(acp.std_)
```

```
[[363.39449027  19.80218531  21.48449053   5.16397779 133.09906503
   11.79833112]]
```

```
#déstandardisation et décentrage
XC = ZC.copy()

#boucle sur les variables
for j in range(p):
    XC[:,j] = ZC[:,j] * acp.std_[0][j] + acp.means_[0][j]

#print des valeurs approchées
print(pandas.DataFrame(XC,columns=D.columns,index=D.index).round(1))
```

Modele	CYL	PUISS	LONG	LARG	POIDS	VMAX
Alfasud TI	1382.0	81.4	398.3	158.3	898.6	162.3
Audi 100	1803.9	85.1	461.1	173.3	1217.6	153.7
Simca 1300	1428.4	69.7	429.3	166.0	1036.5	148.6
Citroen GS Club	1239.3	64.0	409.2	161.3	927.7	148.2
Fiat 132	1729.1	93.9	431.2	165.9	1080.9	165.2
Lancia Beta	1575.8	80.5	432.5	166.5	1067.7	155.6
Peugeot 504	1695.2	82.6	447.9	170.2	1148.2	154.4
Renault 16 TL	1286.5	60.3	424.5	165.0	998.9	142.9
Renault 30	2360.6	130.2	464.0	173.0	1297.2	184.7
Toyota Corolla	1026.8	53.3	395.3	158.2	842.8	143.1
Alfetta 1.66	1785.6	104.0	420.5	163.1	1042.1	174.1
Princess 1800	1750.9	86.2	450.2	170.6	1164.4	156.4
Datsun 200L	2060.6	104.6	465.1	173.8	1265.4	166.6
Taunus 2000	1856.7	99.6	441.1	168.2	1138.5	167.4
Rancho	1484.4	71.4	435.2	167.4	1068.2	148.8
Mazda 9295	1707.3	90.8	433.8	166.6	1089.4	162.5
Opel Rekord	1990.0	104.6	453.3	171.0	1206.7	168.7
Lada 1300	1206.9	60.8	410.2	161.6	928.3	145.8

```
#print des valeurs initiales pour comparaisons
print(D)
```

Modele	CYL	PUISS	LONG	LARG	POIDS	VMAX
Alfasud TI	1350	79	393	161	870	165
Audi 100	1588	85	468	177	1110	160
Simca 1300	1294	68	424	168	1050	152
Citroen GS Club	1222	59	412	161	930	151
Fiat 132	1585	98	439	164	1105	165
Lancia Beta	1297	82	429	169	1080	160
Peugeot 504	1796	79	449	169	1160	154
Renault 16 TL	1565	55	424	163	1010	140
Renault 30	2664	128	452	173	1320	180
Toyota Corolla	1166	55	399	157	815	140
Alfetta 1.66	1570	109	428	162	1060	175

```
Princess 1800    1798     82   445   172   1160   158
Datsun 200L      1998    115   469   169   1370   160
Taunus 2000      1993     98   438   170   1080   167
Rancho           1442     80   431   166   1129   144
Mazda 9295       1769     83   440   165   1095   165
Opel Rekord      1979    100   459   173   1120   173
Lada 1300        1294     68   404   161    955   140
```

Attardons-nous sur la Lancia Beta qui était mal représentée dans le premier plan factoriel avec un COS2 de 11.54% (section 1.3.3). On sait pourquoi maintenant. Par rapport à ses caractéristiques (vitesse, puissance, dimensions) qui la place plutôt parmi les berlines familiales de l'époque, elle devrait présenter une cylindrée plus élevée (1575.8 cm^3 estimée vs. 1297 cm^3 observée).

1.5 Projection des individus supplémentaires

1.5.1 Principe et fonctions de projection

La projection des individus supplémentaires (ou illustratifs selon la finalité poursuivie) est une fonctionnalité importante de l'analyse en composantes principales. L'objectif est de positionner de nouveau individus par rapport à ceux (les individus actifs, l'échantillon d'apprentissage, l'échantillon de référence) qui ont contribué à la construction du repère factoriel.

On peut voir plusieurs motivations à cette démarche :

- Situer les positions des individus qui ont été collectés après coup.
- Identifier parmi ces individus ceux dont les caractéristiques diffèrent significativement de l'échantillon de référence. On parle de détection de nouveautés (novelty detection, « Détection des anomalies sous Python », mars 2020).
- Positionner des individus appartenant à une population différente ou spécifique. Par exemple, dans les années 70, voir comment se placent les véhicules du bloc de l'est aurait pu être intéressant. Il y a déjà une Lada dans les individus actifs, mais on pourrait analyser les caractéristiques des Skoda, Tatra, Dacia et autres Trabant (de l'époque) en référence à ceux du fichier.
- Intégrer après coup les observations jugées trop atypiques ou influentes, qui étaient susceptibles de fausser la construction du repère factoriel. On aurait pu écarter la

Renault 30 des calculs par exemple, du fait de son influence très marquée sur les 2 premiers facteurs, puis l'ajouter comme individu supplémentaire par la suite.

Les coefficients des vecteurs propres (a_{jk}) font office de paramètres dans la fonction de projection permettant de calculer les coordonnées d'un individu supplémentaire (i*) sur le facteur (F_k) :

$$F_{i^*k} = a_{1k}z_{i^*1} + \cdots + a_{jk}z_{i^*j} + \cdots + a_{pk}z_{i^*p}$$

Où (z_{i^*j}) est la valeur centrée et réduite de (x_{i^*j}) avec les paramètres (moyenne, écart-type) calculés sur l'échantillon d'apprentissage.

$$z_{i^*j} = \frac{x_{i^*j} - \bar{x}_j}{\sigma_j}$$

Certains logiciels comme TANAGRA fournit ainsi les coefficients standardisés (a_{jk}) et les paramètres de centrage et réduction pour qu'il soit possible de réaliser le déploiement à l'aide d'autres outils (je demande aux étudiants de le faire sous Excel pour qu'ils décryptent bien le mécanisme).

Factor Score Coefficients

Attribute	Mean	Std-dev	Axis_1	Axis_2
CYL	1631.66667	363.39449	0.42494	-0.12419
PUISS	84.61111	19.80219	0.42179	-0.41577
LONG	433.50000	21.48449	0.42146	0.41182
LARG	166.66667	5.16398	0.38692	0.44609
POIDS	1078.83333	133.09907	0.43051	0.24268
VMAX	158.27778	11.79833	0.35894	-0.61986

Figure 16 – Coefficients et paramètres pour le déploiement sous TANAGRA – ACP – Données "Autos"

On peut même imaginer produire une fonction de projection avec des coefficients bruts (qui s'appliquent directement sur les variables originelles, sans transformations préalables) pour faciliter le déploiement.

$$F_{i^*k} = b_{0k} + b_{1k}x_{i^*1} + \cdots + b_{jk}x_{i^*j} + \cdots + b_{pk}x_{i^*p}$$

Où

- $b_{jk} = \frac{a_{jk}}{\sigma_j}$ pour (j ≥ 1) ;
- $b_{0k} = -\sum_{j=1}^{p} \frac{a_{jk} \times \bar{x}_j}{\sigma_j}$

1.5.2 Exemple sur les données « Autos »

Nous souhaitons positionner deux véhicules supplémentaires dans le plan factoriel : une « Peugeot 304 S » et une « Peugeot 604 ». Voici leurs caractéristiques :

```
#chargement des obs. supplémentaires
IndSup = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="DATA_ACP_IND_SUP",index_col=0)
print(IndSup)
```

```
               CYL  PUISS  LONG  LARG  POIDS  VMAX
Modele
Peugeot 604   2664    136   472   177   1410   180
Peugeot 304 S 1288     74   414   157    915   160
```

On connaît bien ces véhicules (304 S, petite sportive ; 604, routière statutaire), on imagine très bien leurs voisinages respectifs. Voyons si l'ACP saura les positionner correctement.

La fonction « .transform() » de l'objet PCA fait merveille ici, nous pouvons lui envoyer la description brute des individus supplémentaires, sans préparation préalable. Nous obtenons les coordonnées factorielles (2 observations, p = 6 facteurs).

```
#coordonnées factorielles des individus supplémentaires
coordSup = acp.transform(IndSup.values)
print(coordSup)
```

```
[[ 5.56329226 -0.33860928  0.46428878 -0.40214608  0.38981076  0.08102064]
 [-2.21224139 -1.25777905  0.09304388  0.35370189 -0.648528   -0.12473042]]
```

Nous les plaçons dans le plan factoriel pour localiser leurs voisinages parmi les observations actives.

```
#position des points supplémentaires dans le plan
fig, ax = plt.subplots(figsize=(10,10))
ax.plot(acp.row_coord_[:,0],acp.row_coord_[:,1],'wo')
ax.axis([-6.5,+6.5,-6.5,+6.5])
ax.plot([-6.5,+6.5],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-6.5,+6.5],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Carte des individus")

#points actifs
for i in range(n):
    ax.text(acp.row_coord_[i,0],acp.row_coord_[i,1],D.index[i],color="dimgray")
```

```
#points illutratifs
for i in range(IndSup.shape[0]):
    ax.text(coordSup[i,0],coordSup[i,1],IndSup.index[i],color="blue")

plt.show()
```

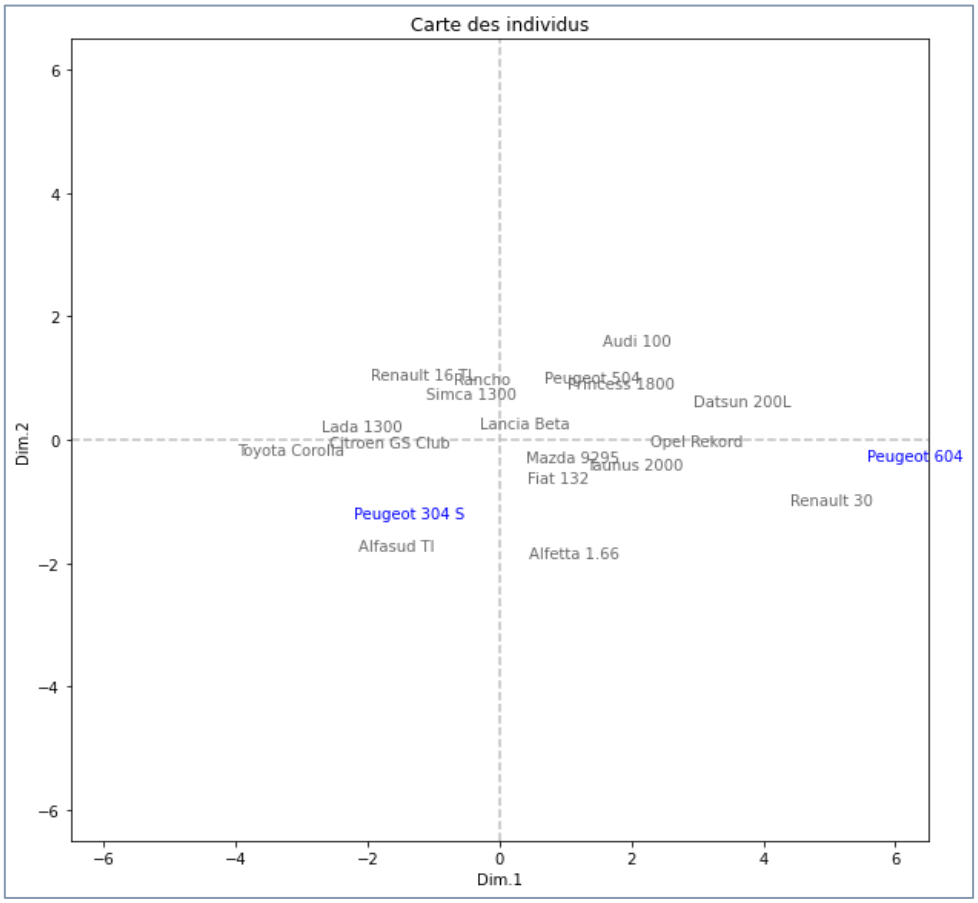


Figure 17 – Position des observations supplémentaires – Données "Autos"

Oui, la « 304 S » est une petite sportive, a l'instar de l'Alfasud TI (première version). Quant à la 604, elle était bien une concurrente directe de la Renault 30. Ces résultats confortent la légitimité des résultats de l'ACP menée sur les données « Autos ».

1.5.3 Ellipse de Hotelling

Pour identifier les « nouveautés » c.-à-d. les individus supplémentaires dont les caractéristiques diffèrent sensiblement de l'échantillon de référence, nous pouvons utiliser l'ellipse de Hotelling (Tenenhaus, 2007, pages 178 à 180).

Le T^2 de Hotelling d'une observation n°i dans un repère de dimension (q) s'écrit :

$$T_i^2 = \sum_{k=1}^{q} \frac{F_{ik}^2}{\lambda_k}$$

On montre que la statistique...

$$\frac{n(n-q)}{q(n^2-1)} T_i^2$$

... suit une loi de Fisher à (q, n-q) degrés de liberté. On peut dès lors définir une zone de confiance au niveau 95%. Une observation supplémentaire est considérée atypique lorsque

$$T_i^2 \geq \frac{q(n^2-1)}{n(n-q)} F_{0.95}(q, n-q)$$

Où $F_{0.95}$ est le quantile de la loi de Fisher.

Dans le plan (q = 2), nous avons une ellipse d'équation :

$$\frac{F_{i1}^2}{\lambda_1} + \frac{F_{i2}^2}{\lambda_2} = \frac{q(n^2-1)}{n(n-q)} F_{0.95}(q, n-q)$$

Pour en dessiner les contours autour de l'origine, nous devons en déterminer le diamètre sur l'axe des abscisses (lorsque $F_2 = 0$) et des ordonnées (lorsque $F_1 = 0$).

Si nous posons $c = \frac{q(n^2-1)}{n(n-q)} F_{0.95} = \frac{2(18^2-1)}{18(18-2)} 3.63 = 8.15$. Les étendues en abscisse et ordonnées sont égales à :

$$\begin{cases} e_1 = 2\sqrt{\lambda_1 \times c} = 12.0 \\ e_2 = 2\sqrt{\lambda_2 \times c} = 5.28 \end{cases}$$

Sous Python, nous calculons les diamètres en abscisse et ordonnée pour les données « Autos ».

```
#nombre de facteurs à prendre en compte
q = 2

#quantile de la loi de Fisher à 95%
pf = stats.f.ppf(0.95,q,n-q)
print(pf)
```

```
3.63372346759163

#seuil
c = (q*(n**2-1))/(n*(n-q))*pf
print(c)

8.150643611334003

#étendue en abscisse
e1 = 2*math.sqrt(acp.eig_[0][0]*c)
print(e1)

12.005471835819806

#étendue en ordonnée
e2 = 2*math.sqrt(acp.eig_[0][1]*c)
print(e2)

5.282975914670268
```

Et nous ajoutons l'ellipse dans le graphique des individus actifs et illustratifs.

```
#classe pour ellipse
from matplotlib.patches import Ellipse

#position des points supplémentaires dans le plan
#rajouter l'ellipse de Hotelling
fig, ax = plt.subplots(figsize=(10,10))
ax.plot(acp.row_coord_[:,0],acp.row_coord_[:,1],'wo')
ax.axis([-6.5,+6.5,-6.5,+6.5])
ax.plot([-6.5,+6.5],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-6.5,+6.5],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Ellipse de Hotelling")

#points actifs
for i in range(n):
    ax.text(acp.row_coord_[i,0],acp.row_coord_[i,1],D.index[i],color="dimgray")

#points illutratifs
for i in range(IndSup.shape[0]):
    ax.text(coordSup[i,0],coordSup[i,1],IndSup.index[i],color="blue")

#ellispe de Hotelling - centrée autour de l'origine
ellipse = Ellipse((0,0),width=e1,height=e2,facecolor='none',edgecolor='tomato',linestyle='--')
ax.add_patch(ellipse)

plt.show()
```

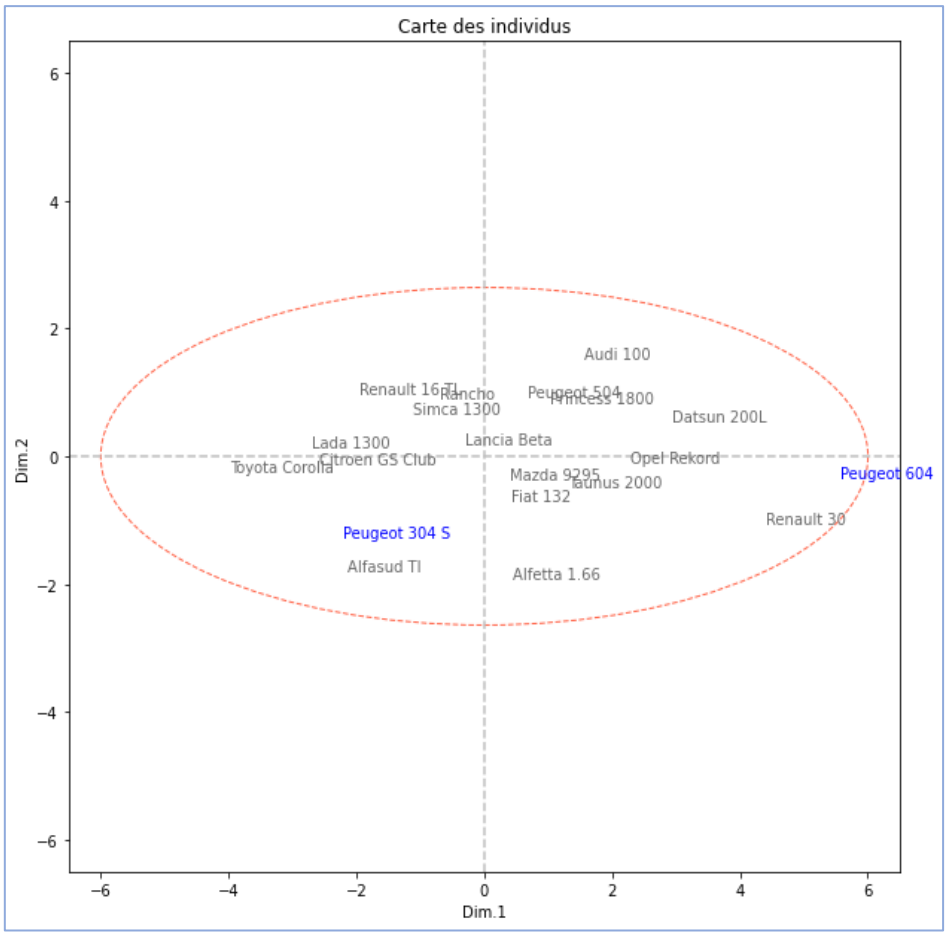


Figure 18 – Ellipse de confiance – Données "Autos"

La Peugeot 604 est en dehors de l'ellipse de confiance pour le niveau de 95% que l'on s'est choisi. Ce résultat n'est pas étonnant lorsqu'on revient sur les données. Elle possède les valeurs plus élevées de cylindrée (à égalité avec la Renault 30), puissance, longueur, largeur (idem Audi 100), poids et vitesse maximum (idem Renault 30). Elle est « hors norme » en ce sens.

1.6 Traitement des variables illustratives

Les variables illustratives ont pour vocation de renforcer l'interprétation des composantes. Elles ne sont pas utilisées pour leur construction, elles interviennent seulement après coup pour mieux comprendre et commenter les résultats. Pour l'exemple des données « Autos », nous avons utilisé en variables actives les caractéristiques intrinsèques des véhicules (largeur, poids, puissance, etc.) pour élaborer les facteurs. Nous nous appuyons ensuite sur des variables illustratives intégrant des considérations subjectives (prix, finition) ou calculées ex-post

(rapport poids-puissance) pour entériner leurs positions relatives dans la représentation factorielle.

Pour les données « Autos », nous chargeons la 3ème feuille de calcul composée des variables illustratives pour les mêmes (n = 18) véhicules.

```
#variables illustratives
varIllus = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="DATA_ACP_VAR_ILLUS",index_col=0)
print(varIllus)
```

```
               FINITION   PRIX  RPOIDPUIS
Modele
Alfasud TI           2_B  30570  11.012658
Audi 100            3_TB  39990  13.058824
Simca 1300           1_M  29600  15.441176
Citroen GS Club      1_M  28250  15.762712
Fiat 132             2_B  34900  11.275510
Lancia Beta         3_TB  35480  13.170732
Peugeot 504          2_B  32300  14.683544
Renault 16 TL        2_B  32000  18.363636
Renault 30          3_TB  47700  10.312500
Toyota Corolla       1_M  26540  14.818182
Alfetta-1.66        3_TB  42395   9.724771
Princess-1800        2_B  33990  14.146341
Datsun-200L         3_TB  43980  11.913043
Taunus-2000          2_B  35010  11.020408
Rancho              3_TB  39450  14.112500
Mazda-9295           1_M  27900  13.192771
Opel-Rekord          2_B  32700  11.200000
Lada-1300            1_M  22100  14.044118
```

L'idée est de conforter les résultats de l'ACP avec ces nouvelles variables. Les indicateurs utilisés ne sont pas les mêmes selon qu'elles sont quantitatives ou qualitatives.

1.6.1 Variable illustrative quantitative

Corrélation. La corrélation avec les facteurs est l'indicateur naturel pour positionner une variable supplémentaire quantitative dans le repère factoriel. Si (y) est la variable à analyser, le coefficient s'écrit :

$$r_y(F_k) = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})(F_{ik} - \bar{F}_k)}{\sigma_y \times \sigma_{F_k}}$$

$$= \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})\, F_{ik}}{\sigma_y \times \sqrt{\lambda_k}}$$

En pratique, il s'agit de faire simplement appel aux librairies de calcul du langage utilisé. Sous Python, nous utilisons corrcoef() de « Numpy ». Pour « prix » et « rapport poids-puissance » sur les deux premiers facteurs, nous avons :

```
#corrélation de PRIX avec les 2 premiers facteurs
cPrix = numpy.corrcoef(varIllus.PRIX,acp.row_coord_[:,:2],rowvar=False)[0,1:]
print(cPrix)

[ 0.77247524 -0.08670844]

#corrélation de RPOIDPUIS avec les 2 premiers facteurs
cRPP = numpy.corrcoef(varIllus.RPOIDPUIS,acp.row_coord_[:,:2],rowvar=False)[0,1:]
print(cRPP)

[-0.58903888  0.67254512]
```

Ainsi :

- Le prix est fortement lié positivement au premier facteur, lequel est déterminé par la cylindrée et l'encombrement des véhicules. Ce n'est pas vraiment étonnant.
- Le rapport poids puissance est un indicateur de performance. Plus il est faible, plus sportif est le véhicule. Il est quasi-également lié aux deux facteurs : négativement au premier, c'est tout à fait naturel puisque cette composante traduit la cylindrée et la puissance ; positivement au second, en opposition à la vitesse maximum, ce que se conçoit très bien.

Insérer ces variables dans le cercle des corrélations donne une vision synthétique toujours bien sympathique.

```
#placement des variables illustratives dans le cercle des corrélations
fig, ax = plt.subplots(figsize=(5,5))
ax.axis([-1,+1,-1,+1])
ax.plot([-1,+1],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-1,+1],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Cercle des corrélations")

#variables actives
for i in range(p):
    ax.text(acp.col_coord_[i,0],acp.col_coord_[i,1],D.columns[i],color="dimgray")

#variables illustratives
ax.text(cPrix[0],cPrix[1],'PRIX',color='brown')
ax.arrow(0,0,cPrix[0]-0.1,cPrix[1]+0.04,color='r',head_width=0.05)

ax.text(cRPP[0],cRPP[1],'RPOIDPUIS',color="red")
ax.arrow(0,0,cRPP[0]+0.2,cRPP[1]-0.1,color='r',head_width=0.05)

#cercle pour faire joli
```

```
ellipse = Ellipse((0,0),width=2,height=2,facecolor='none',edgecolor='silver',linestyle='--')
ax.add_patch(ellipse)

plt.show()
```

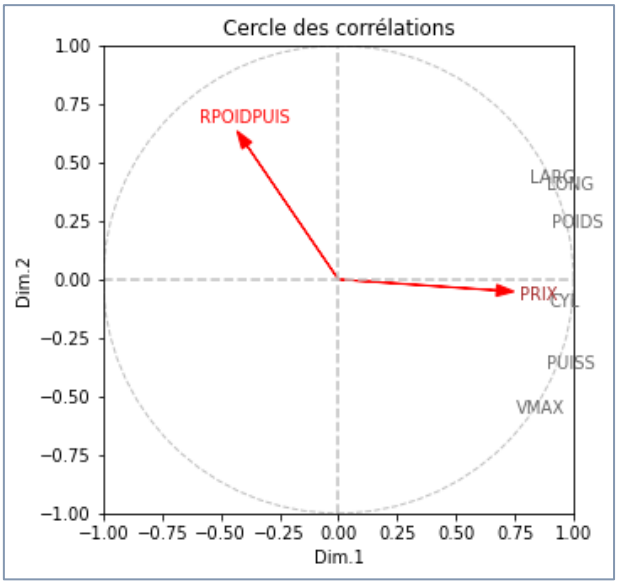


Figure 19 – Cercle des corrélations avec variables illustratives – ACP – Données "Autos"

Qualité de représentation – COS2. Pour consolider les résultats, on peut éventuellement calculer les COS2 pour indiquer la qualité de la représentation d'une variable c.-à-d. le carré de la corrélation. L'intérêt est de disposer d'un indicateur non-signé qui s'additionne d'un facteur à l'autre. Nous n'avons pas la garantie en revanche que la somme des COS2 sur l'ensemble des facteurs soit égale à 1 comme nous pouvons le constater pour la variable PRIX.

```
#corrélation de prix avec l'ensemble des (p = 6) facteurs
cPrixBis = numpy.corrcoef(varIllus.PRIX,acp.row_coord_[:,:],rowvar=False)[0,1:]
print(cPrixBis)

[ 0.77247524 -0.08670844  0.13389277  0.22582891  0.15944978  0.10254878]

#au carré : COS2
print(cPrixBis**2)

[0.596718   0.00751835 0.01792727 0.0509987  0.02542423 0.01051625]

#somme n'est pas égale à 1 sur l'ensemble des facteurs, et c'est normal
print(numpy.sum(cPrixBis**2))

0.7091028079636765
```

Remarque : Au cas où la question viendrait à l'esprit, calculer la contribution n'a aucun sens pour les variables illustratives puisqu'elles n'ont pas participé à l'élaboration du repère factoriel.

1.6.2 Variable illustrative qualitative

Le traitement des variables illustratives qualitatives s'apparente à une analyse de variance à 1 facteur (Rakotomalala R., « Comparaisons de populations – Tests paramétriques », version 1.2, mai 2010 ; section 1.3) : l'objectif est d'apprécier dans quelle mesure les moyennes conditionnelles définies par les groupes associés aux modalités sont suffisamment écartées sur le facteur étudié. Nous disposons d'un indicateur global, le carré du rapport de corrélation ; et d'un indicateur par groupe, la valeur test (« Interpréter la valeur-test », avril 2008).

Moyennes conditionnelles – Coordonnées factorielles. Les moyennes conditionnelles ($\bar{y}_{gk}$) constituent le point de départ de l'analyse, où « g » désigne une des modalités de la variable supplémentaire (y, à G catégories), « k » le facteur sur lequel nous travaillons.

$$\bar{y}_{gk} = \frac{1}{n_g} \sum_{i:y_i=g} F_{ik}$$

Où (n_g) est l'effectif du groupe « g ».

Concrètement, pour la variable illustrative FINITION sur le premier facteur, nous calculons les effectifs par groupe et les moyennes conditionnelles.

```
#data frame temporaire
df = varIllus.copy()
df['F1'] = acp.row_coord_[:,0]
df['F2'] = acp.row_coord_[:,1]
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18 entries, Alfasud TI to Lada-1300
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   FINITION   18 non-null     object
 1   PRIX       18 non-null     int64
 2   RPOIDPUIS  18 non-null     float64
 3   F1         18 non-null     float64
 4   F2         18 non-null     float64
dtypes: float64(3), int64(1), object(1)
memory usage: 864.0+ bytes
None
```

```
#effectif par groupe de FINITION
n_g = pandas.pivot_table(df,values='F1',index='FINITION',aggfunc='count')
print(n_g)
```

```
          F1
FINITION
1_M        5
2_B        7
3_TB       6

#moyennes conditionnelles pour le 1er facteur
m_1 = pandas.pivot_table(df,values='F1',index='FINITION',aggfunc='mean')
print(m_1)

                F1
FINITION
1_M      -2.000355
2_B       0.235313
3_TB      1.392430

#moyennes conditionnelles pour le 2nd facteur
m_2 = pandas.pivot_table(df,values='F2',index='FINITION',aggfunc='mean')
print(m_2)

                F2
FINITION
1_M       0.022579
2_B      -0.045271
3_TB      0.034001
```

Ces moyennes conditionnelles correspondent également aux coordonnées factorielles. Plaçons-les dans notre repère pour disposer d'une vision d'ensemble.

```
#représentation graphique avec les positions
#des points supplémentaires dans le plan
fig, ax = plt.subplots(figsize=(10,10))
ax.plot(acp.row_coord_[:,0],acp.row_coord_[:,1],'wo')
ax.axis([-5,+5,-5,+5])
ax.plot([-5,+5],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-5,+5],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Carte des individus - Modalités de FINITION")

#points actifs
for i in range(n):
    ax.text(acp.row_coord_[i,0],acp.row_coord_[i,1],D.index[i],color="gray")

#position des modalités de la variable illustrative
for g in range(len(m_1.index)):
    ax.text(m_1.F1[g],m_2.F2[g],m_1.index[g],color="red",weight="bold",fontsize=16)

plt.show()
```

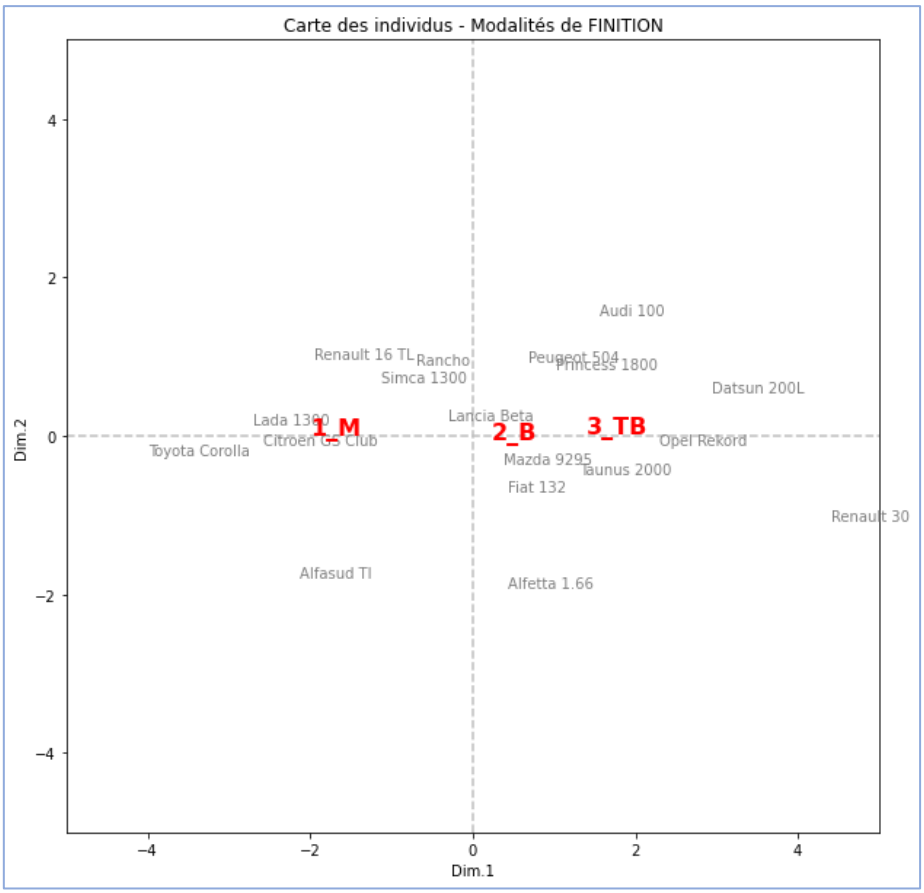


Figure 20 – Position des modalités de la variable illustrative "FINITION" – ACP – Données "Autos"

Remarque : Nous aurions pu également colorier les étiquettes selon le groupe d'appartenance défini par « FINITION ». Mais le positionnement des moyennes conditionnelles est plus en phase avec les calculs ultérieurs.

Manifestement, les niveaux de FINITION se distingue fortement sur le 1er facteur, quasiment pas sur le 2nd. Elle (la finition) est intimement liée à la notion de « gamme » de véhicules mis en avant lors de la lecture des facteurs. Plus le véhicule monte en gamme, plus élevé est le niveau de finition.

Hélas, les résultats sont rarement aussi évidents en pratique. Il nous faut disposer d'un indicateur numérique, qui possède l'avantage de pouvoir s'appliquer à un nombre éventuellement important de variables illustratives qualitatives.

Rapport de corrélation. Le rapport de corrélation témoigne de la dispersion relative des moyennes conditionnelles autour de la moyenne globale. Sachant que les facteurs sont centrés

et que leur variance est égale à la valeur propre, l'expression du carré du rapport de corrélation pour le facteur F_k est simplifiée :

$$\eta^2_{F_k/y} = \frac{\sum_{g=1}^{G} \frac{n_g}{n} \bar{y}_{gk}}{\lambda_k}$$

Nous les produisons très facilement à partir des effectifs et moyennes conditionnelles calculés précédemment.

```
#(carré du) rapport de corrélation sur le 1er facteur
print(numpy.sum((n_g.F1/n)*(m_1.F1**2))/acp.eig_[0][0])

0.40248443698930525

#rapport de corrélation sur le 2nd facteur
print(numpy.sum((n_g.F1/n)*(m_2.F2**2))/acp.eig_[0][1])

0.0015465958518609553
```

Les indicateurs numériques confortent les impressions visuelles, c'est toujours rassurant : la discrimination des groupes de FINTION est surtout tangible sur le 1er facteur ($\eta^2_{F_1/y} = 0.402$ vs. $\eta^2_{F_2/y} = 0.0015$).

Valeur-test. L'idée des « valeurs-test » (VT) s'apparente à un test post-hoc où l'on essaie de distinguer les groupes qui s'écartent le plus de la moyenne globale c.-à-d. de l'origine puisque les facteurs sont centrés. Dans le graphique ci-dessus, les groupes de FINITION se distinguent surtout sur le 1er facteur. Soit. Mais certains sont plus écartés de l'origine que d'autres. Les réponses sont évidentes visuellement, la valeur-test se charge de les matérialiser.

La formule est de nouveau simplifiée dans le cadre de l'ACP. La valeur-test pour la modalité « g » de la variable illustrative « y » sur le facteur F_k s'écrit :

$$VT_{gk} = \frac{\bar{y}_{gk}}{\sqrt{\frac{n - n_g}{n - 1} \times \frac{\lambda_k}{n_g}}}$$

Réalisons les calculs pour le 1er facteur.

```
#valeurs test sur le 1er facteur
vt_1 = (m_1.F1)/numpy.sqrt((n-n_g.F1)/(n-1)*(acp.eig_[0][0]/n_g.F1))
print(vt_1)

FINITION
1_M    -2.432717
2_B     0.368103
3_TB    1.930766
Name: F1, dtype: float64
```

Le groupe de finition moyenne (« 1_M ») est la plus éloigné de l'origine, etc.

Là aussi, la conclusion numérique concorde avec la perception visuelle (Figure 20). L'intérêt est que nous pouvons associer l'indicateur à un test de significativité. En se référant à la normalité asymptotique de la statistique VT, pour un test bilatéral à 5%, on peut considérer que l'écartement est significatif dès lors que ($|VT| \geq 2$), … à peu près. Très à peu près même, parce que d'une part les échantillons ne sont ni indépendants, ni appariés, le groupe est inclus dans l'échantillon global ; d'autre part, tout écart paraît significatif dès que les effectifs augmentent.

En pratique, il faut plutôt voir la valeur-test comme un indicateur qui permet de hiérarchiser rapidement les écarts d'un facteur à l'autre (le poids des facteurs sous forme de variance restituée intervient dans les calculs), attirant notre attention sur les résultats qui seraient de prime abord les plus intéressants.

Cosinus carré – COS2. Le calcul de la qualité de représentation des modalités d'une variable supplémentaire, le COS2, est analogue à celui des individus (section 1.3.3). Nous devons tout d'abord produire la distance globale à l'origine de la modalité dans le repère initial, sur les données (Z), elle nous servira de référence. Ensuite, pour chaque facteur, nous opposons la distance à l'origine locale avec la globale.

Pour une variable (z_j), la moyenne conditionnelle pour la modalité « g » est :

$$\bar{z}_{gj} = \frac{1}{n_g} \sum_{i:y_i=g} z_{ij}$$

La distance à l'origine globale du groupe « g » dans l'espace des variables (z) centrées et éventuellement réduites :

$$d_g = \sqrt{\sum_{j=1}^{p} \bar{z}_{gj}^2}$$

Pour les données « Autos », réalisons les calculs pour la variable supplémentaire FINITION.

Nous créons tout d'abord un data frame temporaire pour accueillir les variables centrées et réduites Z, nous lui accolons la variable supplémentaire FINITION.

```
#transformer Z en data frame
dfZ = pandas.DataFrame(Z,columns=D.columns,index=varIllus.index)

#rajouter la colonne FINITION
dfZ['FINITION'] = varIllus['FINITION']

#vérification
print(dfZ.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18 entries, Alfasud TI to Lada-1300
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   CYL       18 non-null     float64
 1   PUISS     18 non-null     float64
 2   LONG      18 non-null     float64
 3   LARG      18 non-null     float64
 4   POIDS     18 non-null     float64
 5   VMAX      18 non-null     float64
 6   FINITION  18 non-null     object
dtypes: float64(6), object(1)
memory usage: 1.8+ KB
```

Nous calculons ensuite les moyennes des (z_j) conditionnellement à FINITION.

```
#moyennes conditionnelles sur les variables Z
mz_g = pandas.pivot_table(dfZ,index='FINITION',aggfunc='mean')
print(mz_g)
```

```
               CYL      LARG      LONG     POIDS     PUISS      VMAX
FINITION
1_M      -0.777851 -0.826236 -0.823850 -0.825200 -0.909552 -0.735509
2_B       0.253299  0.147542  0.083116 -0.050267 -0.009218  0.170188
3_TB      0.352693  0.516398  0.589573  0.746311  0.768714  0.414371
```

Nous pouvons former les distances à l'origine.

```
#distance à l'origine
disto2_g = numpy.apply_along_axis(arr=mz_g.values,axis=1,func1d=lambda x:numpy.sum(x**2))
print(numpy.sqrt(disto2_g))
```

```
[2.00391116 0.35272284 1.43466388]
```

Pour obtenir le COS2 du groupe « g » sur le facteur (F_k), nous opposons le carré de la distance locale (le carré de la moyenne conditionnelle sur le facteur) avec le carré de la distance globale :

$$COS^2_{gk} = \frac{\bar{y}^2_{gk}}{d^2_g}$$

Pour le 1er facteur de l'ACP sur les données « Autos » :

```
#COS2 sur le premier facteur
print((m_1.values[:,0]**2)/disto2_g)

[0.99645377 0.44506654 0.94199086]
```

Pour le 2nd :

```
#COS2 sur le second facteur
print((m_2.values[:,0]**2)/disto2_g)

[0.00012696 0.01647317 0.00056166]
```

1.7 ACP avec d'autres outils (SAS, TANAGRA, R)

Il n'y a pas que Python dans la vie. Nous explorons d'autres outils qui permettent de pratiquer l'analyse en composantes principales dans cette section. D'une manière assez succincte. Il ne s'agit pas de réaliser des tutoriels qui détailleraient la mise en œuvre de l'ACP, mais plutôt des descriptifs simples résumant les commandes et les principales sorties.

1.7.1 Données « Burger King »

Nous traitons les données « Burger King » accessible sur le site « DASL – Data and Story Library ». Il recense des fichiers de données de taille relativement modérée à visée pédagogique, lesquels sont classés par sujets et par techniques applicables. C'est un des sites que je consulte en priorité lorsque je dois préparer des exercices didactiques pour mes travaux dirigés.

La base « Burger King » recense les (p = 10) caractéristiques actives de (n = 17) produits (ITEM) de l'enseigne éponyme (poids, teneur en calories, en calories grasses, en cholestérol, en lipides, en lipides saturées, etc.). Par rapport aux données accessibles en ligne, j'ai opéré une sélection – plus ou moins arbitraire – de produits de taille similaire. J'ai ensuite transformé les indicateurs

(g, mg, etc.) en ratio par rapport à leur poids (serving_size), laquelle sera utilisée en variable illustrative quantitative. Un indicateur précisant la présence de viande (meat) ou non dans l'aliment fera également office de variable illustrative, qualitative cette fois-ci.

Voici pour information le tableau de données. Evitez de vous escrimer les yeux là-dessus. Je rappelle qu'il est accessible dans le fichier « **Data_Methodes_Factorielles.xlsx** » qui accompagne cet ouvrage.

ITEM	Calories	Fat_Cal	Protein	Fat	Sat_Fat	Chol	Sodium	Carbs	Fiber	Sugar	Serving_	Meat
BK__Double_Stacker	3.237	1.8497	0.1734	0.2081	0.0867	0.6358	6.0116	0.1676	0.0058	0.0405	173	yes
BK_CHICKENFRIES(12pc)	2.9412	1.5294	0.1647	0.1706	0.0294	0.3235	9.5882	0.1882	0.0176	0.0059	170	yes
Buck_Double	2.5949	1.2658	0.1519	0.1392	0.0633	0.538	4.6835	0.1772	0.0063	0.038	158	yes
CHICKENTENDERS_Sandwich	3.0986	1.7606	0.0845	0.1972	0.0317	0.2465	4.2958	0.2465	0.007	0.0282	142	yes
ChocolateFudgeSundae	1.6763	0.4046	0.0347	0.0462	0.0405	0.1156	1.3295	0.3064	0.0058	0.2601	173	no
DanishApple	3.6719	1.5625	0.0469	0.1719	0.0781	0.0391	4.0625	0.4844	0.0078	0.2266	128	no
Double_Cheeseburger	2.6316	1.345	0.152	0.152	0.0702	0.5556	5.614	0.1696	0.0058	0.0351	171	yes
Double_Hamburger	2.4658	1.0959	0.1507	0.1233	0.0548	0.4795	3.5616	0.1918	0.0068	0.0411	146	yes
Double_Bacon_Cheeseburger	2.8177	1.4917	0.1713	0.1657	0.0773	0.5801	6.3536	0.1602	0.0055	0.0387	181	yes
GardenSalad(nochicken)	0.4321	0.216	0.0247	0.0247	0.0154	0.0617	0.6173	0.0432	0.0185	0.0123	162	no
KRAFT_MacaroniandCheese	1.4159	0.3982	0.0619	0.0442	0.0133	0.0885	3.0088	0.1947	0.0088	0.0442	113	no
MuffinBlueberry	3.6283	1.8584	0.0442	0.2035	0.0398	0.7522	3.5398	0.4248	0.0177	0.3097	113	no
Original_Chicken_Sandwichw/oMayo	2.2105	0.7368	0.1263	0.0842	0.0184	0.2368	6.3684	0.2421	0.0158	0.0211	190	yes
Rodeo_Cheeseburger	2.7344	1.25	0.125	0.1328	0.0547	0.3516	4.6875	0.2891	0.0156	0.0703	128	yes
SpicyCHICK_NCRISP_Sandwich	3.3824	1.9853	0.0956	0.2206	0.0368	0.2206	5.9559	0.25	0.0147	0.0294	136	yes
SpicyCHICK_NCRISP_Sandwichw/oMayo	2.6316	0.9649	0.1053	0.1053	0.0219	0.1754	5.8772	0.307	0.0175	0.0351	114	yes
Tacos(2)	1.8966	1.1494	0.0805	0.1322	0.046	0.1724	4.3103	0.1034	0.0287	0.023	174	yes

Figure 21 - Données "Burger King"

1.7.2 ACP avec SAS – PROC PRINCOMP

Les procédures PRINCOMP et FACTOR permettent de réaliser une analyse en composantes principales sous SAS. La seconde, plus générique, plus complète, intègre l'analyse en facteurs principaux (section 2.6). Nous y reviendrons dans le chapitre suivant (chapitre 2). Pour l'heure, nous nous intéressons à PRINCOMP, exclusivement dédiée à l'ACP.

Par défaut, PRINCOMP réalise une ACP normée. Voici la commande utilisée…

```
PROC PRINCOMP DATA=BURGER VARDEF=N PLOTS=ALL;
      VAR Calories Fat_Cal Protein Fat Sat_Fat Chol Sodium Carbs Fiber Sugar;
      ID ITEM;
RUN;
```

… avec les options :

- (VARDEF = N) précise d'utiliser les variances population (1/n) plutôt qu'échantillon.

- (PLOTS = ALL) pour produire tous les graphiques possibles. Seule une partie est reprise dans notre descriptif.
- VAR liste les variables actives.
- ID indique la colonne des identifiants affichés dans les cartes des individus.

Les sorties standards comportent plusieurs sections.

Statistiques descriptives (moyennes, écart-type par variable) et matrice des corrélations. Difficile de discerner quoique ce soit dès que le nombre de variables est un tant soit peu élevé. On s'inquiètera surtout si des anomalies manifestes apparaissent, par exemple un écart-type nul, ou encore une valeur 1 en dehors de la diagonale principale dans la matrice des corrélations.

Tableau des valeurs propres. Il est conforme à la présentation usuelle avec les proportions et proportions cumulées (Figure 22).

Valeurs propres de la matrice de corrélation

	Valeur propre	Différence	Proportion	Cumulé
1	4.52737681	2.08191336	0.4527	0.4527
2	2.44546345	1.03535743	0.2445	0.6973
3	1.41010602	0.73658269	0.1410	0.8383
4	0.67352333	0.18993648	0.0674	0.9056
5	0.48358685	0.13679541	0.0484	0.9540
6	0.34679144	0.25281791	0.0347	0.9887
7	0.09397353	0.07628345	0.0094	0.9981
8	0.01769007	0.01645245	0.0018	0.9999
9	0.00123763	0.00098675	0.0001	1.0000
10	0.00025088		0.0000	1.0000

Figure 22 – Tableau des valeurs propres – SAS PRINCOMP – Données "Burger King"

Scree plot (éboulis des valeurs propres). Couplé avec le « scree plot » et l'évolution de la variance expliquée (Figure 23), il nous donne une idée sur le bon nombre de facteurs à retenir.

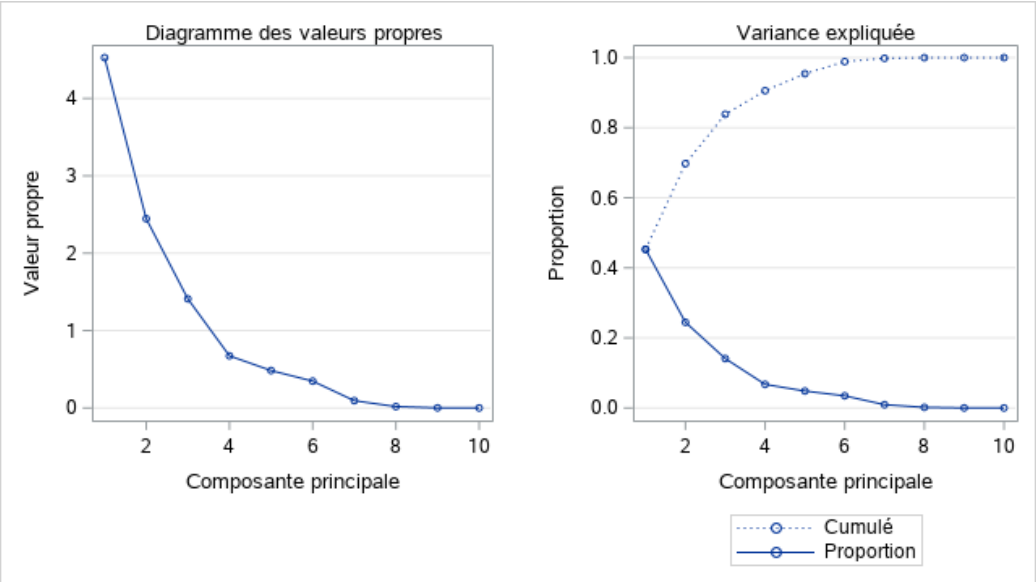


Figure 23 – Eboulis des valeurs propres et évolution de la variance expliquée – SAS PRINCOMP – Données "Burger King"

Le coude est au niveau de la 4ème valeur propre, qui restitue une faible fraction de la variance (6.74%). Nous sommes plutôt enclins à choisir (q = 3) facteurs pour le reste de l'analyse.

Tableau des vecteurs propres. Le tableau des vecteurs propres est un peu brut de fonderie. Rappelons que ses coefficients sont proportionnels aux corrélations. Il nous est donc facile (plus ou moins quand-même, surtout si leur nombre est très élevé) de repérer les variables qui influent sur les facteurs.

Vecteurs propres											
		Prin1	Prin2	Prin3	Prin4	Prin5	Prin6	Prin7	Prin8	Prin9	Prin10
Calories	Calories	0.429586	0.200634	0.172093	-.159961	-.016831	-.053051	0.235316	0.064580	-.717303	-.377932
Fat_Cal	Fat_Cal	0.433175	0.068563	0.204020	0.141989	-.316716	-.222347	-.067957	0.093392	0.572157	-.505393
Protein	Protein	0.296757	-.443465	-.149418	-.168099	0.303688	0.182142	0.246011	0.674128	0.140785	0.067075
Fat	Fat	0.432374	0.057723	0.202147	0.146825	-.333877	-.224672	-.124449	0.133506	-.101319	0.736051
Sat_Fat	Sat_Fat	0.324921	0.031652	-.439768	0.132195	-.347418	0.711817	0.023914	-.233433	0.006562	0.008029
Chol	Chol	0.332742	-.087575	-.263485	0.561567	0.540462	-.267616	0.128021	-.341138	-.004796	0.018082
Sodium	Sodium	0.308206	-.297547	0.345113	-.370276	0.317977	0.209280	-.533716	-.359760	0.031643	0.008856
Carbs	Carbs	0.139828	0.546998	0.141288	-.363616	0.271706	0.116318	0.477273	-.186168	0.354969	0.230773
Fiber	Fiber	-.145802	-.061283	0.670999	0.520801	0.087343	0.447694	0.198688	0.074381	-.020693	-.004188
Sugar	Sugar	0.026680	0.595722	-.115957	0.176208	0.316775	0.167435	-.543064	0.417553	-.022964	-.041612

Figure 24 – Tableau des vecteurs propres – SAS PRINCOMP – Données "Burger King"

Cercles des corrélations. Plus facile sera la lecture du « cercle » des corrélations (il manque le tracé du cercle, mais il s'agit bien de la représentation des variables). SAS les sert par paires, nous nous en tenons aux 3 premiers (1 vs. 2, 1 vs. 3, 2 vs. 3).

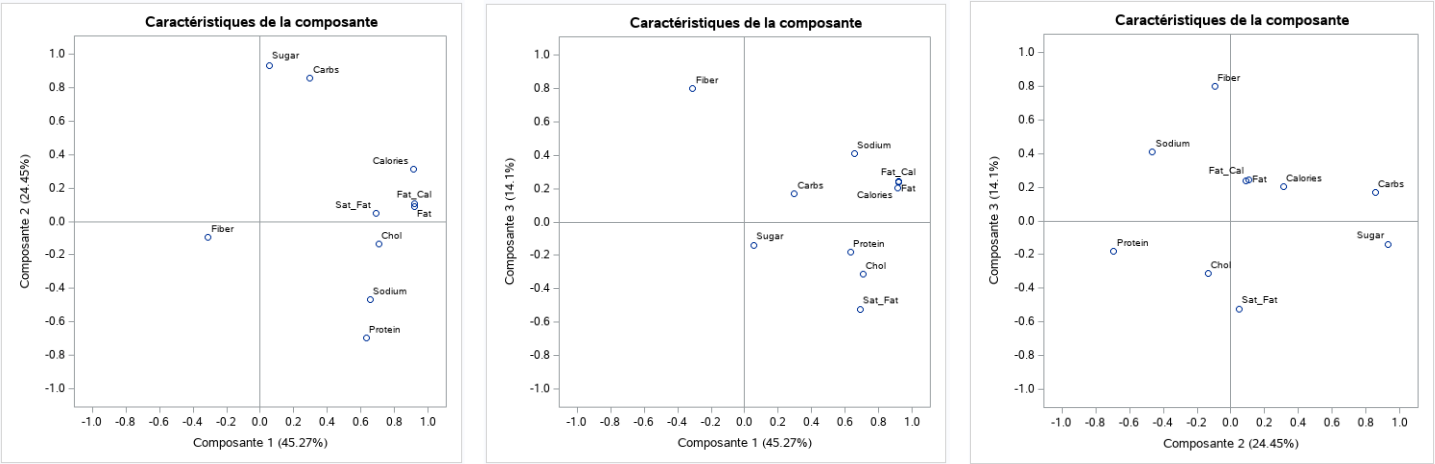


Figure 25 – Cercles des corrélations – SAS PRINCOMP – Données "Burger King"

Sans trop aller dans les détails, puisque notre propos est de faire le tour des fonctionnalités de l'outil et non pas une analyse approfondie des produits de la bouffe rapide, nous noterons que :

- La première composante est une affaire de calories et de graisse grasse chargée en lipides.
- La seconde caractérise une opposition entre les protéines et les glucides. Les variables illustratives nous éclaireront là-dessus.
- La troisième met en lumière la teneur en fibre des aliments, opposée aux graisses saturées.

Cartes des individus. Viennent ensuite les cartes des individus. Pour le premier plan factoriel, nous observons (Figure 26) :

- Avec les « BK_Double_Sytacker » et « Double_Bacon_Burger », on aura tout notre saoul de calories et de lipides saturées. Sans compter le sel. Notre cardiologue se frotte les mains.
- A contrario, on n'en a pas vraiment pour notre argent avec la salade du jardin « GardenSalad (no chiken) ». On ne le saura pas hélas. Le prix ne fait pas partie des variables disponibles.
- La « MuffinBlueBerry » cumule le gras et le sucre. Bon là, il faut carrément prendre un abonnement chez le cardiologue pour avoir des réductions sur les consultations.

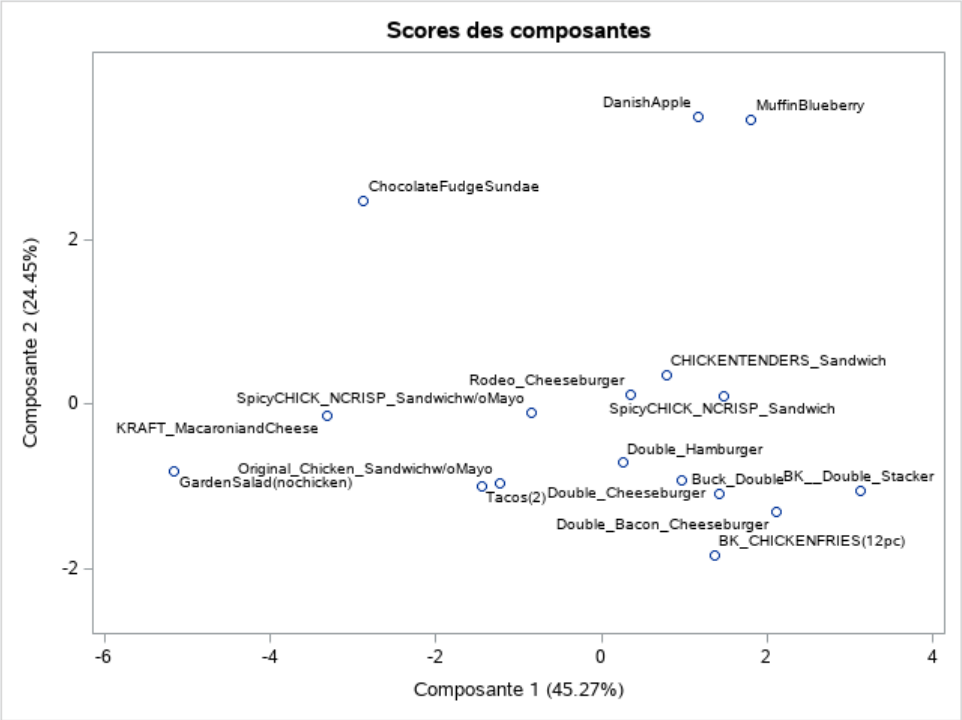


Figure 26 – Carte des individus premier plan factoriel – SAS PRINCOMP – Données "Burger King"

Variables illustratives. Je n'ai pas trouvé comment gérer directement les variables illustratives. Mais comme il est possible de sortir les scores (coordonnées) des individus sur les composantes, un post traitement pour calculer les corrélations (variables illustratives quantitatives), ou les moyennes conditionnelles, rapports de corrélation et valeurs test (qualitatives), est très facile à mettre en œuvre.

Individus supplémentaires. De même, puisque les vecteurs propres et les paramètres des variables (moyennes et écarts-type) sont disponibles, les appliquer sur des individus supplémentaires pour calculer leurs coordonnées factorielles ne devrait pas être insurmontable.

1.7.3 ACP avec TANAGRA

J'ai beaucoup travaillé sur l'analyse en composantes principales dans TANAGRA. Sur les premières versions du logiciel en 2004, je m'étais initialement calé sur PRINCOMP. L'outil a par la suite été enrichi au fur et à mesure, tant au niveau de la présentation des résultats qu'en termes d'outils complémentaires que nous verrons au chapitre 2. Deux tutoriels retracent en détail l'utilisation du composant « **Principal Component Analysis** » de TANAGRA (TUTO 2 ; TUTO 6, qui traite le fichier « Autos »). Nous nous contentons d'une description rapide ici.

Paramétrage. Dans la boîte de dialogue de paramétrage du composant (Figure 27), nous précisons réaliser une ACP normée (Analyze : Correlation Matrix) ; nous intégrons le test de Bartlett (KMO and Bartlett's test of sphericity ; nous parlerons en détail de l'indice KMO plus loin, section 2.3) ; les variables seront triées selon leur pertinence sur les facteurs (Sort variables acc. Loadings), cette fonctionnalité s'avère très utile lorsque nous traitons des grandes bases.

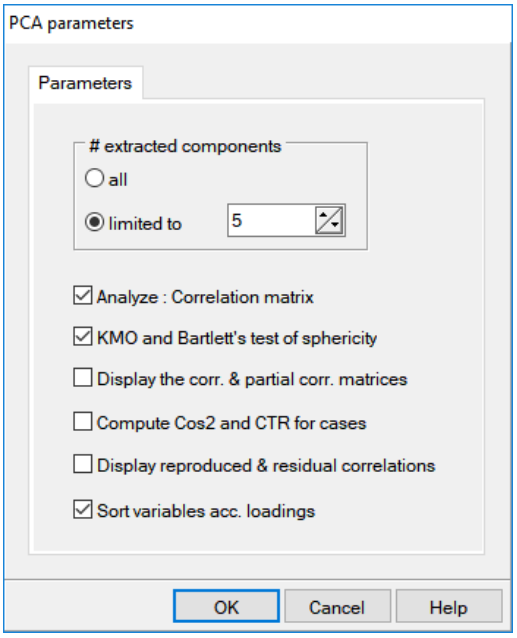


Figure 27 – Boîte de paramétrage – Principal Component Analysis – TANAGRA

Valeurs propres. Le tableau retrace les valeurs propres, à la manière de PRINCOMP.

Eigen values

Matrix trace	10.000000
Average	1.000000

Axis	Eigen value	Difference	Proportion (%)	Histogram	Cumulative (%)
1	4.527377	2.081913	45.27 %		45.27 %
2	2.445463	1.035357	24.45 %		69.73 %
3	1.410106	0.736583	14.10 %		83.83 %
4	0.673523	0.189937	6.74 %		90.56 %
5	0.483587	0.136795	4.84 %		95.40 %
6	0.346791	0.252818	3.47 %		98.87 %
7	0.093974	0.076283	0.94 %		99.81 %
8	0.017690	0.016452	0.18 %		99.99 %
9	0.001238	0.000987	0.01 %		100.00 %
10	0.000251	-	0.00 %		100.00 %
Tot.	10.000000	-	-	-	-

Figure 28 – Tableau des valeurs propres – TANAGRA – Données "Burger King"

Dans l'onglet « **Scree plot** » sont visibles l'éboulis des valeurs propres et l'évolution de la variance expliquée en fonction du nombre de composantes.

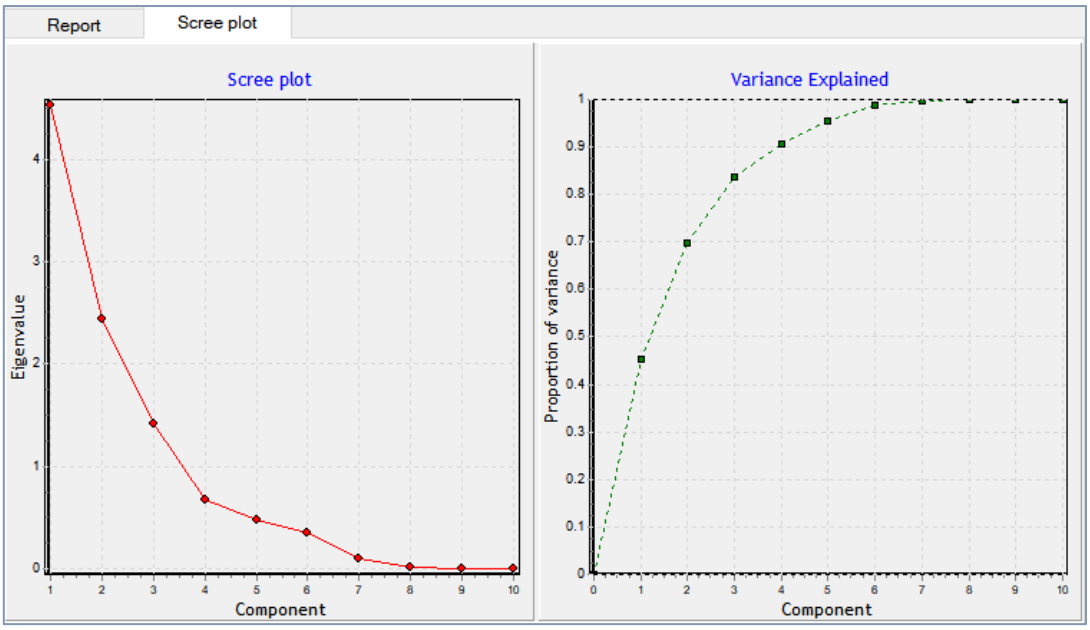


Figure 29 – Eboulis et prop. de variance expliquée – TANAGRA – Données "Burger King"

Détection des composantes pertinentes. Les seuils relatifs aux différentes techniques de détection des composantes pertinentes (section 1.3.1) sont par la suite affichés.

Significance of Principal Components

Global critical values	
Kaiser-Guttman	1
Karlis-Saporta-Spinaki	2.5

Eigenvalue table - Test for significance

Eigenvalues - Significance		
Axis	Eigenvalue	Broken-stick critical values
1	4.527377	2.928968
2	2.445463	1.928968
3	1.410106	1.428968
4	0.673523	1.095635
5	0.483587	0.845635
6	0.346791	0.645635
7	0.093974	0.478968
8	0.017690	0.336111
9	0.001238	0.211111
10	0.000251	0.100000

Figure 30 – Seuils pour l'identification des composantes pertinentes – TANAGRA – Données "Burger King"

Test de Bartlett. De même que les résultats du test de Bartlett

Bartlett's test of sphericity

Bartlett's test	
|CORR.MATRIX|	9.102248E-10
CHISQ	246.3384
d.f.	45
p-value	1.44356E-29

Figure 31 – Test de Bartlett – Données "Burger King"

Corrélation des variables avec les composantes. Le tableau des corrélations inclut les COS2 ponctuels et cumulés. Les variables sont automatiquement triées selon leur contribution aux facteurs. Des codes couleur permettent de distinguer le sens des liaisons. Sur des bases incluant un grand nombre de variables, ce format de présentation est très pratique, il complète avantageusement le cercle des corrélations.

Factor Loadings [Communality Estimates]

Attribute	Axis_1		Axis_2		Axis_3	
-	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)
Fat_Cal	-0.92169	*85 % (85 %)*	-0.10722	*1 % (86 %)*	0.24227	*6 % (92 %)*
Fat	-0.91999	*85 % (85 %)*	-0.09027	*1 % (85 %)*	0.24004	*6 % (91 %)*
Calories	-0.91406	*84 % (84 %)*	-0.31375	*10 % (93 %)*	0.20436	*4 % (98 %)*
Chol	-0.70800	*50 % (50 %)*	0.13695	*2 % (52 %)*	-0.31288	*10 % (62 %)*
Sat_Fat	-0.69135	*48 % (48 %)*	-0.04950	*0 % (48 %)*	-0.52222	*27 % (75 %)*
Sodium	-0.65579	*43 % (43 %)*	0.46530	*22 % (65 %)*	0.40981	*17 % (81 %)*
Protein	-0.63143	*40 % (40 %)*	0.69349	*48 % (88 %)*	-0.17743	*3 % (91 %)*
Sugar	-0.05677	*0 % (0 %)*	-0.93159	*87 % (87 %)*	-0.13770	*2 % (89 %)*
Carbs	-0.29752	*9 % (9 %)*	-0.85539	*73 % (82 %)*	0.16778	*3 % (85 %)*
Fiber	0.31023	*10 % (10 %)*	0.09583	*1 % (11 %)*	0.79680	*63 % (74 %)*
Var. Expl.	4.52738	*45 % (45 %)*	2.44546	*24 % (70 %)*	1.41011	*14 % (84 %)*

Figure 32 – Tableau des corrélations – TANAGRA – Données "Burger King"

Factor Score Coefficients. TANAGRA réunit dans un tableau les éléments (coefficients des fonctions de projection + paramètres de centrage-réduction) nécessaire à un déploiement sur les individus supplémentaires (section 1.5.1). Je m'en sers énormément dans mes enseignements d'initiation à l'analyse factorielle. Les étudiants comprennent assez rapidement les principes de la méthode. Mais lorsqu'on leur fournit un fichier Excel contenant la description d'une d'observation à positionner dans le repère factoriel. C'est une tout autre histoire. Faire la

jonction entre ce tableau des coefficients et le vecteur de description nécessite souvent un petit retour sur le cours avant qu'ils n'intègrent le principe des fonctions scores et la nécessité d'utiliser les moyennes et écarts-type calculés sur l'échantillon de référence pour réaliser l'opération. Je m'ingénie justement à ne mettre qu'un seul individu pour bien montrer qu'utiliser les moyennes et écarts-type de l'échantillon de déploiement n'a absolument aucun sens.

Factor Score Coefficients

Attribute	Mean	Std-dev	Axis_1	Axis_2	Axis_3
Calories	2.5568706	0.8132998	-0.4295863	-0.2006337	0.1720934
Fat_Cal	1.2273059	0.5232124	-0.4331752	-0.0685631	0.2040197
Protein	0.1055059	0.0493569	-0.2967569	0.4434650	-0.1494180
Fat	0.1365706	0.0578927	-0.4323735	-0.0577233	0.2021465
Sat_Fat	0.0457824	0.0226033	-0.3249209	-0.0316519	-0.4397679
Chol	0.3278176	0.2153680	-0.3327416	0.0875746	-0.2634853
Sodium	4.6979706	2.0173383	-0.3082057	0.2975468	0.3451129
Carbs	0.2321294	0.1055080	-0.1398283	-0.5469982	0.1412883
Fiber	0.0121000	0.0065067	0.1458021	0.0612826	0.6709987
Sugar	0.0740765	0.0907744	-0.0266801	-0.5957216	-0.1159569

Figure 33 – Coefficients des fonctions de projection – TANAGRA

Cercle des corrélations. « **Correlation scatterplot** » affiche le cercle des corrélations.

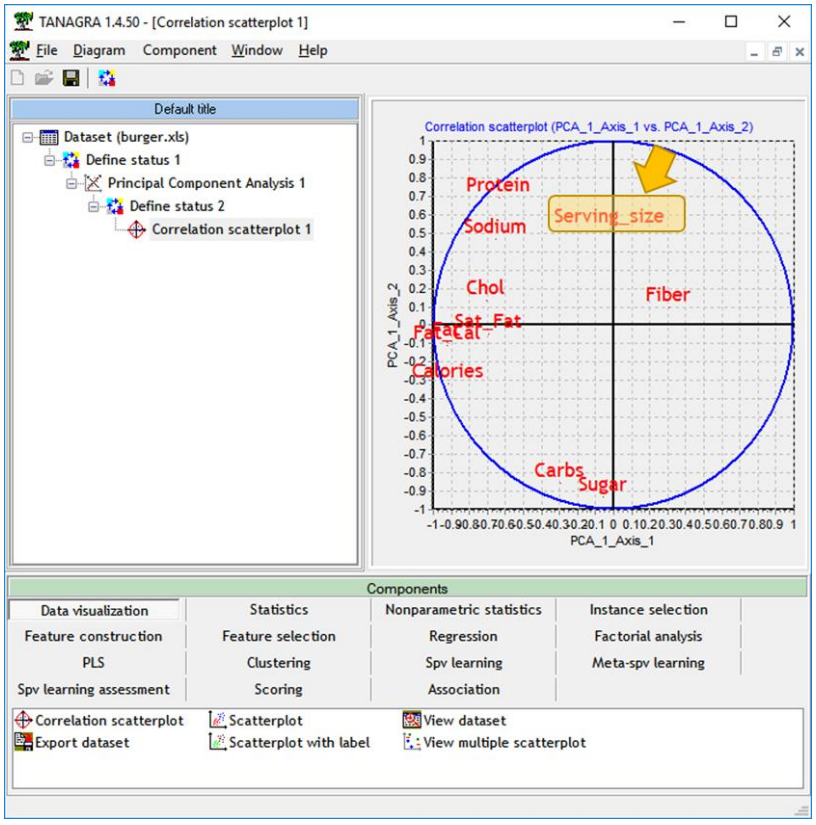


Figure 34 – Cercle des corrélations – Premier plan factoriel – TANAGRA

Le fait qu'il intervienne après l'ACP permet d'intégrer facilement les variables supplémentaires dans la représentation. Nous observons ici que la variable « Serving size » paraît plutôt liée au second facteur (Figure 34).

Carte des individus. Il en est de même pour la carte des individus. Ce qui nous autorise à croiser les facteurs avec tout autre variable. Ci-dessous (Figure 35), nous opposons le 2nd facteur (glucides vs. protéines) à la variable illustrative « Serving size » dans un graphique. Il donne un éclairage nouveau sur le type des produits de notre échantillon. Nous avons (peut-être hein, c'est à vous de voir…) deux populations dans notre fichier. Ceux à gauche correspondent à des desserts. Est-ce qu'on peut les mélanger aux aliments salés dans notre analyse ? La question reste ouverte. Quoiqu'il en soit, pour en revenir à l'outil, la représentation des étiquettes des observations est possible si leur nombre n'excède pas 255 (« **Scatterplot with label** » de TANAGRA).

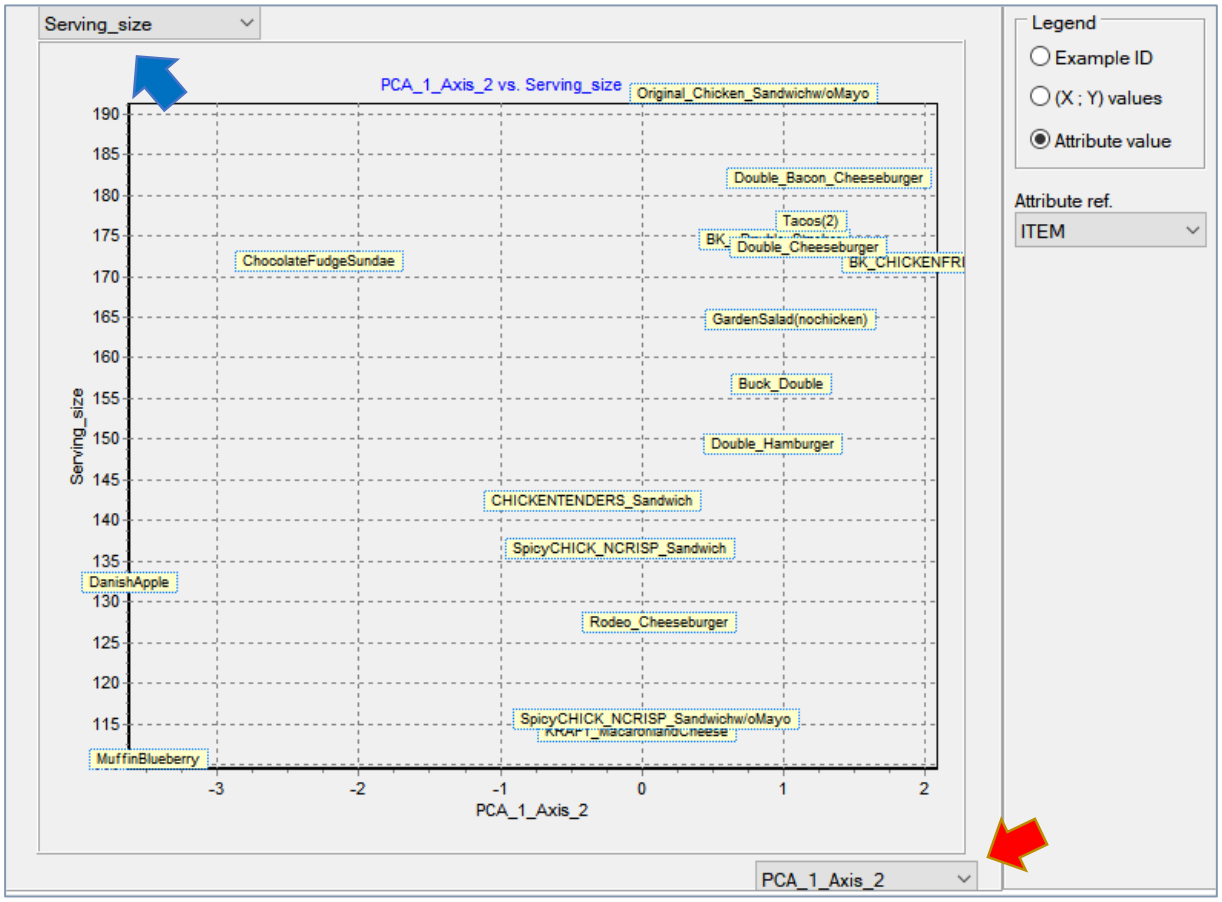


Figure 35 – Représentation des individus – 2nd facteur vs. Servig Size – TANAGRA

Variable illustrative quantitative. Nous avions vu ci-dessus que les variables illustratives peuvent intervenir dans les graphiques pour analyser finement leurs associations avec les facteurs. Si elles sont nombreuses, il faut une automatisation. L'outil « **Linear Correlation** » permet de calculer et éprouver les liaisons d'un ensemble de variables quelconques avec les facteurs. Nous constatons ainsi que « Serving Size » est en relation avant tout avec le 2nd facteur (Figure 36).

Results

Y	X	r	r²	t	Pr(>|t|)
Serving_size	PCA_1_Axis_1	-0.0167	0.0003	-0.0647	0.9492
Serving_size	PCA_1_Axis_2	0.5233	0.2738	2.3782	0.0311
Serving_size	PCA_1_Axis_3	-0.2453	0.0602	-0.9800	0.3426
Serving_size	PCA_1_Axis_4	-0.0875	0.0077	-0.3403	0.7383
Serving_size	PCA_1_Axis_5	-0.0001	0.0000	-0.0003	0.9998

Figure 36 – Corrélation de la variable illustrative avec les facteurs – TANAGRA

Variable illustrative qualitative. La prise en compte des variables illustratives qualitatives indépendamment de l'outil d'ACP autorise également beaucoup de souplesse (Figure 37).

Results

Attribute_Y	Attribute_X	Description				Statistical test		
PCA_1_Axis_1	Meat	Value	Examples	Average	Std-dev	Variance decomposition		
		yes	12	-0.6974	1.3661	Source	Sum of square	d.f.
		no	5	1.6738	3.0245	BSS	19.8456	1
		All	17	0.0000	2.1932	WSS	57.1198	15
						TSS	76.9654	16
						Significance level		
						Statistics	Value	Proba
						Fisher's F	5.211579	0.037441
PCA_1_Axis_2	Meat	Value	Examples	Average	Std-dev	Variance decomposition		
		yes	12	0.7023	0.6649	Source	Sum of square	d.f.
		no	5	-1.6854	2.0365	BSS	20.1209	1
		All	17	0.0000	1.6119	WSS	21.4519	15
						TSS	41.5729	16
						Significance level		
						Statistics	Value	Proba
						Fisher's F	14.069317	0.001927
PCA_1_Axis_3	Meat	Value	Examples	Average	Std-dev	Variance decomposition		
		yes	12	0.1299	1.3548	Source	Sum of square	d.f.
		no	5	-0.3118	0.8792	BSS	0.6885	1
		All	17	0.0000	1.2240	WSS	23.2833	15
						TSS	23.9718	16
						Significance level		
						Statistics	Value	Proba
						Fisher's F	0.443546	0.515528

Figure 37 – Traitement des variables illustratives qualitatives – TANAGRA

Avec l'outil « **One-Way ANOVA** », nous constatons que la différenciation « Meat / No Meat » (produit contenant de la viande ou pas) se joue surtout sur le second facteur. Le graphique croisant le 2nd facteur avec « Serving Size » nous avait déjà un peu éclairer là-dessus. Il oppose les aliments sucrés (nécessairement sans viande) et salés (qui peuvent être éventuellement à base de viande).

Pour récapituler, voici le diagramme de traitements sous TANAGRA (Figure 38).

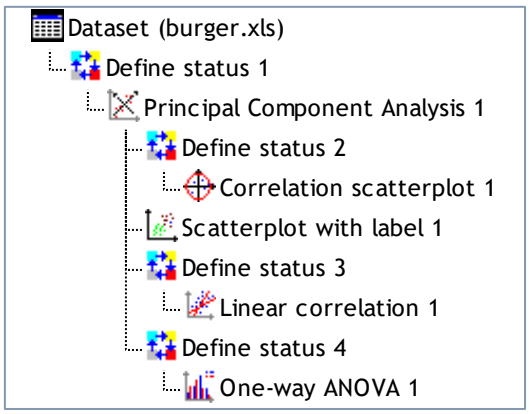


Figure 38 – Diagramme de traitements pour l'ACP – TANAGRA – Données "Burger King"

1.7.4 ACP avec R – Package « FactoMineR »

Plusieurs packages intègrent l'analyse en composantes principales sous R, « FactoMiner » est de ceux qui font référence. Les traitements des individus supplémentaires et variables illustratives sont prévus nativement. Il suffit de se laisser guider.

Dans cette section, nous importons les données sous R, nous lançons l'ACP avec la fonction PCA et nous faisons le tour des sorties proposées.

Nous utilisons le package « openxlsx » pour charger la feuille Excel. Les variables illustratives quantitatives et qualitatives sont respectivement en 11ème et 12ème position.

```
#chargement des données - BURGER, feuille Excel numéro 4
library(openxlsx)
D <- read.xlsx("Data_Methodes_Factorielles.xlsx",sheet=4,rowNames=TRUE)

#vérification de la structure
str(D)

## 'data.frame':    17 obs. of  12 variables:
##  $ Calories    : num  3.24 2.94 2.59 3.1 1.68 ...
```

```
##  $ Fat_Cal     : num  1.85 1.529 1.266 1.761 0.405 ...
##  $ Protein     : num  0.1734 0.1647 0.1519 0.0845 0.0347 ...
##  $ Fat         : num  0.2081 0.1706 0.1392 0.1972 0.0462 ...
##  $ Sat_Fat     : num  0.0867 0.0294 0.0633 0.0317 0.0405 0.0781 ...
##  $ Chol        : num  0.636 0.324 0.538 0.246 0.116 ...
##  $ Sodium      : num  6.01 9.59 4.68 4.3 1.33 ...
##  $ Carbs       : num  0.168 0.188 0.177 0.246 0.306 ...
##  $ Fiber       : num  0.0058 0.0176 0.0063 0.007 0.0058 0.0078 0.0058 ...
##  $ Sugar       : num  0.0405 0.0059 0.038 0.0282 0.2601 ...
##  $ Serving_size: num  173 170 158 142 173 128 171 146 181 162 ...
##  $ Meat        : chr  "yes" "yes" "yes" "yes" ...
```

Il faut installer « FactoMiner » au préalable si la librairie n'est pas présente sur votre machine.

Nous la chargeons ensuite et nous faisons appel à la fonction PCA.

```
#chargement de la librairie FactoMineR
library(FactoMineR)

#ACP - ncp = 3 facteurs demandés – quanti.sup = 11e var -> Serving_size
#quali.sup = 12e variable -> Meat
acp <- PCA(D,ncp=3,quanti.sup=11,quali.sup=12)
```

Par défaut, l'outil affiche la carte des individus dans le premier plan factoriel (Figure 39).

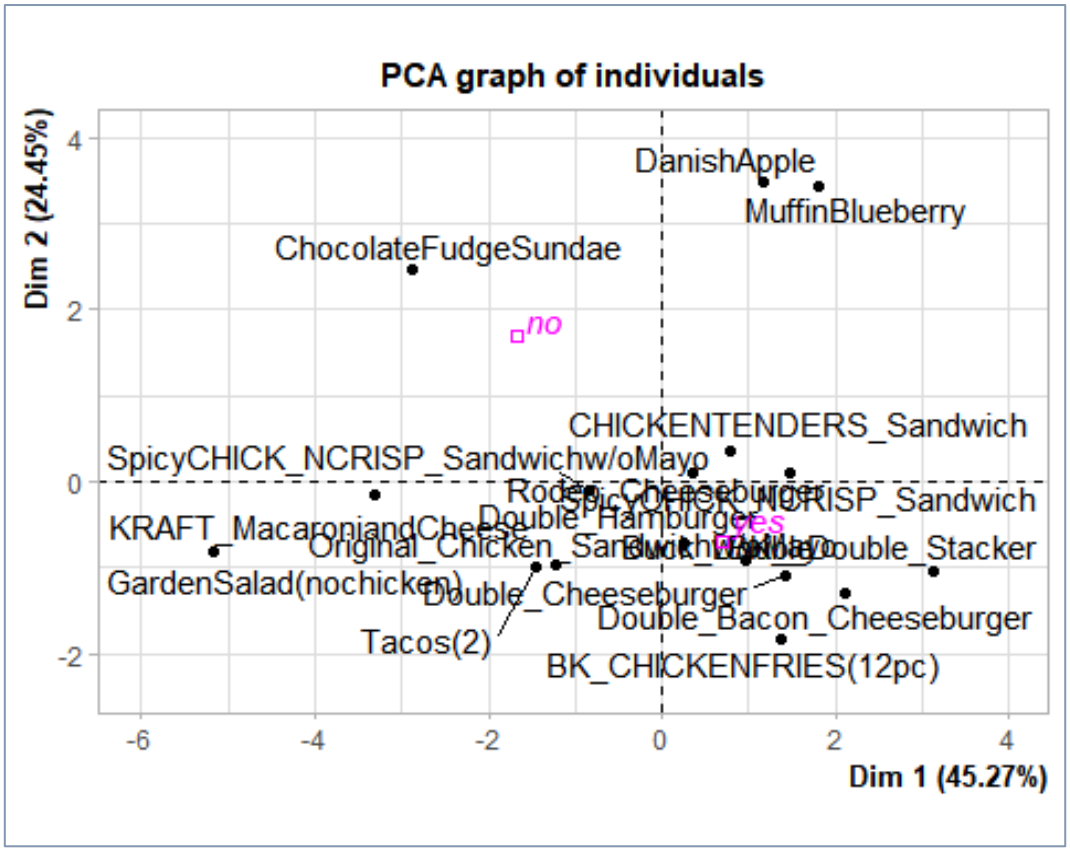


Figure 39 – Carte des individus – FactoMiner

Les moyennes conditionnelles liées aux modalités de la variable illustrative qualitative (Meat = yes ou no) sont automatiquement intégrées dans le graphique.

Puis, nous disposons – toujours automatiquement – du cercle des corrélations du premier plan factoriel, intégrant également l'illustrative « Serving Size » (Figure 40).

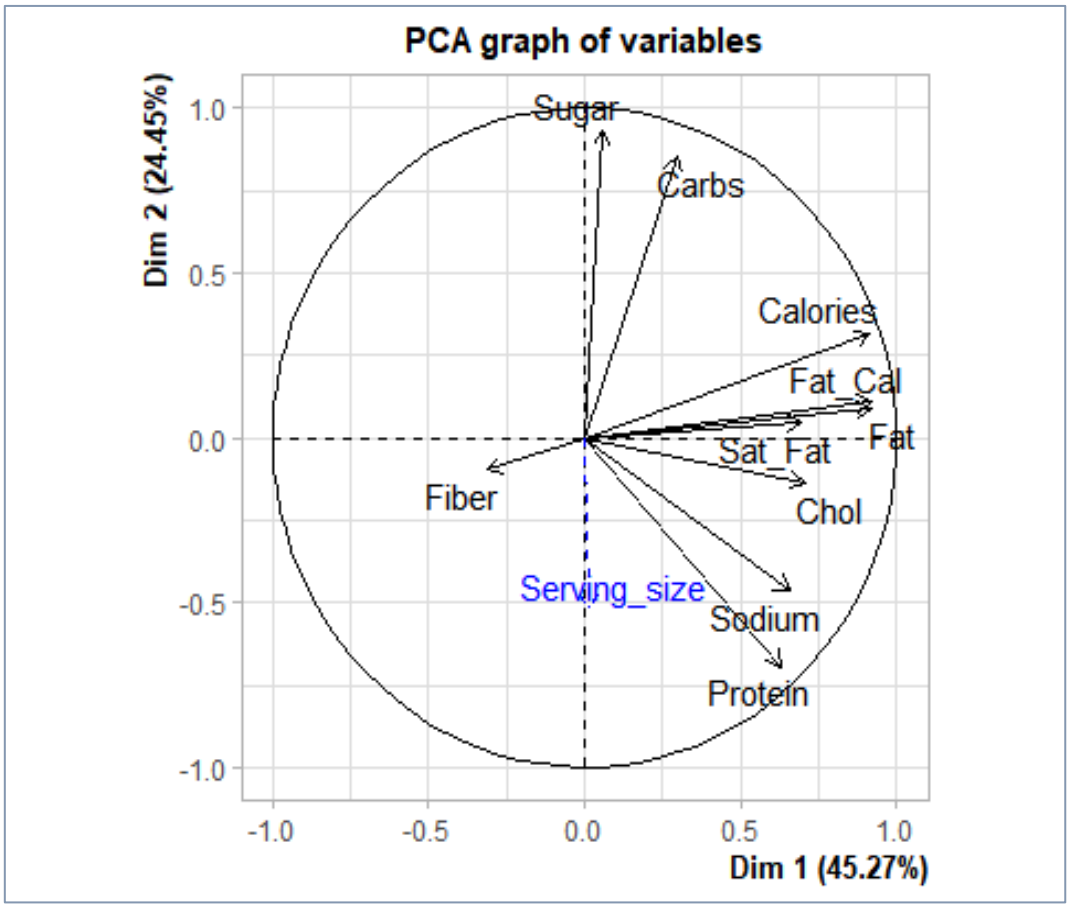


Figure 40 – Cercle des corrélation – FactoMineR

La fonction summary() détaille les résultats de l'analyse.

```
#ACP - ncp = 2 facteurs demandés
summary(acp)

##
## Call:
## PCA(X = D, ncp = 3, quanti.sup = 11, quali.sup = 12)
##
##
## Eigenvalues – Tableau des valeurs propre, % et % cumulé
##                        Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7
## Variance               4.527   2.445   1.410   0.674   0.484   0.347   0.094
## % of var.             45.274  24.455  14.101   6.735   4.836   3.468   0.940
## Cumulative % of var.  45.274  69.728  83.829  90.565  95.401  98.868  99.808
##                        Dim.8   Dim.9  Dim.10
## Variance               0.018   0.001   0.000
```

```
## % of var.              0.177   0.012   0.003
## Cumulative % of var.  99.985  99.997 100.000
##
## Individuals (the 10 first) - Coordonnées factorielles des 10 premiers ind.
## Avec les COS2 et les CTR
##                                Dist    Dim.1    ctr   cos2    Dim.2    ctr
## BK__Double_Stacker          |  3.568 |  3.127 12.708  0.768 | -1.047  2.635
## BK_CHICKENFRIES(12pc)       |  3.193 |  1.367  2.427  0.183 | -1.833  8.082
## Buck_Double                 |  1.916 |  0.971  1.226  0.257 | -0.926  2.060
## CHICKENTENDERS_Sandwich     |  2.054 |  0.785  0.800  0.146 |  0.347  0.289
## ChocolateFudgeSundae        |  4.194 | -2.871 10.710  0.469 |  2.462 14.580
## DanishApple                 |  4.122 |  1.175  1.793  0.081 |  3.487 29.240
## Double_Cheeseburger         |  2.232 |  1.421  2.625  0.406 | -1.083  2.820
## Double_Hamburger            |  1.697 |  0.262  0.089  0.024 | -0.716  1.233
## Double_Bacon_Cheeseburger   |  2.833 |  2.108  5.772  0.554 | -1.309  4.120
## GardenSalad(nochicken)      |  5.389 | -5.164 34.651  0.918 | -0.820  1.617
##
##                              cos2    Dim.3    ctr   cos2
## BK__Double_Stacker          0.086 | -1.211  6.113  0.115 |
## BK_CHICKENFRIES(12pc)       0.330 |  1.895 14.977  0.352 |
## Buck_Double                 0.233 | -1.334  7.426  0.485 |
## CHICKENTENDERS_Sandwich     0.029 |  0.454  0.862  0.049 |
## ChocolateFudgeSundae        0.345 | -1.610 10.814  0.147 |
## DanishApple                 0.715 | -0.017  0.001  0.000 |
## Double_Cheeseburger         0.235 | -1.306  7.113  0.342 |
## Double_Hamburger            0.178 | -1.368  7.801  0.649 |
## Double_Bacon_Cheeseburger   0.213 | -1.310  7.154  0.214 |
## GardenSalad(nochicken)      0.023 | -0.285  0.340  0.003 |
##
## Variables - Corrélations des variables, avec COS2 et CTR
##
##                              Dim.1    ctr   cos2    Dim.2    ctr   cos2
## Calories                    |  0.914 18.454  0.836 |  0.314  4.025  0.098 |
## Fat_Cal                     |  0.922 18.764  0.850 |  0.107  0.470  0.011 |
## Protein                     |  0.631  8.806  0.399 | -0.693 19.666  0.481 |
## Fat                         |  0.920 18.695  0.846 |  0.090  0.333  0.008 |
## Sat_Fat                     |  0.691 10.557  0.478 |  0.049  0.100  0.002 |
## Chol                        |  0.708 11.072  0.501 | -0.137  0.767  0.019 |
## Sodium                      |  0.656  9.499  0.430 | -0.465  8.853  0.217 |
## Carbs                       |  0.298  1.955  0.089 |  0.855 29.921  0.732 |
## Fiber                       | -0.310  2.126  0.096 | -0.096  0.376  0.009 |
## Sugar                       |  0.057  0.071  0.003 |  0.932 35.488  0.868 |
##
##                             Dim.3    ctr   cos2
## Calories                    0.204  2.962  0.042 |
## Fat_Cal                     0.242  4.162  0.059 |
## Protein                    -0.177  2.233  0.031 |
## Fat                         0.240  4.086  0.058 |
## Sat_Fat                    -0.522 19.340  0.273 |
## Chol                       -0.313  6.942  0.098 |
## Sodium                      0.410 11.910  0.168 |
## Carbs                       0.168  1.996  0.028 |
## Fiber                       0.797 45.024  0.635 |
## Sugar                      -0.138  1.345  0.019 |
##
## Supplementary continuous variable - Variable illustrative quantitative
## Avec Corrélation et COS2 (carré de la corrélation)
##                              Dim.1   cos2    Dim.2   cos2    Dim.3   cos2
```

```
## Serving_size              |  0.017  0.000 | -0.523  0.274 | -0.245  0.060 |
##
##
## Supplementary categories – Variable illustrative qualitative
## Avec dist. A l'origine et moyennes cond. des modalités, COS2 et valeur-test
##                              Dist    Dim.1   cos2 v.test    Dim.2   cos2
## no                          |  2.402 | -1.674  0.486 -2.031 |  1.685  0.492
## yes                         |  1.001 |  0.697  0.486  2.031 | -0.702  0.492
##                            v.test    Dim.3   cos2 v.test
## no                          2.783 | -0.312  0.017 -0.678 |
## yes                        -2.783 |  0.130  0.017  0.678 |
```

2 Plus loin avec l'analyse en composantes principales

Généralement, les ouvrages s'arrêtent à la fin du chapitre précédent lors de la présentation de l'analyse en composantes principales. Lorsque je propose de faire une « séance de remise à niveau » sur l'ACP en Master 2 (5ème année), les étudiants me laissent faire, mais je sens bien qu'ils s'interrogent : « mais qu'est-ce qu'il veut nous dire sur l'ACP qu'on ne connaît pas déjà ? ». Il faut dire qu'ils ont ressassé l'affaire depuis la 2ème année pour certains d'entre eux.

Heureusement, au bout d'une quinzaine de minutes, je sens d'un coup un regain d'intérêt palpable quand, ayant fini de passer en revue les différents thèmes précédents (je le fais très vite, il n'y a absolument aucun intérêt à leur refaire des choses qu'ils ont déjà vu par ailleurs), j'attaque les sujets que nous étudierons dans ce chapitre. Ils ont pour point commun de donner un éclairage complémentaire sur la pratique de l'ACP. Ils nous permettent d'approfondir nos analyses et de mieux mettre en évidence les informations que recèlent les données.

2.1 Techniques avancées pour identifier le nombre de facteurs

Le choix du nombre de facteurs à retenir est très important en ACP disions-nous plus haut (section 1.3.1). Nous avions étudié à cette occasion plusieurs approches qui ne nécessitent pas de recalculs à partir des données.

Dans cette section, nous étudions les techniques de rééchantillonnage (TUTO 3, section 8, voir en particulier la bibliographie du tutoriel). L'idée est de réitérer l'analyse en composantes principales sur différentes versions des données pour déterminer empiriquement : dans les deux

premiers cas (sections 2.1.1 et 2.1.2), les seuils d'acceptations des valeurs propres ; dans le troisième cas (section 2.1.3), une intervalle de confiance des valeurs propres, qui fait office de test de significativité (sont acceptés les facteurs dont les valeurs propres sont « significativement » supérieures à 1 dans une ACP normée).

2.1.1 Analyse parallèle

Principe. L'analyse parallèle cherche à obtenir les seuils d'acceptation des valeurs propres par simulation. On calcule les valeurs propres sur un fichier de données possédant les mêmes caractéristiques « n » (nombre d'observations) et « p » (nombre de variables), mais sous l'hypothèse nulle d'indépendance entre les variables. La procédure repose sur une approche Monte Carlo.

Voici les principales étapes :

0. Réaliser une ACP sur notre jeu de données, collecter les valeurs propres (λ_k).
1. Générer aléatoirement un jeu de données de mêmes dimensions « n » et « p » que notre fichier. Chaque variable suit une loi normale N(0,1).
2. Lancer l'ACP, recueillir la séquence des « p » valeurs propres.
3. Répéter T fois les étapes (1) et (2).
4. Calculer la moyenne des valeurs propres (μ_k) pour chaque facteur.
5. On considère qu'un facteur est pertinent si ($\lambda_k > \mu_k$).

Si on souhaite être plus exigeant, nous calculons le quantile d'ordre 0.95 [$q_k^{0.95}$] (pour un test à 5%) à l'étape n°4, et nous utilisons ce seuil à l'étape n°5.

Application sur les données « Autos ». Nous menons l'étude sur les données « Autos ». Pour plus de clarté, nous déroulons les étapes depuis l'importation des données. Nous réalisons dans un premier temps l'ACP normée et nous affichons l'éboulis des valeurs propres.

```
#chargement - index_col = 0 pour indiquer que la colonne n°0 est un label
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="DATA_ACP_ACTIF",index_col=0)

#affichage des caractéristiques
print(D.info())
```

```
#nombre de variables
p = D.shape[1]

#nombre d'observations
n = D.shape[0]

#matrice des X
X = D.values
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18 entries, Alfasud TI to Lada 1300
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   CYL     18 non-null     int64
 1   PUISS   18 non-null     int64
 2   LONG    18 non-null     int64
 3   LARG    18 non-null     int64
 4   POIDS   18 non-null     int64
 5   VMAX    18 non-null     int64
dtypes: int64(6)
memory usage: 1008.0+ bytes
```

```
#importer la classe de calcul
from fanalysis.pca import PCA

#instancier l'objet de calcul
acp = PCA(std_unit=True,row_labels=D.index,col_labels=D.columns)

#lancer les calculs
acp.fit(X)

#valeurs propres - la ligne n°0 est celle qui nous intéresse
print(acp.eig_)
```

```
[[4.42085806e+00 8.56062289e-01 3.73066077e-01 2.13922089e-01
  9.28012120e-02 4.32902727e-02]
 [7.36809677e+01 1.42677048e+01 6.21776796e+00 3.56536815e+00
  1.54668687e+00 7.21504545e-01]
 [7.36809677e+01 8.79486725e+01 9.41664404e+01 9.77318086e+01
  9.92784955e+01 1.00000000e+02]]
```

La 1ère valeur propre ($\lambda_1 = 4.42$), la seconde ($\lambda_2 = 0.856$), etc.

```
%matplotlib inline

#librairie graphique
import matplotlib.pyplot as plt

#graphique - éboulis des v.p.
fig, ax = plt.subplots(figsize=(5,5))
ax.plot(range(1,p+1),acp.eig_[0],".-")
ax.set_xlabel("Nb. facteurs")
ax.set_ylabel("Val. propre")
ax.set_title("Scree plot")

plt.show()
```

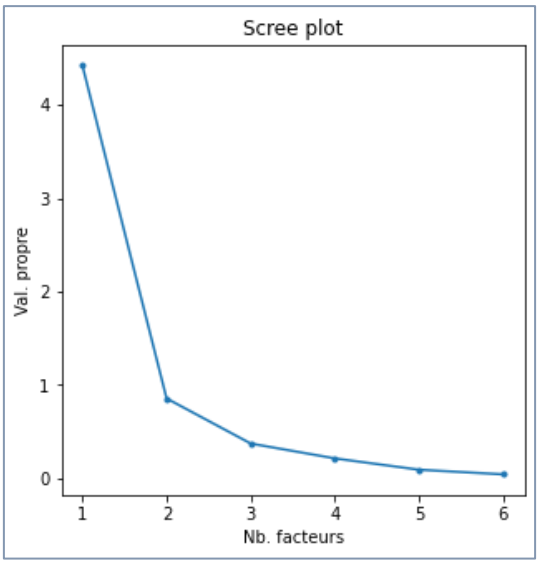


Figure 41 – Eboulis des v.p. – Données "Autos" – ACP normée

Nous créons une version des données sous Ho pour comprendre le principe : chaque colonne, avec « n » valeurs, est générée aléatoirement suivant une loi normale (0, 1) ; elles (les colonnes) sont ensuite assemblées dans une structure data frame de la librairie « Pandas » ; spécifier les mêmes noms de variables que la base originelle n'est pas indispensable.

```
#import numpy
import numpy

#génération d'un jeu de données aléatoire
dfRnd = pandas.DataFrame(numpy.random.normal(size=(n,p)),columns=D.columns)
print(dfRnd)
```

```
         CYL     PUISS      LONG      LARG     POIDS      VMAX
0   0.684290  0.458590 -0.874872 -0.428761  0.411120  0.782854
1  -0.247597  1.000708 -1.374008 -0.293345  0.765486 -0.290459
2   0.156849  0.053340  0.550891 -0.765567  1.976240  0.076709
3  -1.111681  1.430449 -0.325686  0.693433  1.275941  1.433965
4  -0.471864  0.556294  0.732748  1.173023 -0.813639  0.300650
5  -0.953868 -0.545724 -0.187413  0.491266 -0.174796  1.661552
6   1.040280 -0.826886 -0.395895 -1.489726 -0.239506  0.292094
7   0.632855  0.773890  0.821613 -1.282169  2.760042 -0.154054
8   3.018858  0.247155 -0.117227  0.695464 -0.427619 -0.109417
9  -0.641985 -1.475389 -0.244709  1.848373 -0.106885  0.882752
10  0.091420 -0.058579 -0.579384 -0.351569  0.062986 -0.346489
11 -0.476201 -1.091564  0.029138 -1.328488  2.443171 -0.975464
12 -0.228892 -1.689826  0.143777  1.071191 -1.102371  1.386065
13  2.028515  0.327974  0.062028 -0.993507 -0.205182 -0.796925
14  1.395241  0.258581 -0.708619 -0.736578  0.182213  0.492361
15 -1.395860  0.684979  0.086123 -0.767820 -0.703503  1.539871
16 -0.451712 -0.589282  0.047663 -1.522865  0.701168  0.955639
17  0.117394  0.136842 -1.348196 -1.025415  0.109330 -1.590271
```

Nous pouvons dés lors réitérer (T = 10000) fois l'analyse parallèle. Les séquences de valeurs propres sont stockées dans une matrice (mRes) de dimension (T = 10000, p = 6).

```
#nombre de réplication
T = 10000

#matrice de résultats
mRes = numpy.zeros(shape=(T,p))

#répéter T fois
for i in range(T):

    #génération des données sous H0
    dfRnd = pandas.DataFrame(numpy.random.normal(size=(n,p)),columns=D.columns)

    #lancer l'ACP
    tempAcp = PCA(std_unit=True,row_labels=D.index,col_labels=D.columns)
    tempAcp.fit(dfRnd.values)

    #récupérer les valeurs propres
    mRes[i,:] = tempAcp.eig_[0]

#quantile à 95% des val.propres sous H0
seuils = numpy.quantile(mRes,q=0.95,axis=0)
print(seuils)
```

```
[2.26807573 1.64595914 1.2531848  0.97134689 0.72330397 0.48193917]
```

Nous choisissons comme seuils d'acceptation – valeurs propres théoriques sous Ho – le quantile d'ordre 95% des valeurs propres générées sous l'hypothèse d'indépendance des variables. **Le seuil pour la 1ère est 2.268, pour la seconde 1.654, etc.** Rien de mieux qu'un graphique pour comparer les valeurs propres observées et théoriques.

```
#graphique – éboulis des v.p.
fig, ax = plt.subplots(figsize=(5,5))
ax.plot(range(1,p+1),acp.eig_[0],".-")
ax.set_xlabel("Nb. facteurs")
ax.set_ylabel("Val.propre")
ax.set_title("Scree plot avec seuils")

#insertion des seuils
ax.plot(range(1,p+1),seuils,'r.--')

plt.show()
```



Figure 42 – Eboulis des v.p. et seuils de l'analyse parallèle – Données "Autos"

Avec cette procédure, seul le premier facteur serait significatif ($\lambda_1 = 4.42 > 2.268$).

Tables statistiques. Pour les réfractaires à la programmation, il existe des tables statistiques dans certains ouvrages. Les valeurs critiques sont générées selon la même démarche, mis à part qu'elles sont basées sur des réplications plus nombreuses. Dans (Husson et al, 2009, pages 204 à 207), des tables à 95% ont été élaborée avec T = 10000 essais. On lit par exemple (Table A.1) que le premier axe est significatif si la proportion de variance restituée est supérieure à (n = 18, p = 6) 37.6%. Ramené sur notre base, le seuil devient 0.376 x 6 = 2.25, très proche de ce que nous avions trouvé (2.268).

Calcul de la p-value du test. Plutôt que de comparer λ_k avec le quantile d'ordre 0.95 des valeurs produites par la simulation, nous pouvons obtenir directement une sorte de p-value du test en comptabilisant – pour chaque facteur – la proportion des valeurs supérieures à λ_k. Si elle est plus petite que α=0.05, on peut considérer que l'axe est significatif. La décision est totalement cohérente avec la comparaison au quantile.

Analyse parallèle pour l'ACP non normée. La méthode peut être étendue à l'ACP non normée. Il suffit pour cela d'utiliser la moyenne (loc) et l'écart-type (scale) empirique de chaque variable lors de l'appel de la fonction numpy.random.normal() durant la génération des données.

2.1.2 Technique de randomisation

Principe. L'analyse parallèle est robuste par rapport à la normalité (utilisée lors de la génération des données). Néanmoins, elle peut paraître inutilement restrictive. Nous pouvons nous en affranchir en utilisant une stratégie de randomisation pour la génération des données non corrélées, en exploitant les observations disponibles.

Nous formons toujours un échantillon de taille (n x p). Pour chaque variable, traitée de manière indépendante, nous mélangeons aléatoirement les valeurs. De fait, la corrélation entre les variables, si elle existait, est totalement altérée. La procédure est répétée T fois. Les valeurs critiques sont définies à partir du quantile d'ordre 0.95 (ou de la moyenne si on souhaite être moins restrictif).

Voici les étapes de cette procédure que l'on peut voir, à bien des égards, comme une variante de l'analyse parallèle.

0. Réaliser une ACP sur notre jeu de données, collecter les valeurs propres (λ_k).
1. Générer une version du jeu de données où les valeurs sont mélangées aléatoirement dans les colonnes, traitées indépendamment les unes des autres. Les corrélations entre les descripteurs, si elles existaient, sont totalement altérées. En revanche, les moyennes et écarts-type par variable ne sont pas modifiées.
2. Lancer l'ACP, recueillir la séquence des « p » valeurs propres.
3. Répéter T fois les étapes (1) et (2).
4. Calculer la moyenne des valeurs propres (μ_k) pour chaque facteur.
5. On considère qu'un facteur est pertinent si ($\lambda_k > \mu_k$).

Si on souhaite être plus exigeant toujours, nous optons pour le quantile d'ordre 0.95 [$q_k^{0.95}$] (pour un test à 5%) à l'étape n°4, et nous utilisons ce seuil à l'étape n°5.

Manipulation des données. Dans le code Python ci-dessous, pour comprendre le processus de perturbation des données, nous affichons tout d'abord les valeurs initiales de notre fichier de données « Autos », nous calculons les moyennes par variable.

```
#tableau initial de données
print(X)
```

```
[[1350   79  393  161  870  165]
 [1588   85  468  177 1110  160]
 [1294   68  424  168 1050  152]
 [1222   59  412  161  930  151]
 [1585   98  439  164 1105  165]
 [1297   82  429  169 1080  160]
 [1796   79  449  169 1160  154]
 [1565   55  424  163 1010  140]
 [2664  128  452  173 1320  180]
 [1166   55  399  157  815  140]
 [1570  109  428  162 1060  175]
 [1798   82  445  172 1160  158]
 [1998  115  469  169 1370  160]
 [1993   98  438  170 1080  167]
 [1442   80  431  166 1129  144]
 [1769   83  440  165 1095  165]
 [1979  100  459  173 1120  173]
 [1294   68  404  161  955  140]]
```

```
#moyennes par colonne
print(numpy.mean(X,axis=0))
```

```
[1631.66666667   84.61111111  433.5         166.66666667 1078.83333333
  158.27777778]
```

Puis nous mélangeons les valeurs intra-colonnes. Maintenant, le véhicule qui présentait une cylindrée de 1442 (Rancho), se voit attribué une puissance de 128 (celle de la Renault 30), une longueur de 440 (Mazda 9295), etc.

```
#données perturbées
XShuffle = numpy.apply_along_axis(arr=X,axis=0,func1d=lambda x: numpy.random.permutation(x))
print(XShuffle)
```

```
[[1442  128  440  169  930  175]
 [1222   98  404  173 1050  165]
 [1798  115  424  165 1060  158]
 [1998   82  428  177  870  154]
 [1588   59  424  161 1095  173]
 [1769   80  469  169 1160  151]
 [1350   79  445  166 1129  152]
 [1585   82  459  169  955  180]
 [1979   98  449  161 1080  140]
 [1166  109  468  157 1120  144]
 [1294   85  399  173 1320  160]
 [1570   83  438  163 1105  160]
 [1796  100  452  170 1110  140]
 [2664   55  439  164  815  140]
 [1297   68  431  162 1160  160]
 [1565   79  393  168 1010  165]
 [1993   55  429  172 1370  167]
 [1294   68  412  161 1080  165]]
```

Les lignes sont perturbées, mais les moyennes par colonne restent identiques.

```
#et la moyenne
print(numpy.mean(XShuffle,axis=0))
```

```
[1631.66666667   84.61111111  433.5         166.66666667 1078.83333333
  158.27777778]
```

Calculs des seuils de significativité sur les données « Autos ». Nous réitérons l'étude sur les données « Autos » en adoptant le schéma de l'analyse parallèle. Nous effectuons (T = 10000) réplications, le quantile d'ordre 0.95 fait office de seuil d'acceptation des facteurs.

```
#nombre de réplication
T = 10000

#matrice de résultats
mResRand = numpy.zeros(shape=(T,p))

#répéter T fois
for i in range(T):

    #génération des données sous H0
    XShuffle = numpy.apply_along_axis(arr=X,axis=0,func1d=lambda x: numpy.random.permutation(x))

    #lancer l'ACP
    tempAcpRand = PCA(std_unit=True,row_labels=D.index,col_labels=D.columns)
    tempAcpRand.fit(XShuffle)

    #récupérer les valeurs propres
    mResRand[i,:] = tempAcpRand.eig_[0]

#quantile à 95% des val.propres sous H0
seuilsRand = numpy.quantile(mResRand,q=0.95,axis=0)
print(seuilsRand)
```

```
[2.26487264 1.63667585 1.25267298 0.97314427 0.72179874 0.48051084]
```

Nous notons que les seuils sont très proches de ceux de l'analyse parallèle qui, pour rappel, étaient de (2.268, 1.646, 1.253, 0.971, 0.723, 0.482). Les deux approches sont assez similaires, sur nos données tout du moins.

Il ne reste plus qu'à faire apparaître ces nouveaux seuils dans le « scree plot » (Figure 43).

```
#graphique avec les seuils
fig, ax = plt.subplots(figsize=(5,5))
ax.plot(range(1,p+1),acp.eig_[0],".-")
ax.set_xlabel("Nb. facteurs")
ax.set_ylabel("Val.propre")
ax.set_title("Scree plot avec seuils")

#seuils
ax.plot(range(1,p+1),seuilsRand,'m.--')

plt.show()
```

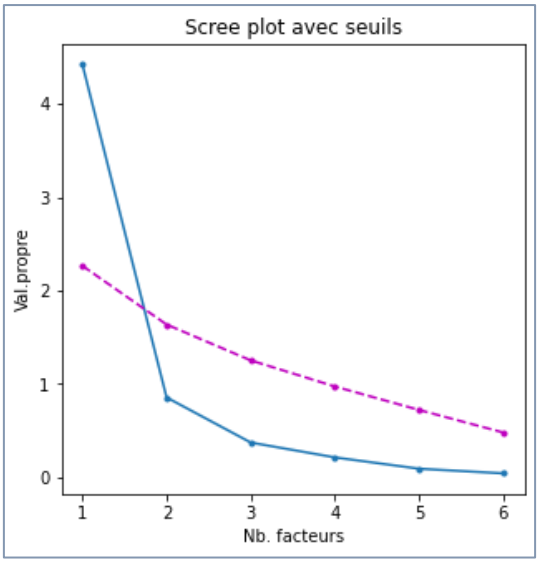


Figure 43 – Eboulis des v.p. et seuils d'acceptation – Données "Autos"

A l'instar de l'analyse parallèle, seul le premier facteur serait licite avec cette technique.

2.1.3 Analyse bootstrap

Appréhender la variabilité associée à chaque valeur propre est l'objectif de la méthode bootstrap. Pour ce faire, nous collectons les différentes versions de λ_k en répétant T fois la procédure suivante : on effectue un tirage avec remise des observations pour obtenir un échantillon de taille « n » (certaines observations reviennent plusieurs fois, d'autres ne sont pas présentes) ; nous calculons l'ACP sur ces données ; nous stockons la valeur propre pour chaque axe.

La décision est basée sur une variante de la règle de Kaiser. Un facteur est jugé pertinent si le quantile d'ordre 0.05 ($\lambda_k^{0.05}$) (la borne basse de l'intervalle de confiance) des valeurs propres bootstrap est supérieure à 1.

Dans le code qui suit, nous affichons tout d'abord pour vérifications les données initiales. Puis, nous montrons comment générer un échantillon bootstrap de taille « n ». Nous comparons les deux dataset.

```
#échantillon initial
print(D)
```

```
                 CYL  PUISS  LONG  LARG  POIDS  VMAX
Modele
Alfasud TI      1350     79   393   161    870   165
Audi 100        1588     85   468   177   1110   160
Simca 1300      1294     68   424   168   1050   152
Citroen GS Club 1222     59   412   161    930   151
Fiat 132        1585     98   439   164   1105   165
Lancia Beta     1297     82   429   169   1080   160
Peugeot 504     1796     79   449   169   1160   154
Renault 16 TL   1565     55   424   163   1010   140
Renault 30      2664    128   452   173   1320   180
Toyota Corolla  1166     55   399   157    815   140
Alfetta 1.66    1570    109   428   162   1060   175
Princess 1800   1798     82   445   172   1160   158
Datsun 200L     1998    115   469   169   1370   160
Taunus 2000     1993     98   438   170   1080   167
Rancho          1442     80   431   166   1129   144
Mazda 9295      1769     83   440   165   1095   165
Opel Rekord     1979    100   459   173   1120   173
Lada 1300       1294     68   404   161    955   140


#un échantillon bootstrap - Pandas possède une fonction dédiée
DBoot = D.sample(n=n,replace=True)
print(DBoot)

                 CYL  PUISS  LONG  LARG  POIDS  VMAX
Modele
Taunus 2000     1993     98   438   170   1080   167
Taunus 2000     1993     98   438   170   1080   167
Alfetta 1.66    1570    109   428   162   1060   175
Simca 1300      1294     68   424   168   1050   152
Peugeot 504     1796     79   449   169   1160   154
Simca 1300      1294     68   424   168   1050   152
Renault 16 TL   1565     55   424   163   1010   140
Opel Rekord     1979    100   459   173   1120   173
Toyota Corolla  1166     55   399   157    815   140
Opel Rekord     1979    100   459   173   1120   173
Taunus 2000     1993     98   438   170   1080   167
Renault 16 TL   1565     55   424   163   1010   140
Opel Rekord     1979    100   459   173   1120   173
Simca 1300      1294     68   424   168   1050   152
Renault 16 TL   1565     55   424   163   1010   140
Fiat 132        1585     98   439   164   1105   165
Fiat 132        1585     98   439   164   1105   165
Toyota Corolla  1166     55   399   157    815   140
```

Dans un tirage avec remise, certains véhicules sont répétés 3 fois (Taunus 2000, Renault 16 TL, Simca 1300, Opel Rekord), d'autres 2 fois (Fiat 132, Toyota Corolla). Plusieurs, présents dans la base initiale, sont absents de l'échantillon bootstrap (Alfasud TI, Citroën GS Club, etc.).

Une fois intégré ce principe, nous pouvons démarrer l'expérimentation. Nous demandons (T = 10000) échantillons bootstrap. Notre référence pour tester la significativité (valeur propre supérieure à 1 ou pas puisque nous travaillons sur une ACP normée) sera le quantile d'ordre 0.05.

```
#nombre de réplication
T = 10000

#matrice de résultats
mResBoot = numpy.zeros(shape=(T,p))

#répéter T fois
for i in range(T):

    #génération des données - une réplication bootstrap
    DBoot = D.sample(n=n,replace=True)

    #lancer l'ACP
    tempAcpBoot = PCA(std_unit=True,row_labels=D.index,col_labels=D.columns)
    tempAcpBoot.fit(DBoot.values)

    #récupérer les valeurs propres
    mResBoot[i,:] = tempAcpBoot.eig_[0]

#quantile à 5% des val.propres
seuilsBoot = numpy.quantile(mResBoot,q=0.05,axis=0)
print(seuilsBoot)
```

```
[3.67465571 0.49767014 0.20560515 0.08830189 0.03451719 0.00687916]
```

Seul le premier facteur est pertinent. C'est toujours mieux avec une représentation graphique.

```
#graphique des bornes basses des intervalle de confiance bootstrap
fig, ax = plt.subplots(figsize=(5,5))
ax.set_xlabel("Nb. facteurs")
ax.set_ylabel("Val. propre")
ax.set_title("Scree plot - Pire cas (5%)")

ax.plot(range(1,p+1),seuilsBoot,"k.-")
ax.plot([1,6],[1,1],'r--') #seuil v.p. = 1
plt.show()
```

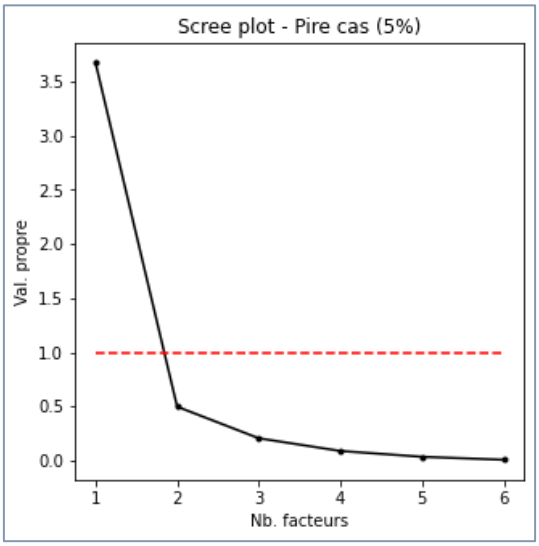


Figure 44 – Borne basse de l'intervalle de confiance bootstrap des v.p. – Données "Autos"

2.1.4 Conclusion sur les techniques de rééchantillonnage

Les fondements scientifiques des techniques de rééchantillonnage pour l'identification des facteurs pertinents sont solides. Il existe une littérature étayée sur le sujet. Mais leur utilité pratique n'est pas toujours évidente, en particulier lors du traitement des grandes bases de données. Dans une phase exploratoire où l'on cherche des solutions en tentant différentes analyses, leur gourmandise en temps de calcul peut se révéler rapidement rédhibitoire, surtout pour un gain finalement peu concluant par rapport aux approches empiriques. Ce problème ne se pose pas sur les petites bases. Pour notre fichier « Autos », la génération et le traitement des 10000 réplications n'ont pris que quelques secondes sur mon antique PC.

2.2 Rotation des facteurs – Rotation VARIMAX

2.2.1 Principe de la rotation orthogonale

Les résultats de l'ACP sont compliqués à lire lorsque l'association des variables aux facteurs est difficile à discerner, confuse. Ce type de résultats, qui est assez fréquent finalement, tient au mécanisme interne de l'analyse en composantes principales. En effet, si l'on s'en tient à la première composante – l'idée est la même pour les suivantes – elle procède à une optimisation mixant les corrélations de l'ensemble des variables avec le facteur (section 1.1.3.3) :

$$\lambda_1 = \sum_{j=1}^{p} r_j^2(F_1)$$

Il n'est pas rare d'obtenir un facteur où les variables lui sont plus ou moins uniformément corrélées, conduisant bien à une valeur optimale de (λ_1), mais difficilement interprétable. C'est ce que nous observons par exemple pour les données « Autos » où les corrélations avec le premier facteur va de 0.905 (CYL) à 0.754 (VMAX). Nous avons alors deux phénomènes compliqués à gérer :

- Les variables présentent des contributions similaires dans la définition du facteur.
- La qualité de représentation des variables est uniformément répartie sur plusieurs facteurs.

L'objectif de la rotation orthogonale est de faire pivoter les facteurs que l'on choisit de traiter (paramètre de l'algorithme) de manière à ce que :

- Les associations entre les variables et les facteurs soient plus tranchés, clarifiant ainsi la nature de ces derniers ;
- En préservant l'orthogonalité entre les facteurs ;
- Et en préservant la proportion cumulée de variance expliquée c.-à-d. la qualité globale de la représentation.

Rotation VARIMAX. La rotation VARIMAX est certainement la plus populaire des techniques de rotations orthogonales qui répond à ces propriétés. Elle cherche à maximiser la variance des carrés des corrélations intra-facteurs c.-à-d. à rendre les plus disparates possibles les contributions des variables afin de faciliter l'interprétation des facteurs, tout en maintenant la qualité de la représentation. Elle repose sur un algorithme itératif.

Voyons l'exemple des données « Autos » pour comprendre l'intérêt du mécanisme. Les calculs ont été réalisés avec TANAGRA (TUTO 8).

L'ACP initiale fournit le tableau des corrélations suivantes pour les (q = 2) premiers facteurs, auquel nous avons adjoint les contributions.

Attribute	Axis_1		Axis_2	
	Corr.	CTR (%)	Corr.	CTR (%)
POIDS	0.905	*18.5*	0.225	*5.89*
CYL	0.893	*18.1*	-0.115	*1.54*
PUISS	0.887	*17.8*	-0.385	*17.29*
LONG	0.886	*17.8*	0.381	*16.96*
LARG	0.814	*15.0*	0.413	*19.90*
VMAX	0.755	*12.9*	-0.574	*38.42*
Var. Expl.	4.42086	*74% (74%)*	0.85606	*14% (88%)*

Figure 45 - ACP - Corrélations et contributions avant rotation - Données "Autos"

Le premier plan factoriel restitue 74% + 14% = 88% de l'information disponible. Toutes les variables contribuent quasi-égalitairement sur le premier facteur (fond jaune, |corrélation| > 0.5 ; fond orange, |corrélation| > 0.75). L'interprétation n'est pas des plus évidente.

La méthode VARIMAX opère une rotation des facteurs dans le repère factoriel (il faut spécifier le nombre [q] de facteurs à manipuler dans le pivotement). Nous visualisons cela sur la matrice des corrélations (Figure 54).

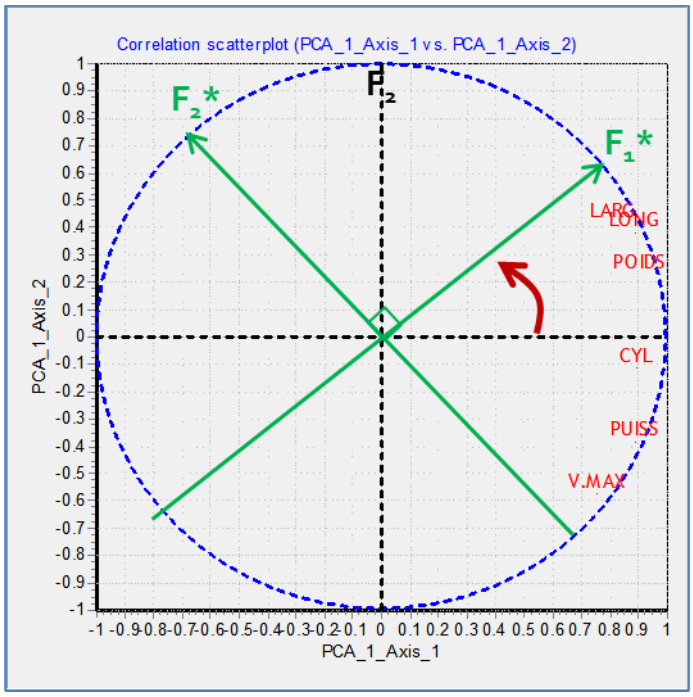


Figure 46 – ACP – Rotation VARIMAX – Cercle des corrélations

Les définitions des facteurs est plus limpide : (F1) avec LARG, LONG et POIDS, désigne l'encombrement des véhicules ; (F2) avec PUISS et VMAX, désigne leurs performances. CYL tient une position intermédiaire, son influence est partagée sur les deux facteurs. Voyons le tableau des corrélations et contributions après rotation (Figure 54 ; Attention : TANAGRA a automatiquement trié les variables pour que l'on distingue mieux les « blocs » d'associations variables-facteurs).

Attribute	Axis_1		Axis_2	
	Corr.	CTR (%)	Corr.	CTR (%)
LONG	0.917	*29.3*	0.298	*3.7*
LARG	0.884	*27.2*	0.226	*2.1*
POIDS	0.829	*23.9*	0.428	*7.6*
CYL	0.596	*12.4*	0.675	*19.0*
VMAX	0.189	*1.2*	0.929	*35.9*
PUISS	0.413	*5.9*	0.874	*31.7*
Var. Expl.	2.87114	*48 % (48 %)*	2.40578	*40 % (88 %)*

Figure 47 – ACP – Corrélations et contributions après rotation VARIMAX – Données "Autos"

La qualité globale est préservée (88%) mais, par rapport à la représentation initiale, elle est mieux dispatchée entre les facteurs (48% + 40% = 88%).

De nouvelles informations intéressantes se font jour lorsque nous représentons les véhicules dans ce nouveau plan factoriel (Figure 54) :

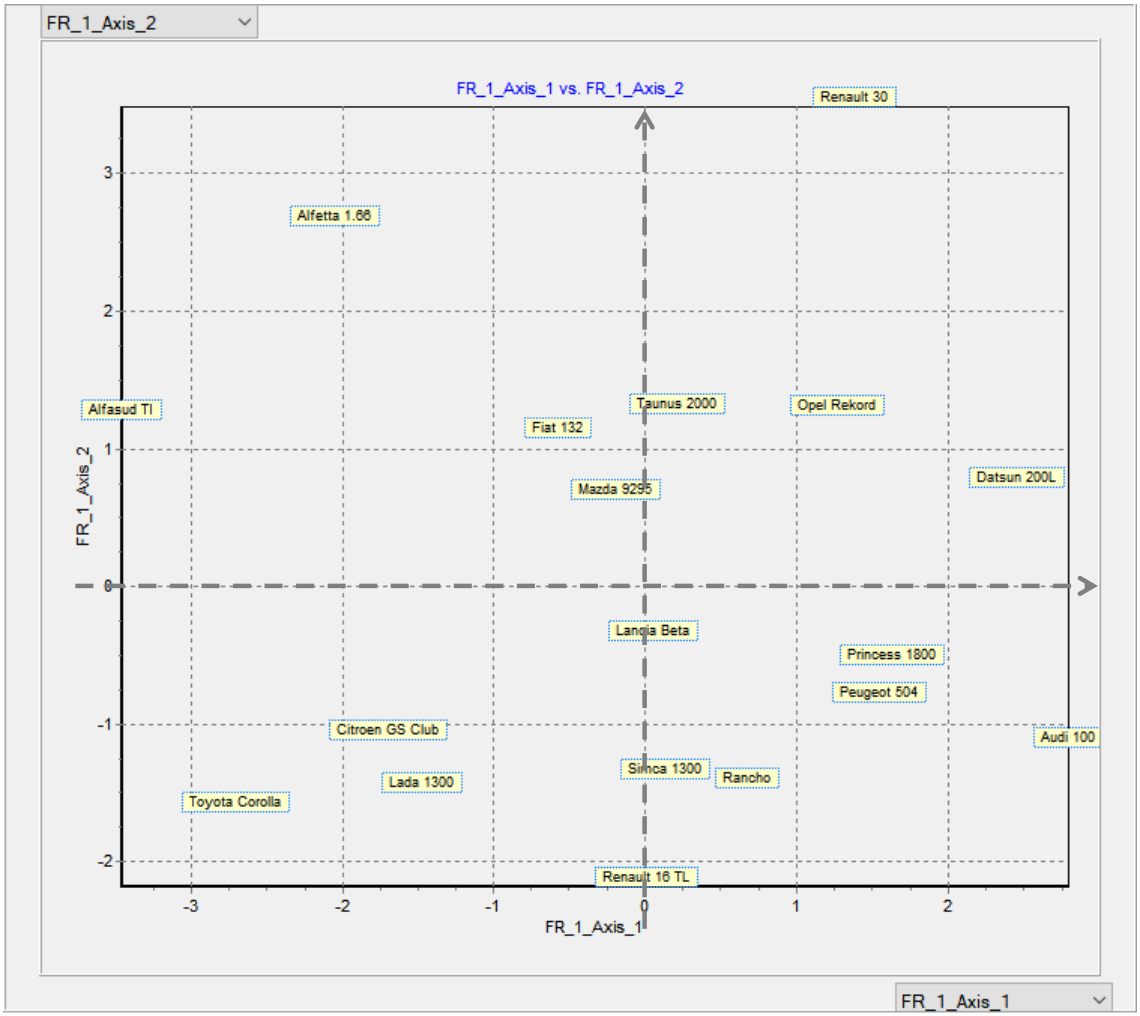


Figure 48 – Représentation dans le plan pivoté – Rotation VARIMAX – Données "Autos"

- La Renault 30 n'est pas si grande finalement, on peut le constater facilement en triant le tableau de données selon la longueur des véhicules. Elle se démarque clairement par ses performances en revanche.
- L'Alfasud TI est petite (composante F1), mais elle se situe au niveau des berlines toniques en termes de performances (composante F2).
- La Renault 16 TL est une berline moyenne, mais elle est réellement anémique.
- Etc.

On notera que les positions relatives (proximités entre les individus) n'ont pas été modifiées.

Intérêt de la rotation des facteurs. Très honnêtement, je ne comprends pas très bien pourquoi on n'insiste pas plus sur la rotation des facteurs dans la littérature francophone. Il faut l'utiliser à bon escient bien sûr. Mais n'oublions pas que l'objectif de l'ACP est de disposer d'une représentation en dimension réduite avec des composantes interprétables, la rotation y contribue. Elle peut même aider à déterminer le nombre de facteurs à retenir. Dans l'exemple « Autos », les différentes techniques (sections 1.3.1 et 2.1) semblaient militer pour le choix d'une seule composante. En fixant quand-même (q = 2) et en opérant une rotation VARIMAX, nous constatons que le deuxième facteur avait toute son importance dans la compréhension des relations sous-jacentes entre les variables et les individus.

Rotation QUARTIMAX. La rotation QUARTIMAX est une alternative orthogonale qui travaille sur les COS2. Elle cherche à maximiser la qualité de la représentation des variables sur le moins de facteurs possibles. On constate généralement qu'elle a tendance à produire un facteur associé à la majorité des variables. Ce qui va à l'encontre de notre objectif de clarifier l'interprétation.

Rotations obliques. Des techniques de pivotement qui ne préservent pas l'orthogonalité des facteurs existent, on parle de « rotation oblique » (ex. oblimin direct, rotation promax). Leur intérêt est questionnable. La perte de l'orthogonalité est un problème pour les représentations graphiques. Et si son maintien n'est pas vital pour l'analyse, autant se tourner vers les méthodes de clustering de variables (ex. section 2.5).

2.2.2 Rotation VARIMAX dans les logiciels

La rotation orthogonale des facteurs est proposée dans TANAGRA (TUTO 8). Ses sorties ont servi à illustrer la description de la méthode dans cette section. Elle est disponible dans d'autres logiciels, en tout premier lieu SAS PROC FACTOR qui m'a beaucoup aidé pour calibrer mon implémentation de la méthode VARIMAX dans TANAGRA. Sous R, nous faisons confiance à la librairie « psych » (TUTO 9, section 5.1).

2.2.2.1 SAS – Proc FACTOR

La PROC FACTOR englobe l'analyse en composantes principales et l'analyse en facteurs principaux sous SAS. Voici l'instruction utilisée pour le traitement des données « Autos » :

```
PROC FACTOR DATA = AUTOS
        METHOD = PRINCIPAL
        NFACTORS = 2
        VARDEF = N
        ROTATE = VARIMAX;
        VAR CYL PUISS LONG LARG POIDS VMAX;
RUN;
```

- METHOD indique la technique utilisée, « PRINCIPAL » pour l'analyse en composantes principales ;
- NFACTORS = 2 pour ne produire que 2 facteurs ;
- VARDEF = N pour utiliser (1/n) lors du calcul des écarts-type ;
- ROTATE = VARIMAX pour spécifier une rotation VARIMAX sur les (NFACTORS = 2) précédemment demandés.

Les sorties sont standards. Nous retiendrons en particulier :

- Avant l'opération de rotation : les tableaux des corrélations des variables avec les facteurs et les variances expliquées.

2 facteurs seront retenus par le critère NFACTOR.

Représentation du facteur			
		Factor1	Factor2
CYL	CYL	0.89346	0.11491
PUISS	PUISS	0.88686	0.38469
LONG	LONG	0.88615	-0.38103
LARG	LARG	0.81354	-0.41274
POIDS	POIDS	0.90519	-0.22453
VMAX	VMAX	0.75471	0.57352

Variance expliquée par chaque facteur	
Factor1	Factor2
4.4208581	0.8560623

Figure 49 – Proc Factor – Données "Autos" – Avant Rotation Varimax

- La matrice de rotation dans le plan qui réalise le changement de base.

La procédure FACTOR
Méthode de rotation : Varimax

Matrice de transformation orthogonale		
	1	2
1	0.75185	0.65934
2	-0.65934	0.75185

Figure 50 – Proc Factor – Matrice de rotation dans le plan – Données "Autos"

– Le tableau des corrélations après rotation et les variances restituées par les facteurs.

Caractéristique du facteur de rotation			
		Factor1	Factor2
CYL	CYL	0.59598	0.67549
PUISS	PUISS	0.41314	0.87397
LONG	LONG	0.91748	0.29780
LARG	LARG	0.88379	0.22608
POIDS	POIDS	0.82860	0.42801
VMAX	VMAX	0.18928	0.92881

Variance expliquée par chaque facteur	
Factor1	Factor2
2.8711381	2.4057822

Figure 51 – Proc Factor – Après Rotation Varimax – Données "Autos"

PROC FACTOR propose une liste impressionnante de techniques de rotation des facteurs.

2.2.2.2 R – Le package « psych »

La librairie « psych » est très intéressante pour l'analyse de données multivariée. Elle intègre l'analyse en composantes principale sans et avec plusieurs algorithmes de rotation.

Nous chargeons les données et nous réalisons l'ACP usuelle dans un premier temps.

```
library(openxlsx)
D <- read.xlsx("Data_Methodes_Factorielles.xlsx",sheet=1,rowNames=TRUE)
str(D)

## 'data.frame':    18 obs. of  6 variables:
##  $ CYL  : num  1350 1588 1294 1222 1585 ...
##  $ PUISS: num  79 85 68 59 98 82 79 55 128 55 ...
##  $ LONG : num  393 468 424 412 439 429 449 424 452 399 ...
##  $ LARG : num  161 177 168 161 164 169 169 163 173 157 ...
##  $ POIDS: num  870 1110 1050 930 1105 ...
##  $ VMAX : num  165 160 152 151 165 160 154 140 180 140 ...
```

```r
#Librairie 'psych'
library(psych)

#acp sans rotation
acp_simple <- principal(D,nfactors=2,rotate='none')
print(acp_simple$loadings)
```

```
##
## Loadings:
##       PC1    PC2
## CYL    0.893  0.115
## PUISS  0.887  0.385
## LONG   0.886 -0.381
## LARG   0.814 -0.413
## POIDS  0.905 -0.225
## VMAX   0.755  0.574
##
##                  PC1   PC2
## SS loadings    4.421 0.856
## Proportion Var 0.737 0.143
## Cumulative Var 0.737 0.879
```

Les résultats sont tout à fait conformes aux autres outils.

L'option (rotate) définit le type de rotation. Nous faisons le choix de « varimax », qui est l'option par défaut si nous ne la spécifions pas.

```r
#acp avec rotation varimax
acp_varimax <- principal(D,nfactors=2,rotate='varimax')
print(acp_varimax$loadings)
```

```
##
## Loadings:
##       RC1   RC2
## CYL   0.598 0.674
## PUISS 0.415 0.873
## LONG  0.918 0.295
## LARG  0.884 0.224
## POIDS 0.830 0.426
## VMAX  0.192 0.928
##
##                  RC1   RC2
## SS loadings    2.88 2.397
## Proportion Var 0.48 0.399
## Cumulative Var 0.48 0.879
```

Les résultats sont très proches de ceux de SAS et TANAGRA. N'oublions pas qu'il s'agit d'une heuristique et non pas d'un algorithme exact. Ces écarts très minimes ne sont en rien dérangeants.

2.3 Indices de compressibilité de l'information

On peut considérer l'ACP comme une compression de l'information. Elle n'est possible que si les données présentent une certaine redondance. Si les variables sont parfaitement corrélées, un seul axe factoriel suffit, il restituera 100% de l'information disponible. A l'inverse, si elles sont deux à deux orthogonales, a fortiori si elles sont deux à deux indépendantes, le nombre adéquat de facteurs à retenir est égal au nombre de variables. Dans ce dernier cas, la matrice de corrélation – impliquée dans le calcul de la solution – est la matrice unité (ou matrice identité). Il est illusoire d'espérer obtenir un résumé efficace en un nombre de facteurs réduit.

Remarque : Attention, si les variables sont deux à deux orthogonales, la compression avec l'ACP n'est pas satisfaisante, mais ça ne veut pas dire pour autant que les données ne sont pas porteuses d'information, l'inertie globale n'est pas nulle.

Une inspection rapide de la matrice des corrélations ou encore une représentation graphique de type « heatmap » des corrélations croisées (par exemple, Figure 7) peuvent être une première piste. Certaines références (TUTO 7, page 4) indiquent que lorsque les corrélations croisées sont toutes supérieures à 0.3 en valeur absolue, l'analyse en composantes principales est appropriée. Mais cette approche est très empirique. De plus, elle devient impraticable lorsque le nombre de variables augmente.

Le test de sphéricité de Bartlett est une première approche numérique pour analyser la compressibilité de l'information. Elle vérifie dans quelle mesure la matrice des corrélations observées s'écarte de la matrice unité. Comme nous l'avions indiqué plus haut (section 1.3.1.4), le test conclut quasiment toujours à un écartement très significatif, d'autant plus que les effectifs augmentent, et surtout, il ne nous donne pas d'indications sur le rôle des différentes variables qui composent la base de données.

Dans cette section, nous nous intéressons à l'indice KMO (Kaiser, Mayer, Olkin). On parle aussi d'indice MSA (Measure of Sampling Adequacy ; l'appellation n'est pas très heureuse, on ne voit pas très bien en quoi les données seraient adéquates ou pas...). Il participe de la même idée : est-ce qu'il est possible de trouver une factorisation efficace des données ?

Pour y répondre, il s'appuie sur une stratégie différente du test de Bartlett. Le point de départ est toujours la matrice de corrélation. On sait que les variables sont plus ou moins liées dans la base. La corrélation brute entre deux variables est influencée par les (p-2) autres. Nous utilisons la corrélation partielle pour mesurer la relation (nette) entre deux variables en retranchant l'influence des autres. L'indice cherche alors à confronter la corrélation brute avec la corrélation partielle. Si la seconde est nettement plus faible (en valeur absolue), cela veut dire que la liaison est effectivement déterminée par les autres variables. Cela accrédite l'idée de redondance, et donc la possibilité de mettre en place une réduction efficace de l'information. A contrario, si la seconde est équivalente, voire plus élevée, en valeur absolue, cela veut dire qu'il y a une relation directe entre les deux variables. Elle sera difficilement prise en compte par l'ACP. Dans les faits, ces deux variables détermineront souvent un axe factoriel à elles seules. C'est tout l'intérêt de l'indice KMO. Il permet d'apprécier en amont l'intérêt de l'ACP en termes de réduction de dimensionnalité, mais aussi les modalités selon lesquelles pèseront les différentes variables sur les résultats de l'analyse.

2.3.1 Matrice des corrélations partielles

La corrélation de deux variables en défalquant l'influence des (p-2) autres peut être obtenue de différentes manières (https://en.wikipedia.org/wiki/Partial_correlation) : via la formule récursive ; via la corrélation des résidus des variables suite à une régression sur les variables de contrôle ; via l'inversion de la matrice des corrélations brutes (R).

Cette dernière solution est la plus immédiate dans le cadre de l'ACP où (R) joue un rôle central dans l'analyse. Dans ce qui suit, la matrice R est de termes (r_{lj}) où $(l = 1, \dots, p\ ; j = 1, \dots, p)$; son inverse R^{-1} (v_{lj}) et la matrice des corrélations partielles P (ρ_{lj}).

Alors :

$$\rho_{lj} = -\frac{v_{lj}}{\sqrt{v_{ll} \times v_{jj}}}$$

Données « Autos ». Sur les données « Autos », nous calculons la matrice des corrélations après avoir chargé les données.

```
#chargement des données - index_col = 0 pour indiquer que la colonne n°0 est un la
bel
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="DATA_ACP_ACTIF",index_col=0)

#affichage des caractéristiques
print(D.info())

#nombre de variables
p = D.shape[1]

#nombre d'observations
n = D.shape[0]

#matrice des X
X = D.values
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18 entries, Alfasud TI to Lada 1300
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   CYL     18 non-null     int64
 1   PUISS   18 non-null     int64
 2   LONG    18 non-null     int64
 3   LARG    18 non-null     int64
 4   POIDS   18 non-null     int64
 5   VMAX    18 non-null     int64
dtypes: int64(6)
memory usage: 1008.0+ bytes
None
```

```
#Librairie numpy
import numpy

#matrice des corrélations
R = numpy.corrcoef(X,rowvar=False)
print(R)
```

```
[[1.         0.79662771 0.70146192 0.62975716 0.78895203 0.66493402]
 [0.79662771 1.         0.64136235 0.52083197 0.765293   0.84437948]
 [0.70146192 0.64136235 1.         0.84926635 0.86809028 0.47592847]
 [0.62975716 0.52083197 0.84926635 1.         0.71687392 0.47294527]
 [0.78895203 0.765293   0.86809028 0.71687392 1.         0.4775956 ]
 [0.66493402 0.84437948 0.47592847 0.47294527 0.4775956  1.        ]]
```

Le niveau global des corrélations brutes est élevé. Toutes les corrélations sont de surcroît positives. Relevons quelques résultats que nous approfondirons plus bas : r(PUISS, VMAX) = 0.844 ; r(POIDS, VMAX) = 0.477. Cette dernière corrélation paraît assez étrange, une relation positive entre le poids et la vitesse maximum ne correspond pas vraiment à ce que l'on sait des voitures, ou alors dans les descentes peut-être, et encore il faudrait que la pente soit très raide.

Nous inversons R puis nous formons la matrice des corrélations partielles. La diagonale de RHO est juste un artefact de calcul, il ne faut pas en tenir compte.

```
#inversion de la matrice
V = numpy.linalg.inv(R)
print(V)
```

```
[[ 3.77201352 -0.69347038  0.31247993 -0.43395472 -1.96161216 -0.92921187]
 [-0.69347038 11.1188198   0.74480091  2.28233732 -6.86128033 -7.08436656]
 [ 0.31247993  0.74480091  7.20419514 -3.19842205 -4.48616669 -0.61010269]
 [-0.43395472  2.28233732 -3.19842205  4.19760475 -0.82030424 -1.70985103]
 [-1.96161216 -6.86128033 -4.48616669 -0.82030424  9.95728271  4.86536606]
 [-0.92921187 -7.08436656 -0.61010269 -1.70985103  4.86536606  6.37511213]]
```

```
#matrice des corrélations partielles
RHO = numpy.zeros(shape=(p,p))

#double boucle
for l in range(p):
    for j in range(p):
        RHO[l,j] = -V[l,j]/numpy.sqrt(V[l,l]*V[j,j])

print(RHO)
```

```
[[-1.          0.10708089 -0.05994359  0.10905787  0.32007821  0.18948909]
 [ 0.10708089 -1.         -0.08321816 -0.33407941  0.65208679  0.84144893]
 [-0.05994359 -0.08321816 -1.          0.58162393  0.52967839  0.09002566]
 [ 0.10905787 -0.33407941  0.58162393 -1.          0.1268831   0.33053206]
 [ 0.32007821  0.65208679  0.52967839  0.1268831  -1.         -0.61066241]
 [ 0.18948909  0.84144893  0.09002566  0.33053206 -0.61066241 -1.        ]]
```

La corrélation r(PUISS, VMAX / (p-2) autres) = 0.841. Elle n'est pas impactée par le retrait de l'information apportée par les autres variables. Il y a une relation intrinsèque forte entre ces descripteurs. On le comprend aisément, plus une voiture est puissante, plus elle sera rapide.

r(POIDS, VMAX / (p-2) autres) = -0.610. Elle devient négative, plus en phase avec nos connaissances. L'apparente liaison positive mesurée par la corrélation brute était en réalité déterminée par les autres descripteurs. Nous en saurons davantage plus loin, lorsque nous étudierons l'influence relative des variables.

Pour disposer d'une vision synthétique des corrélations brutes et partielles, nous affichons côte-à-côte les graphiques « heatmap ». Encore une fois, il ne faut pas tenir compte de la diagonale des corrélations partielles ici.

```
#librairie graphique
import seaborn as sns
```

```
#heatmap pour identifier visuellement les corrélations fortes
sns.heatmap(R,xticklabels=D.columns,yticklabels=D.columns,vmin=-1,vmax=+1,center=0,cmap="RdBu",linewidths=0.5)

#heatmap pour identifier visuellement les corrélations partielles fortes
sns.heatmap(RHO,xticklabels=D.columns,yticklabels=D.columns,vmin=-1,vmax=+1,center=0,cmap="RdBu",linewidths=0.5)
```

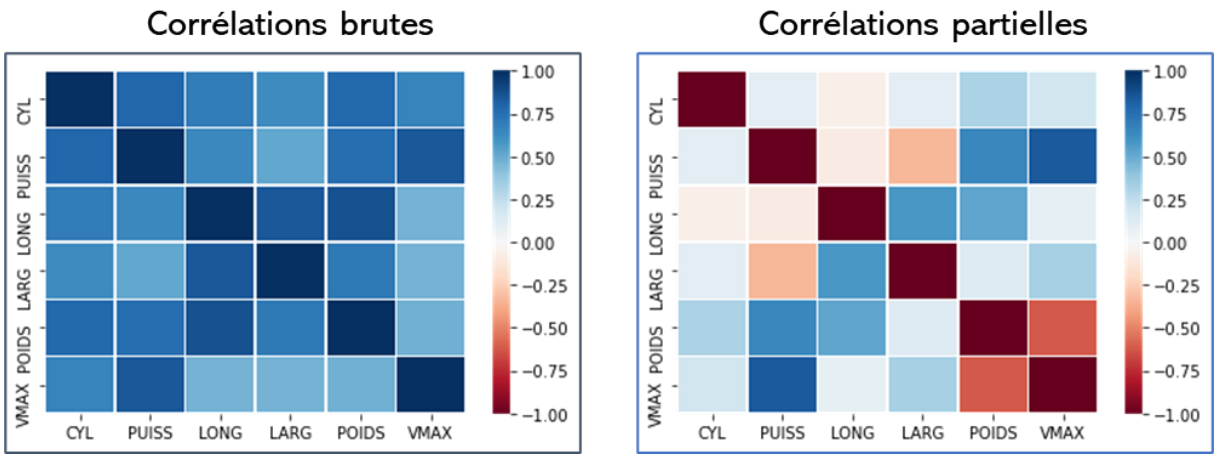


Figure 52 – Corrélations brutes et partielles – Données "Autos"

2.3.2 Indice KMO global

L'indice KMO global est basé sur l'opposition entre les corrélations brutes et partielles. Il est défini comme suit :

$$KMO = \frac{\sum_l \sum_{j \neq l} r_{lj}^2}{\sum_l \sum_{j \neq l} r_{lj}^2 + \sum_l \sum_{j \neq l} \rho_{lj}^2}$$

L'indice KMO varie entre 0 et 1. S'il est proche de 0, les corrélations partielles sont identiques aux corrélations brutes. Dans ce cas, une compression efficace n'est pas possible. Les variables sont deux à deux orthogonales. S'il est proche de 1, nous aurons un excellent résumé de l'information sur les premiers axes factoriels.

On nous donne parfois ici et là des grilles de lecture : « bon » lorsque supérieur à 0.8 ; « mauvais » en dessous de 0.5 (documentation SAS). Dans ce dernier cas, SAS conseille d'y remédier en supprimant les variables en cause ou en rajoutant d'autres descripteurs qui soient en lien avec elles pour préciser l'analyse. Pourquoi pas. N'oublions pas simplement que l'indice KMO cherche à mesurer la compressibilité de l'information. Une valeur élevée ne veut pas dire qu'on va tirer des résultats extraordinairement exploitables à partir des données.

Sous Python, les calculs sont très simples à partir des matrices des corrélations brutes et partielles. Nous n'oublions pas de retrancher la diagonale en effectuant les sommes.

```
#indice KMO global
KMO = (numpy.sum(R**2)-p)/((numpy.sum(R**2)-p) + (numpy.sum(RHO**2)-p))
print(KMO)

0.7400088239713861
```

(KMO = 0.74), il y a une redondance certaine dans les données. On ne s'en étonne pas au regard du niveau globalement élevé des corrélations brutes. Mais on ne connait pas à ce stade les variables qui en sont responsables, et celles qui ne suivent pas la tendance générale.

2.3.3 Indice KMO par variable

L'indice KMO d'une variable se concentre sur le différentiel des corrélations brutes et partielles le concernant :

$$KMO_j = \frac{\sum_{l \neq j} r_{lj}^2}{\sum_{l \neq j} r_{lj}^2 + \sum_{l \neq j} \rho_{lj}^2}$$

Python fait merveille dans ce contexte, une ligne de commande suffit pour obtenir les $\mathrm{KMO_j}$.

```
#indice KMO par variables
KMO_j = (numpy.sum(R**2,axis=0)-1)/((numpy.sum(R**2,axis=0)-1)+(numpy.sum(RHO**2,axis=0)-1))

#affichage
print(pandas.DataFrame(KMO_j,index=D.columns))

              0
CYL    0.939956
PUISS  0.674346
LONG   0.803384
LARG   0.783650
POIDS  0.693091
VMAX   0.597664
```

Les variables sont globalement fortement liées, ce n'est pas une surprise. CYL (0.9399) pèse sur l'ensemble des descripteurs. VMAX (0.5976) fait un peu bande à part, c'est pour cette raison qu'elle est plus déterminante sur le second facteur.

2.3.4 Indice KMO dans les logiciels

Les indices KMO (MSA) sont peu présents dans les références françaises. Il en est de même des outils de l'école française de l'analyse des données. Il faut regarder du côté des logiciels anglo-saxons pour y avoir accès. C'est en lisant la documentation de SAS et SPSS que j'ai pu en saisir la portée. Je l'ai par la suite intégrée dans TANAGRA (TUTO 7).

Nous utilisons SAS dans cette section avec la PROC FACTOR. Voici l'instruction utilisée pour le traitement des données « Autos » :

```
PROC FACTOR DATA = AUTOS
        METHOD = PRINCIPAL
        NFACTORS = 2
        VARDEF = N
        MSA;
        VAR CYL PUISS LONG LARG POIDS VMAX;
RUN;
```

- METHOD indique la technique utilisée, « PRINCIPAL » pour l'analyse en composantes principales ;
- NFACTORS = 2 pour ne produire que 2 facteurs ;
- VARDEF = N pour utiliser (1/n) lors du calcul des écarts-type ;
- MSA pour inclure les indicateurs KMO dans les résultats.

Nous disposons alors de 2 tableaux supplémentaires :

- La matrice des corrélations partielles (Figure 53) ;

La procédure FACTOR
Méthode de facteur initiale : Composantes principales

Corrélations partielles contrôlant toutes les autres variables

		CYL	PUISS	LONG	LARG	POIDS	VMAX
CYL	CYL	1.00000	0.10708	-0.05994	0.10906	0.32008	0.18949
PUISS	PUISS	0.10708	1.00000	-0.08322	-0.33408	0.65209	0.84145
LONG	LONG	-0.05994	-0.08322	1.00000	0.58162	0.52968	0.09003
LARG	LARG	0.10906	-0.33408	0.58162	1.00000	0.12688	0.33053
POIDS	POIDS	0.32008	0.65209	0.52968	0.12688	1.00000	-0.61066
VMAX	VMAX	0.18949	0.84145	0.09003	0.33053	-0.61066	1.00000

Figure 53 – Matrices des corrélations partielles – SAS PROC FACTOR – Données "Autos"

– Les indices KMO globaux et par variable (Figure 54)

Mesure d'adéquation de l'échantillonnage de Kaiser : MSA globale = 0.74000882					
CYL	PUISS	LONG	LARG	POIDS	VMAX
0.93995630	0.67434615	0.80338422	0.78364969	0.69309077	0.59766411

Figure 54 – Indices KMO globaux et par variables – Données "Autos"

Nous retrouvons exactement les résultats mis en avant sous Python.

2.4 ACP sur corrélations partielles

Dans certaines situations, l'analyse en composantes principales propose des résultats guère décisifs, parce qu'évidents. C'est le cas lorsque l'étude est dominée par l'influence de quelques variables qui pèsent exagérément sur toutes les autres. On parle « d'effet taille » ou « facteur taille » (Saporta, 2006, section 7.3.3.3). Il est alors généralement conseillé d'ignorer la première composante pour se concentrer sur l'étude des suivantes qui différencient les individus de « taille » semblable (le second facteur est appelé « facteur de forme »). Mais ce n'est pas aussi simple car nous sommes alors confrontés à d'autres problèmes. Par exemple, les guides usuels (règle de Kaiser, scree plot, etc) pour la détection du nombre adéquat de facteurs deviennent inopérants. En effet, mis à part le premier, les axes sont portés par des valeurs propres très faibles, laissant à penser qu'ils correspondent à une information résiduelle, négligeable. Bien malin est celui qui saurait dès lors déterminer les facteurs intéressant pour l'interprétation.

Nous avons vu une première solution dans ce chapitre avec la rotation orthogonale (section 2.2). En dispatchant mieux l'information sur les facteurs, nous mettons au jour les informations noyées. C'était le cas du facteur « performance » dans les données « Autos », que l'on percevait confusément, mais qui apparaît de manière évidente après la rotation VARIMAX.

L'indice KMO est une second piste parce qu'elle permet de cerner l'influence relative des variables dans l'analyse (section 2.3). Dans le cas des données « Autos », nous constatons que la variable « CYL » impacte toutes les autres ($\mathrm{KMO}_{\mathrm{CYL}} = 0.9399$). Ce qui se conçoit très bien. Dans notre pool de véhicules des années 70, il n'y pas de diesel, il n'y a pas de turbo, la

différenciation des gammes repose exclusivement sur la cylindrée. Ici, les connaissances du domaine et le calcul numérique se rejoignent, ce qui est toujours rassurant.

Dans cette section, nous nous intéressons à l'analyse en composantes principales sur corrélations partielles. L'objectif est d'annihiler le rôle prééminent (néfaste) d'une ou plusieurs variables qui faussent la perception de l'information véhiculée par les données, permettant ainsi d'affiner les résultats et de mieux comprendre les relations sous-jacentes entre les variables restantes. Nous travaillons toujours sur les données « Autos » pour conserver une certaine cohérence dans notre progression pédagogique. Une étude plus réaliste a été menée sur des données comportant un plus grand nombre de variables. Il s'agissait d'étudier la circonférence des différentes parties du corps humain en retranchant l'effet patent de la taille, du poids et du genre (TUTO 10).

2.4.1 Corrélation partielle

La notion de corrélation partielle a été avancée précédemment (section 2.3.1). Nous l'approfondissons dans cette section.

Le coefficient de corrélation mesure l'intensité de la liaison linéaire entre deux variables. Il n'est pas rare qu'il soit faussé par une ou d'autres caractères, laissant à penser à tort l'existence ou l'absence d'une relation. On parle de facteurs confondants. La littérature regorge d'exemples de corrélations numériquement élevées mais qui ne résistent pas à l'interprétation (Rakotomalala R., « Analyse de corrélation », version 1.1 », mars 2015, chapitre 5) : corrélation positive entre les ventes de lunettes noires et de cornets de glaces, influencé en réalité par l'ensoleillement ; corrélation entre le prix de vente des véhicules et leur consommation en carburant, influencé par la gamme ; corrélation entre la longueur des cheveux et la taille des personnes, influencé par le genre ; etc. L'idée de la corrélation partielle d'ordre 1 est de mesurer le degré de liaison entre deux variables (X_1) et (X_2) en contrôlant l'influence d'une tierce variable (Y). Elle est définie comme suit :

$$r_{x_1,x_2/y} = \frac{r_{x_1,x_2} - r_{x_1,y} \times r_{x_2,y}}{\sqrt{1 - r_{x_1,y}^2} \times \sqrt{1 - r_{x_2,y}^2}}$$

La formule symbolise parfaitement le concept sous-jacent. Au numérateur, leurs liaisons avec la variable modératrice (Y) sont retranchées de la relation brute entre (X_1, X_2). Le dénominateur est avant tout un terme de normalisation pour que la corrélation partielle varie entre [-1 ; +1].

Remarque : Il est possible de calculer les corrélations partielles d'ordre supérieur en utilisant une formule de récurrence ou en passant par les résidus de la régression (voir « Analyse de corrélation », section 5.3).

Dans le cadre de l'analyse en composantes principales, neutraliser l'influence d'une ou plusieurs variables qui pèsent sur l'étude permet de juguler le fameux « effet taille » qui fausse la première composante. Le dessein est de ramener les individus dans un référentiel commun qui les rend directement comparables. Pour reprendre l'exemple des données « Autos », nous avons identifié le rôle central de CYL. En retranchant son influence dans le calcul des corrélations croisées, nous menons une étude « à cylindrée égale » c.-à-d. nous comparerons l'encombrement en considérant que tous les véhicules ont la même cylindrée, nous ferons de même pour l'analyse de la performance.

Dans ce qui suit, nous calculons les corrélations brutes entre les variables hors CYL des données « Autos », puis les corrélations partiellement à CYL. Une première comparaison nous donnera des premières indications sur la nature des relations entre les variables.

```
#chargement - index_col = 0 pour indiquer que la colonne n°0 est un label
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="DATA_ACP_ACTIF",index_col=0)

#affichage des caractéristiques
print(D.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18 entries, Alfasud TI to Lada 1300
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   CYL     18 non-null     int64
 1   PUISS   18 non-null     int64
 2   LONG    18 non-null     int64
 3   LARG    18 non-null     int64
 4   POIDS   18 non-null     int64
 5   VMAX    18 non-null     int64
dtypes: int64(6)
memory usage: 1008.0+ bytes
None
```

```
#nombre d'observations - n = 18 observations
n = D.shape[0]

#matrice des X sans la variable CYL
X = D.iloc[:,1:].values
print(X)
```

```
[[  79  393  161  870  165]
 [  85  468  177 1110  160]
 [  68  424  168 1050  152]
 [  59  412  161  930  151]
 [  98  439  164 1105  165]
 [  82  429  169 1080  160]
 [  79  449  169 1160  154]
 [  55  424  163 1010  140]
 [ 128  452  173 1320  180]
 [  55  399  157  815  140]
 [ 109  428  162 1060  175]
 [  82  445  172 1160  158]
 [ 115  469  169 1370  160]
 [  98  438  170 1080  167]
 [  80  431  166 1129  144]
 [  83  440  165 1095  165]
 [ 100  459  173 1120  173]
 [  68  404  161  955  140]]
```

```
#nombre de variables -- p = 5 maintenant
p = X.shape[1]
```

Nous calculons la matrice des corrélations brutes à partir de X.

```
#librairie numpy
import numpy

#matrice des correlations brutes
R = numpy.corrcoef(X,rowvar=False)
print(R)
```

```
[[1.         0.64136235 0.52083197 0.765293   0.84437948]
 [0.64136235 1.         0.84926635 0.86809028 0.47592847]
 [0.52083197 0.84926635 1.         0.71687392 0.47294527]
 [0.765293   0.86809028 0.71687392 1.         0.4775956 ]
 [0.84437948 0.47592847 0.47294527 0.4775956  1.        ]]
```

Deux stratégies sont possibles pour calculer les corrélations partielles. La première consiste à utiliser la formule ci-dessus. La seconde, que nous allons utiliser parce qu'elle est plus générique et nous permettra d'enchaîner avec l'ACP, consiste à calculer les résidus des régressions des descripteurs avec la variable modératrice. Explicitons cela.

Nous souhaitons calculer la corrélation partielle $r(X_1, X_2 / CYL)$. Nous formons les résidus de la régression de (X1 sur CYL) et (X2 sur CYL)

$$\begin{cases} e_1 = x_1 - (a_0 + a_1\, CYL) \\ e_2 = x_2 - (b_0 + b_1\, CYL) \end{cases}$$

La corrélation brute $r(e_1, e_2) = r(X_1, X_2 \,/\, CYL)$

L'intérêt est double :

- Généraliser l'approche à plusieurs variables modératrices ne pose aucune difficulté.
- Présenter la matrice des résidus $E = (e_1 \mid e_2 \mid \ldots)$ à un programme standard d'ACP permet de réaliser l'analyse en composantes principales sur les corrélations partielles.

Pour les données « Autos », nous formons la matrice des résidus, puis nous calculons les corrélations partielles.

```
#variable de contrôle
y = D.CYL.values

#librairie numpy
import numpy

#calculer le résidu de la régression de chaque variable avec CYL
E = numpy.apply_along_axis(arr=X,axis=0,func1d=lambda x:x-numpy.polyval(numpy.polyfit(y,x,1),y))
print(E.shape)
```

```
(18, 5)
```

Les corrélations brutes sur E correspondent aux corrélations partielles.

```
#matrice des corrélations partielles vs. CYL
RPART = numpy.corrcoef(E,rowvar=False)
print(RPART)
```

```
[[ 1.          0.19163511  0.04078385  0.36829487  0.69698446]
 [ 0.19163511  1.          0.736086    0.71854683  0.01785118]
 [ 0.04078385  0.736086    1.          0.46097645  0.09341514]
 [ 0.36829487  0.71854683  0.46097645  1.         -0.1024223 ]
 [ 0.69698446  0.01785118  0.09341514 -0.1024223   1.        ]]
```

Un graphique « heatmap » permet d'identifier ce qui distingue les deux matrices.

```
#librairie graphique
import seaborn as sns

#heatmap pour identifier visuellement les corrélations fortes
sns.heatmap(R,xticklabels=D.columns[1:],yticklabels=D.columns[1:],vmin=-1,vmax=+1,
center=0,cmap="RdBu",linewidths=0.5)
```

```
#heatmap pour identifier visuellement les corrélations partielles fortes
sns.heatmap(RPART,xticklabels=D.columns[1:],yticklabels=D.columns[1:],vmin=-1,vmax
=+1,center=0,cmap="RdBu",linewidths=0.5)
```

$r(X_1,X_2)$ $r(X_1,X_2/CYL)$

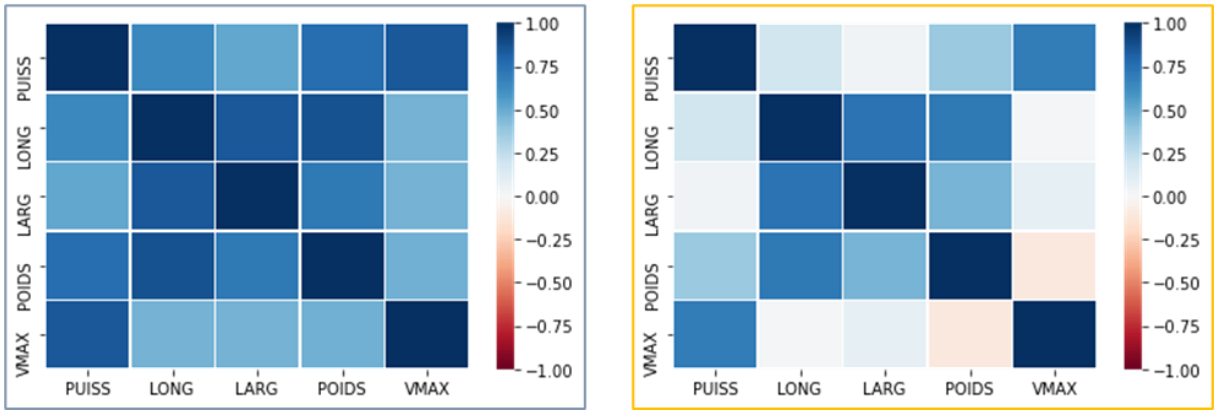


Figure 55 – Corrélations brutes et partielles relativement à CYL – Données "Autos"

Nous avions entraperçu ces résultats précédemment (section 2.3.1). Mais ici, « cylindrée » est la seule variable de contrôle. Nous observons par exemple :

- A cylindrée égale, la liaison entre le POIDS et la VMAX est plutôt négative. Oui, l'embonpoint n'aide pas à aller vite. La relation brute était positive parce que les voitures lourdes possédaient un moteur conséquent qui les rendaient rapides.
- PUISS et VMAX restent intrinsèquement liés. Il faut de la puissance pour être véloce, indépendamment de la cylindrée.

Dans la section suivante, nous lançons l'ACP sur les résidus E. Il faut bien garder à l'esprit que la lecture des résultats se fera toujours « à cylindrée égale ».

2.4.2 ACP sur corrélations partielles

Nous faisons appel au package « fanalysis » pour réaliser l'ACP.

```
#ACP sur les résidus de la régression sur CYL
from fanalysis.pca import PCA
acp = PCA(std_unit=True,row_labels=D.index,col_labels=D.columns[1:])
acp.fit(E)

#les valeurs propres
print(acp.eig_[0])

[2.40105829 1.61317102 0.69868647 0.18349256 0.10359165]
```

Première information importante, deux facteurs sont distinctement perceptibles maintenant, ils restituent 80% de l'information disponible. Nous affichons le diagramme des valeurs propres.

```
#librairie graphique
import matplotlib.pyplot as plt
#éboulis des v.p.
fig,ax=plt.subplots(figsize=(5,5))
ax.plot(range(1,p+1),acp.eig_[0],".-")
ax.set_xlabel("Nb. facteurs")
ax.set_ylabel("Val. propres")
plt.title("Eboulis des v.p.")
#seuil de Kaiser
ax.plot([1,p],[1,1],"r--",linewidth=1)
plt.show()
```

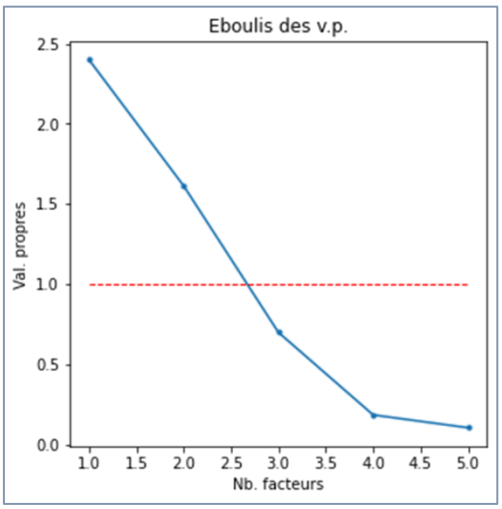


Figure 56 – Eboulis des v.p. – ACP sur corrélations partielles

Puis le cercle des corrélations.

```
#cercle des corrélations
acp.correlation_circle(num_x_axis=1,num_y_axis=2,figsize=(5,5))
```

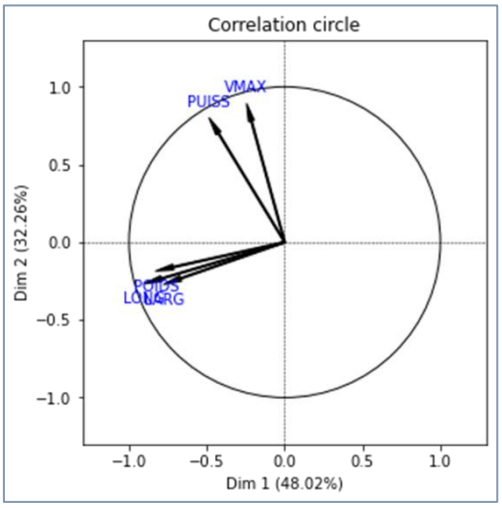


Figure 57 – Cercle des corrélations – ACP sur corrélations partielles

Les deux dimensions que recèlent les données sont clairement perceptibles contrairement à l'ACP sur les corrélations brutes (section 1.3.2). A cylindrée égale : l'encombrement et la performance distinguent les véhicules.

Les positions des observations dans le plan factoriel vont nous réserver des surprises.

```
#représentation des individus
fig,ax = plt.subplots(figsize=(10,10))
ax.axis([-3.5,+3.5,-3.5,+3.5])
ax.plot([-3.5,+3.5],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-3.5,+3.5],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Carte des individus")

for i in range(n):
    ax.text(acp.row_coord_[i,0],acp.row_coord_[i,1],D.index[i],color='navy')

plt.show()
```

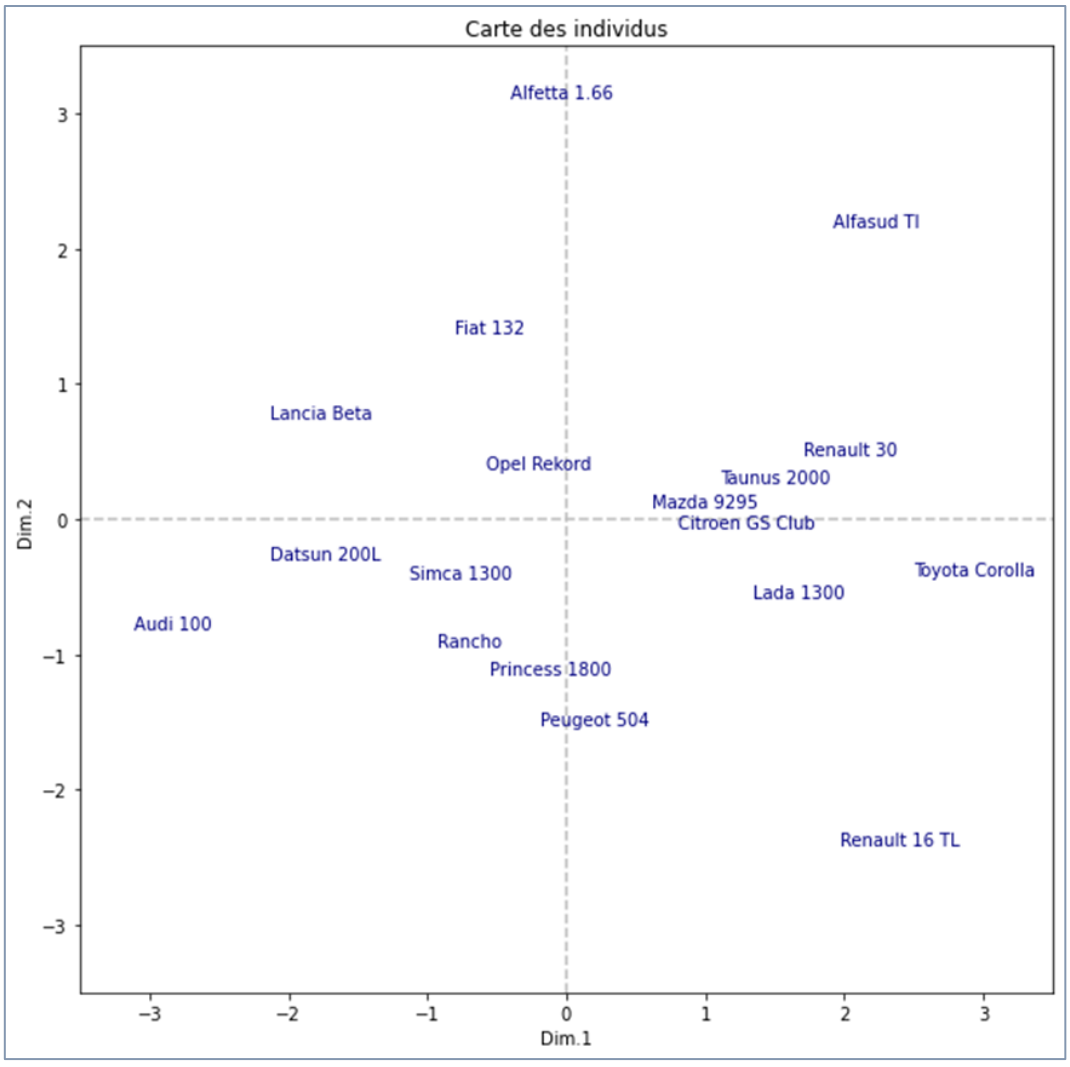


Figure 58 – Carte des individus – ACP sur corrélations partielles

Nous constatons entres autres que :

- La position relative de la Renault 30 a radicalement changé. Elle n'est pas si encombrante, elle serait même positionnée dans le voisinage de la Toyota Corolla. Ça paraît extraordinaire de prime abord. Mais en creusant un peu, on se rend compte qu'en ramenant toutes les valeurs à cylindrée égale (en divisant toutes les variables par la cylindrée par ex.), elle ferait même partie des « petits » véhicules. Et ses performances ne sont pas très brillantes, encore une fois eu égard à sa cylindre.
- L'Alfetta 1.66 se démarque réellement par ses performances. Elle se distinguait déjà précédemment (Figure 13), mais si l'on tient compte de sa cylindrée, sa puissance est réellement remarquable (109 ch pour une cylindrée de 1570 cm3 quand ses homologues développent entre 55 et 98 ch).
- La Renault 16 TL est réellement anémique (55 ch) pour son moteur de 1565 cm^3. Mais bon, si on connaît un peu l'histoire, il s'agissait d'une version bas de gamme volontairement bridée qui roulait à l'essence ordinaire (argh...) avec un carburateur Solex. Le même moteur s'est révélé largement plus volontaire sur la Renault 12 Gordini ou l'Alpine Renault A 110 par exemple (je dis ça au cas où l'on croirait que je fais une fixation négative sur les Renault).
- L'Alfasud TI est définitivement une petite teigneuse.

De manière générale, faire le parallèle entre les cartes des individus des ACP sur corrélations brutes et partielles se révèle très instructif.

2.4.3 ACP sur corrélations partielles dans les logiciels

En réalité, tout programme d'ACP peut réaliser une analyse sur corrélations partielles si nous lui passons la matrice des résidus défalqués de l'influence des variables de contrôle. Une seule ligne de code nous a suffi pour construire cette dernière sous Python. Je montre également comment réaliser la manipulation sous TANAGRA lorsque nous souhaitons introduire plusieurs variables de contrôle (TUTO 10). Dans le même tutoriel est décrite une étude sous R, toujours avec le même principe.

A ma connaissance, SAS est un des rares à la proposer nativement, tant dans la PROC FACTOR (TUTO 10, section 7) que dans la PROC PRINCOMP. L'étude attentive de leur documentation et de leurs sorties m'ont beaucoup éclairé sur les tenants et aboutissants de la méthode.

Avec PRINCOMP, nous exécutons les instructions suivantes sur les données « Autos » :

```
PROC PRINCOMP DATA=AUTOS VARDEF=N PLOTS=ALL;
        VAR PUISS LONG LARG POIDS VMAX;
        PARTIAL CYL;
        ID MODELE;
RUN;
```

L'option **PARTIAL** est celle qui nous intéresse. PRINCOMP produit notamment :

- La matrice des corrélations brutes,

Matrice de corrélation

		PUISS	LONG	LARG	POIDS	VMAX	CYL
PUISS	PUISS	1.0000	0.6414	0.5208	0.7653	0.8444	0.7966
LONG	LONG	0.6414	1.0000	0.8493	0.8681	0.4759	0.7015
LARG	LARG	0.5208	0.8493	1.0000	0.7169	0.4729	0.6298
POIDS	POIDS	0.7653	0.8681	0.7169	1.0000	0.4776	0.7890
VMAX	VMAX	0.8444	0.4759	0.4729	0.4776	1.0000	0.6649
CYL	CYL	0.7966	0.7015	0.6298	0.7890	0.6649	1.0000

- Les statistiques de la régression de chaque variable avec CYL, le coefficient de détermination R^2 et la racine carrée de l'erreur quadratique moyenne (RMSE),

Statistiques de régression

	PUISS	LONG	LARG	POIDS	VMAX
R carré	0.6346157133	0.4920488255	0.3965940822	0.6224453031	0.4421372471
RMSE	11.969831966	15.312145385	4.0113369935	81.783358912	8.812198339

- Les coefficients standardisés de la régression des variables sur CYL, sur les variables centrées et réduites donc, raison pour laquelle nous n'avons pas de constante,

Coef. de régression normalisés

	PUISS	LONG	LARG	POIDS	VMAX
CYL	0.7966277131	0.7014619202	0.6297571613	0.7889520284	0.6649340171

- La matrice des corrélations partielles,

Matrice de corrélation partielle						
		PUISS	LONG	LARG	POIDS	VMAX
PUISS	PUISS	1.0000	0.1916	0.0408	0.3683	0.6970
LONG	LONG	0.1916	1.0000	0.7361	0.7185	0.0179
LARG	LARG	0.0408	0.7361	1.0000	0.4610	0.0934
POIDS	POIDS	0.3683	0.7185	0.4610	1.0000	-.1024
VMAX	VMAX	0.6970	0.0179	0.0934	-.1024	1.0000

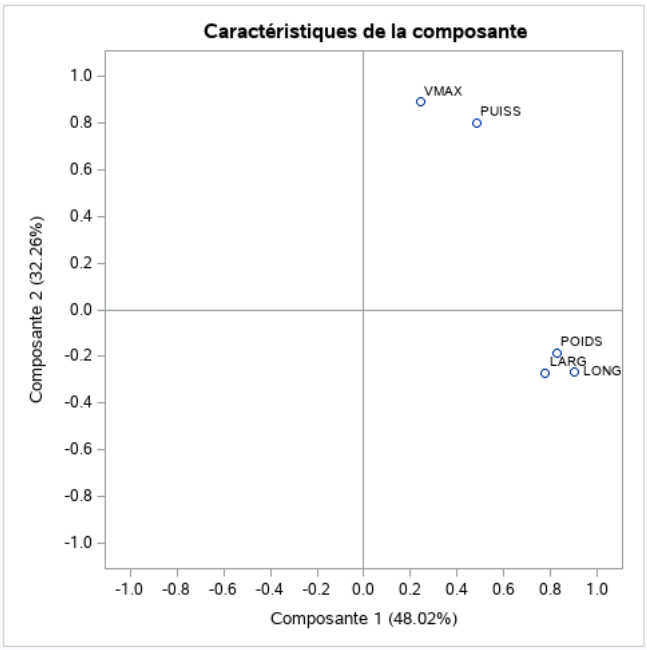
– Les valeurs propres issues de l'ACP sur corrélations partielles,

Valeurs propres de la matrice de corrélation partielle				
	Valeur propre	Différence	Proportion	Cumulé
1	2.40105829	0.78788727	0.4802	0.4802
2	1.61317102	0.91448455	0.3226	0.8028
3	0.69868647	0.51519391	0.1397	0.9426
4	0.18349256	0.07990092	0.0367	0.9793
5	0.10359165		0.0207	1.0000

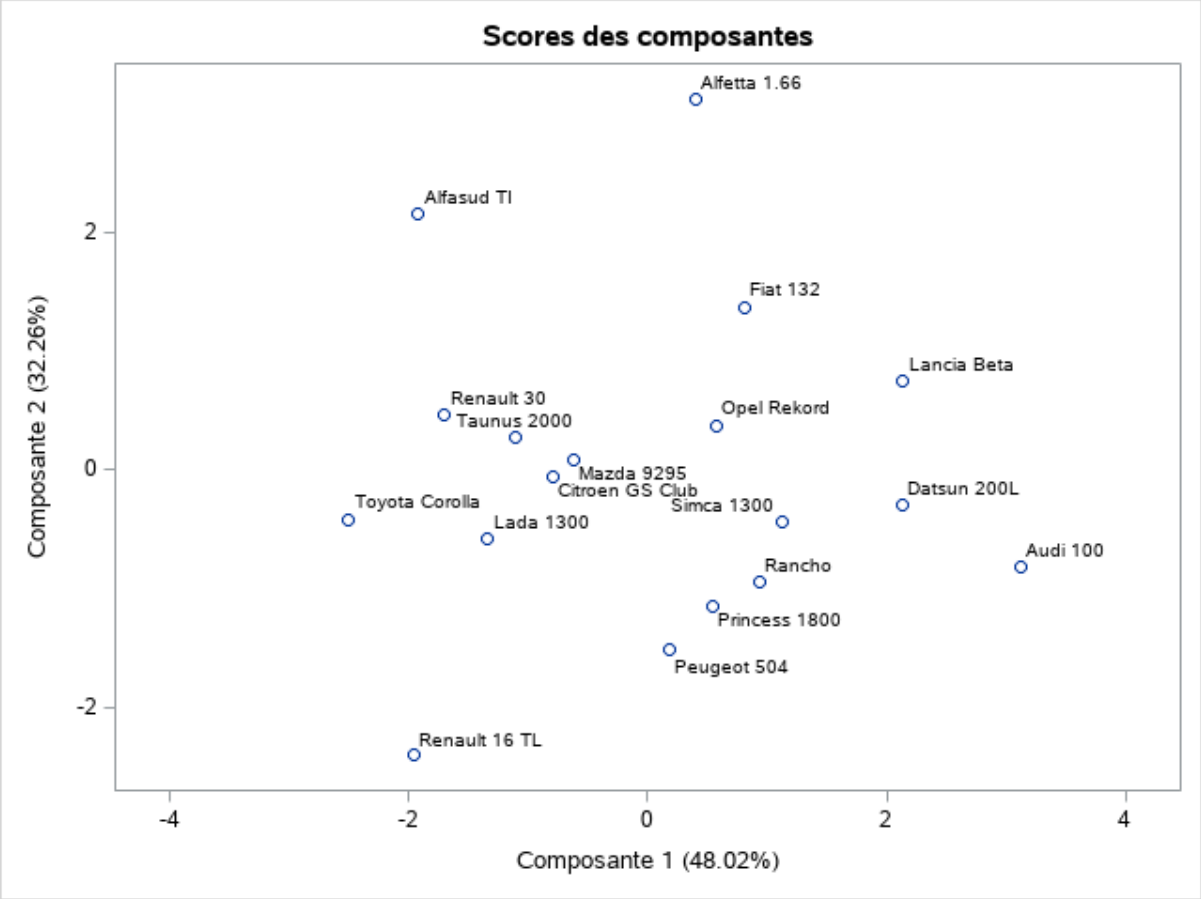
– Les vecteurs propres,

Vecteurs propres						
		Prin1	Prin2	Prin3	Prin4	Prin5
PUISS	PUISS	0.313261	0.629746	-.342052	0.182354	-.595848
LONG	LONG	0.582638	-.210391	0.081886	-.756140	-.194462
LARG	LARG	0.501658	-.213622	0.615081	0.550326	-.146703
POIDS	POIDS	0.534691	-.146082	-.583082	0.251071	0.538278
VMAX	VMAX	0.157622	0.701556	0.397474	-.170597	0.543953

– Et parmi les sorties graphiques, le « cercle » des corrélations dans le premier plan factoriel,



– La carte des individus dans le même espace de représentation.



Les résultats sont en tous points identiques à l'étude que nous avions mené sous Python dans la section précédente. Les intitulés de certains tableaux dans les sorties sont édifiants : « Statistiques de Régression », « Coefficients de régression normalisés ». PRINCOMP procède au calcul des composantes principales à partir des résidus de la régression, il est possible d'introduire plusieurs variables de contrôle par ce biais.

2.5 Clustering de variables – VARCLUS

2.5.1 Principe et intérêt de la classification de variables

Classification de variables. Dans la majorité des ouvrages, la classification de variables est décrite très sommairement. Les auteurs se contentent le plus souvent de la présenter comme un cas particulier de la typologie où le coefficient de corrélation r est utilisé pour mesurer la

proximité entre les variables, $(1-r)$ étant alors un indice de dissimilarité naturel [ou $(1-|r|)$, $(1-r^2)$, si l'on ne veut pas tenir compte du sens de la relation].

Pourtant, la classification de variables peut être très utile dans la recherche des structures sous-jacentes dans les données. Elle permet de repérer les groupes de variables redondantes, emmenant le même type d'information ; de distinguer les groupes de variables rapportant des informations complémentaires. Nous disposons ainsi de précieuses indications sur les principales « dimensions » qui composent les données.

Cette méthode peut également être mise en œuvre dans une stratégie de réduction/sélection des variables. Pour chaque groupe de variables, une variable « moyenne » unique pourra être produite et utilisée dans les analyses ultérieures, en tant que variable prédictive présentée aux méthodes supervisées par exemple, réduisant considérablement la dimensionnalité.

Classification de variables autour de composantes latentes. Cette approche repose sur l'idée de représenter chaque groupe de variables par le premier facteur de l'analyse en composantes principales (Vigneau E., Qannari E. M., « Clustering of Variables Around Latent Components », in Communication in Statistics-Simulation and Computation, 32(4), p. 1131-1150, 2003) . Il joue le rôle de « moyenne » avec une définition que nous comprenons parfaitement : il est élaboré de manière à maximiser la somme des carrés des corrélations avec les variables (section 1.1.3.3). Intellectuellement, c'est très satisfaisant. J'ai personnellement toujours eu des réticences à implémenter la typologie de variables parce que j'avais du mal à intégrer ce que représentait une variable synthétique qui serait une moyenne transversale non pondérée des descripteurs composant le groupe. Au moins, nous savons interpréter un axe factoriel. Nous disposons de plus d'un indicateur de qualité de restitution avec la valeur propre. La lecture des résultats est autrement plus intuitive.

Quel rapport avec l'ACP ? D'une certaine manière l'ACP peut être vue comme une procédure de classification floue (fuzzy clustering) des variables où : (1) les clusters sont représentés par les facteurs, avec la contrainte d'orthogonalité ; (2) l'appartenance des variables aux groupes est nuancée, le COS2 peut servir d'indicateur pour mesurer le degré d'adhésion. A l'inverse, avec la classification des variables, (a) nous n'avons plus la contrainte d'orthogonalité des

« dimensions », on peut la voir comme une ACP oblique, et (b) nous obtenons un partitionnement c.-à-d. chaque variable est assignée à un seul groupe.

De fait, l'analyse en composantes principales et la classification de variables en général correspondent à deux manières différentes d'appréhender le même problème qu'est la recherche des structures sous-jacentes aux données. Le lien est d'autant plus patent que, concernant la méthode VARCLUS que nous présenterons dans la section suivante, l'algorithme utilise explicitement et répétitivement l'analyse en composantes principales pour construire la partition des variables.

2.5.2 L'algorithme VARCLUS

VARCLUS est une méthode descendante, divisive et récursive, de classification automatique. Nous disposons d'une hiérarchie de partitions.

Pour subdiviser un groupe de variables à chaque étape, l'algorithme calcule les deux premiers facteurs de l'ACP, les fait pivoter pour maximiser l'alignement des variables avec les axes. La PROC VARCLUS de SAS s'appuie sur la rotation orthogonale QUARTIMAX. J'ai préféré utiliser la rotation VARIMAX - qui me paraît plus efficace - dans TANAGRA (TUTO 11). Si la valeur propre associée au second axe est supérieure à 1 (paramètre modifiable), les variables sont subdivisées en 2 sous-ensembles selon l'axe qui leur est le plus proche au sens du COS2 (au sens du carré de la corrélation). Après chaque subdivision, SAS procède à une réaffectation des variables pour maximiser la variance expliquée.

L'algorithme traite ensuite récursivement, indépendamment l'une de l'autre, les deux sous-groupes de variables. La processus s'arrête lorsque la seconde valeur propre de l'ACP est inférieure à 1 lors de la tentative de subdivision d'un groupe.

A l'issue du partitionnement, chaque cluster est représenté par la première composante de l'ACP composée sur le sous-ensemble de variables qui le constitue. La proportion de variance restituée correspond à sa valeur propre. Les composantes des clusters ne sont pas orthogonales.

Les logiciels affichent les résultats sous la forme d'un dendrogramme. Dans SAS, les réaffectations à chaque étape assurent de ne pas avoir d'inversions dans la hiérarchie indicée. Dans TANAGRA, il matérialise avant tout la séquence des subdivisions (TUTO 11).

2.5.3 Détail des calculs avec la PROC VARCLUS de SAS

La PROC VARCLUS comprend plusieurs options par rapport à l'algorithme générique : le seuil sur la seconde valeur propre est modulable, il peut être défini également sur la variance restituée du 2nd facteur, la moyenne non-pondérée (centroïde) peut faire office de variable représentative des clusters, etc. Nous nous en tenons aux paramètres par défaut dans cette section.

Nous utilisons la base « Burger King » pour décrypter le processus (section 1.7.1). Rappelons qu'elle décrit la composition de (n = 17) produits de l'enseigne à l'aide de (p = 10) variables actives. L'objectif est de créer une partition des variables de manière à mettre en évidence les principales caractéristiques des produits.

Voici la commande sous SAS :

```
PROC VARCLUS DATA=BURGER;
        VAR Calories Fat_Cal Protein Fat Sat_Fat Chol Sodium Carbs Fiber Sugar;
RUN;
```

2.5.3.1 Etape 1. ACP sur la totalité des variables.

Synthèse de classification pour 1 classe

Cluster	Membres	Variation cluster	Variation expliquée	Proportion expliquée	Seconde valeur propre
1	10	10	4.527377	0.4527	2.4455

Variation totale expliquée = 4.527377 Proportion = 0.4527

Le cluster 1 sera fractionné car il a la deuxième plus grande valeur propre, 2.445463, qui est supérieure à la valeur MAXEIGEN=1.

SAS nous annonce que la 1ère valeur propre est égale à 4.527377, la seconde à 2.4455. Celle-ci étant supérieure à MAXEIGEN = 1, une première partition est initiée en affectant les variables à la composante qui leur est la plus corrélée.

2.5.3.2 Etape 2. ACP dans les deux-sous groupes de variables.

Nous comprenons dans le tableau suivant que le 1er groupe est composé de 8 variables, le second de 2. Une ACP est réalisée dans les deux sous-groupes. Nous obtenons :

Synthèse de classification pour 2 classes

Cluster	Membres	Variation cluster	Variation expliquée	Proportion expliquée	Seconde valeur propre
1	8	8	4.464196	0.5580	1.3703
2	2	2	1.770141	0.8851	0.2299

Variation totale expliquée = 6.234336 Proportion = 0.6234

- Dans le 1er sous-groupe de 8 variables, la variance du premier facteur de l'ACP est 4.464196, elle restitue 55.8% de l'information disponible. La seconde valeur propre est de 1.3703 (> 1). Il sera possible d'initier une subdivision de ce cluster.
- Dans le 2nd sous-groupe de 2 variables, le premier facteur regroupe 88.51% de l'information, la valeur propre du second facteur est 0.2299. Mis à part la configuration très particulière de deux variables orthogonales, il ne sera jamais possible de subdiviser un cluster composé de 2 variables avec les paramètres par défaut.
- La « variation totale expliquée » correspond à la somme des variations expliquées des premiers facteurs des ACP. En la divisant par le nombre de variables à segmenter (p = 10), nous disposons de la proportion.

SAS enchaîne avec une série de tableaux.

Liste des variables par cluster et tableau des R2. Le tableau des R2 recense les variables dans chaque groupe. Plusieurs indicateurs permettent d'apprécier la qualité de l'affectation : « Propre Cluster » indique le R^2 de la variable avec son groupe c.-à-d. le carré de la corrélation de la variable avec le représentant de la classe, le premier axe de l'ACP sur les variables composant le groupe ; « Le plus proche suivant » indique le R^2 de la variable avec le groupe le plus proche, si cette valeur est plus grande que la première, il y a matière à s'inquiéter.

L'indicateur (1-R^2 Rapport) indique justement le rapport

$$1 - R^2\ rapport = \frac{(1 - R^2)\ propre\ cluster}{(1 - R^2)\ plus\ proche}$$

Plus petite est sa valeur, meilleure est l'affectation de la variable au groupe. S'il est supérieur à 1, cela voudrait dire que la variable est plus corrélée avec un autre cluster qu'avec son propre groupe d'appartenance. Il y a un problème.

Il n'y a pas lieu de se pencher outre mesure sur ce tableau à ce stade. Le processus de partitionnement n'étant pas terminé.

2 Clusters		R-carré avec		1-R**2 Rapport	Libellé de la variable
Cluster	Variable	Propre Cluster	Le plus proche suivant		
Cluster 1	Calories	0.7733	0.2285	0.2939	Calories
	Fat_Cal	0.8298	0.0444	0.1781	Fat_Cal
	Protein	0.4674	0.2258	0.6879	Protein
	Fat	0.8299	0.0366	0.1765	Fat
	Sat_Fat	0.4815	0.0208	0.5294	Sat_Fat
	Chol	0.5262	0.0007	0.4742	Chol
	Sodium	0.4627	0.0515	0.5665	Sodium
	Fiber	0.0933	0.0184	0.9236	Fiber
Cluster 2	Carbs	0.8851	0.0462	0.1205	Carbs
	Sugar	0.8851	0.0003	0.1150	Sugar

Coefficients du score normalisés. Ce tableau recense les coefficients standardisés (définies sur les variables centrées et réduites) des fonctions de projection sur les premières composantes des clusters. Le coefficient d'une variable est naturellement nul pour les groupes qui ne lui sont pas associés. Nous notons la concordance avec le tableau précédent.

Coefficients du score normalisés			
Cluster		1	2
Calories	Calories	0.196981	0.000000
Fat_Cal	Fat_Cal	0.204055	0.000000
Protein	Protein	0.153143	0.000000
Fat	Fat	0.204071	0.000000
Sat_Fat	Sat_Fat	0.155444	0.000000
Chol	Chol	0.162492	0.000000
Sodium	Sodium	0.152373	0.000000
Carbs	Carbs	0.000000	0.531473
Fiber	Fiber	-.068430	0.000000
Sugar	Sugar	0.000000	0.531473

Structure de classe indique la corrélation des variables avec l'ensemble des composantes représentant chaque classe (nous n'en avons que 2 à ce stade du processus de partitionnement). Il traduit également le degré d'appartenance, mais avec une vision plus complète. Si une variable présente une corrélation élevée en valeur absolue avec plusieurs facteurs, nous concluons qu'elle présente un fort rayonnement et impacte la structure globale des relations entre les variables.

Structure de classe			
Cluster		1	2
Calories	Calories	0.879361	0.477971
Fat_Cal	Fat_Cal	0.910941	0.210736
Protein	Protein	0.683662	-.475146
Fat	Fat	0.911015	0.191396
Sat_Fat	Sat_Fat	0.693932	0.144068
Chol	Chol	0.725394	0.027157
Sodium	Sodium	0.680224	-.227023
Carbs	Carbs	0.214889	0.940782
Fiber	Fiber	-.305485	-.135502
Sugar	Sugar	-.017450	0.940782

Comme pour le tableau de R2, s'attarder sur ce tableau n'est pas intéressant à ce stade. Il en sera autrement lors de l'inspection du résultat final.

Comme l'ACP sur le groupe de 8 variables présente un second facteur porté par une valeur propre supérieure au seuil (MAXEIGEN = 1). Il est subdivisé en deux sous-groupes. Et ainsi de suite....

2.5.3.3 Résultat final de la partition

A l'issue du processus, nous disposons d'une partition en 4 classes d'effectifs (3, 2, 3, 2) qui explique 82.83% de la variance totale. La seconde valeur propre nous donne une idée de la compacité des groupes. Elle varie en sens inverse de la proportion expliquée par le 1er facteur.

La « Synthèse de classification » résume ces principaux indicateurs.

Synthèse de classification pour 4 classes					
Cluster	Membres	Variation cluster	Variation expliquée	Proportion expliquée	Seconde valeur propre
1	3	3	2.881318	0.9604	0.1180
2	2	2	1.770141	0.8851	0.2299
3	3	3	1.882167	0.6274	0.7260
4	2	2	1.749269	0.8746	0.2507

Variation totale expliquée = 8.282894 Proportion = 0.8283

Nous observons l'appartenance aux classes dans le tableau des R2. Nous observons les « dimensions » suivantes dans les produits de l'enseigne :

- La teneur en graisses, caloriques à souhait.
- La teneur en glucides.
- La teneur en fibres, opposée aux graisses saturées.
- La teneur en protéine et sodium. Elle caractérise sûrement les produits carnés.

4 Clusters		R-carré avec			
Cluster	Variable	Propre Cluster	Le plus proche suivant	1-R**2 Rapport	Libellé de la variable
Cluster 1	Calories	0.9198	0.2686	0.1097	Calories
	Fat_Cal	0.9835	0.2590	0.0222	Fat_Cal
	Fat	0.9780	0.2609	0.0298	Fat
Cluster 2	Carbs	0.8851	0.1742	0.1392	Carbs
	Sugar	0.8851	0.2675	0.1569	Sugar
Cluster 3	Sat_Fat	0.7671	0.2855	0.3260	Sat_Fat
	Chol	0.5846	0.2913	0.5862	Chol
	Fiber	0.5305	0.0198	0.4790	Fiber
Cluster 4	Protein	0.8746	0.3259	0.1860	Protein
	Sodium	0.8746	0.3247	0.1857	Sodium

Les tableaux « Coefficients du score normalisés » et « Structure de classe » précisent ces résultats. Nous ne nous y attardons pas dans notre contexte.

Penchons-nous plutôt sur le dernier tableau récapitulatif qui retrace les étapes du partitionnement.

Nombre de clusters	Total Variation expliquée par Clusters	Proportion de Variation expliquée par clusters	Proportion minimale expliquée par un cluster	Seconde valeur propre max. dans un cluster	R-carré min. pour une variable	Rapport 1-R**2 max. pour une variable
1	4.527377	0.4527	0.4527	2.445463	0.0032	
2	6.234336	0.6234	0.5580	1.370331	0.0933	0.9236
3	7.274261	0.7274	0.6685	1.079944	0.4442	0.7178
4	8.282894	0.8283	0.6274	0.726025	0.5305	0.5862

L'évolution des différents indicateurs en fonction du nombre de groupes peut nous aider à identifier la bonne solution. Même si, soyons honnête, la tâche est aussi difficile que lors de la classification automatique des observations.

2.5.3.4 Dendrogramme

SAS propose enfin un dendrogramme (Figure 59) qui permet de lire plus simplement la hiérarchie de solutions. Les paliers correspondent à la proportion de variance expliquée du tableau précédent. Comme en classification ascendante hiérarchique (CAH), même si l'algorithme est descendant pour VARCLUS, les écarts entre paliers peuvent constituer une indication sur la solution à adopter.

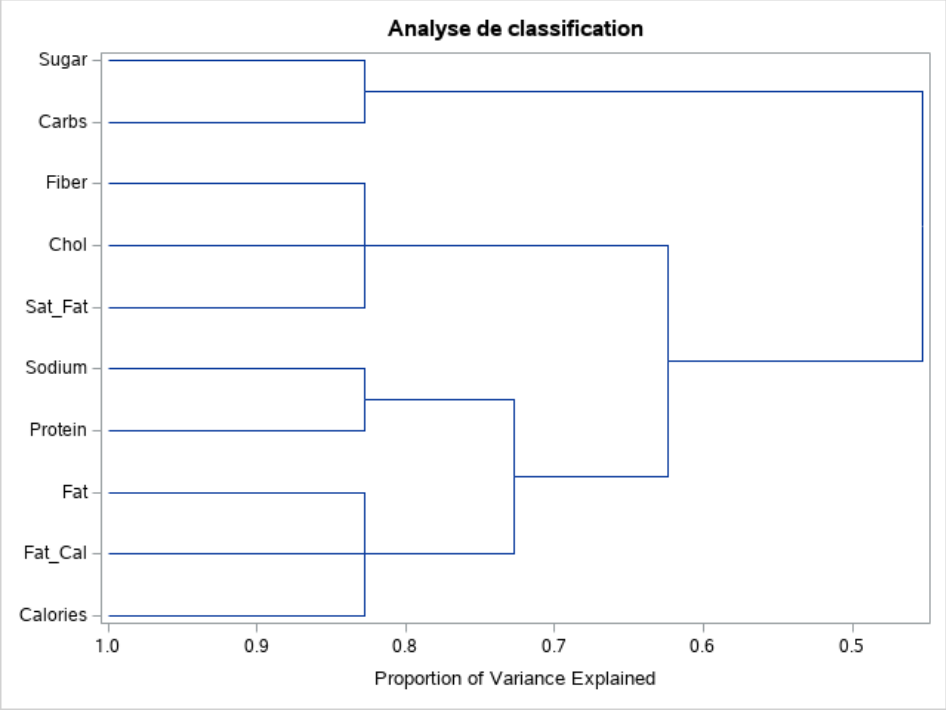


Figure 59 – Dendrogramme – VARCLUS – Données "Burger King"

2.5.4 VARCLUS avec le package « varclushi » sous Python

J'ai identifié le package « varclushi » en effectuant des recherches sur le web. Après quelques tests, il me paraît tenir la route. Voyons son utilisation sur les données « Burger King ».

Nous chargeons les données, nous récupérons les variables actives (« Calories », ..., « Sugar »).

```
#chargement - index_col = 0 pour indiquer que la colonne n°0 est un label
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="BURGER_ACP",index_col=0)

#affichage des caractéristiques
print(D.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 17 entries, BK__Double_Stacker to Tacos(2)
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Calories      17 non-null     float64
 1   Fat_Cal       17 non-null     float64
 2   Protein       17 non-null     float64
 3   Fat           17 non-null     float64
 4   Sat_Fat       17 non-null     float64
 5   Chol          17 non-null     float64
 6   Sodium        17 non-null     float64
 7   Carbs         17 non-null     float64
 8   Fiber         17 non-null     float64
 9   Sugar         17 non-null     float64
 10  Serving_size  17 non-null     int64
 11  Meat          17 non-null     object
dtypes: float64(10), int64(1), object(1)
memory usage: 1.7+ KB
```

```
#récupérer les variables actives
X = D.iloc[:,:-2]
print(X.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 17 entries, BK__Double_Stacker to Tacos(2)
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Calories  17 non-null     float64
 1   Fat_Cal   17 non-null     float64
 2   Protein   17 non-null     float64
 3   Fat       17 non-null     float64
 4   Sat_Fat   17 non-null     float64
 5   Chol      17 non-null     float64
 6   Sodium    17 non-null     float64
 7   Carbs     17 non-null     float64
 8   Fiber     17 non-null     float64
 9   Sugar     17 non-null     float64
dtypes: float64(10)
```

```
memory usage: 1.5+ KB
```

Nous initions le clustering après avoir importé la classe de calcul.

```
#librairie
from varclushi import VarClusHi
vc = VarClusHi(X,maxeigval2=1,maxclus=None)
vc.varclus()
```

```
<varclushi.varclushi.VarClusHi at 0x2716e94e788>
```

```
#propriétés de l'objet
print(dir(vc))
```

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format
__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subcl
ass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__re
duce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__'
, '__weakref__', '_calc_tot_var', '_reassign', '_reassign_rs', '_rsquarespu', '_va
rclusspu', 'clusters', 'correig', 'corrs', 'df', 'feat_list', 'info', 'maxclus', '
maxeigval2', 'n_rs', 'pca', 'rsquare', 'speedup', 'varclus']
```

La documentation en ligne n'est pas très diserte sur les fonctionnalités de l'outil. Nous comprenons plus ou moins le rôle des fonctions et propriétés à partir de leurs noms.

Nous utilisons la propriété « clusters » pour obtenir le détail des groupes.

```
#détail des groupes
vc.clusters
```

```
OrderedDict([(0, ClusInfo(clus=['Calories', 'Fat_Cal', 'Fat'], eigval1=2.881317958
2029385, eigval2=0.11802483451854862, eigvecs=array([[-0.56500245, -0.82434165],
                    [-0.58424886,  0.36966646],
                    [-0.58260236,  0.42872783]]), varprop=0.9604393194009796)),
             (1, ClusInfo(clus=['Protein', 'Sodium'], eigval1=1.7492687429438147,
eigval2=0.2507312570561852, eigvecs=array([[ 0.70710678, -0.70710678],
                    [ 0.70710678,  0.70710678]]), varprop=0.8746343714719074)),
             (2, ClusInfo(clus=['Sat_Fat', 'Fiber', 'Chol'], eigval1=1.88216657776
23848, eigval2=0.7260248340990941, eigvecs=array([[-0.63840362,  0.04838188],
                    [ 0.53088122,  0.75031894],
                    [-0.55732033,  0.65930318]]), varprop=0.6273888592541281)),
             (3, ClusInfo(clus=['Carbs', 'Sugar'], eigval1=1.7701408572769308, eig
val2=0.22985914272306907, eigvecs=array([[ 0.70710678, -0.70710678],
                    [ 0.70710678,  0.70710678]]), varprop=0.8850704286384654))])
```

Nous distinguons tour à tour : la liste des variables ; les valeurs propres de la 1ère et 2ème composantes ; les vecteurs propres qui permettent de projeter les individus sur ces composantes ; la proportion de variance restituée par chaque facteur.

Le tableau des R2 est fourni.

```
#tableau des r2
print(vc.rsquare)
```

	Cluster	Variable	RS_Own	RS_NC	RS_Ratio
0	0	Calories	0.919797	0.268559	0.109651
1	0	Fat_Cal	0.983528	0.259028	0.022230
2	0	Fat	0.977993	0.260903	0.029776
3	1	Protein	0.874634	0.325904	0.185976
4	1	Sodium	0.874634	0.324749	0.185658
5	2	Sat_Fat	0.767094	0.285525	0.325982
6	2	Fiber	0.530460	0.019814	0.479031
7	2	Chol	0.584612	0.291346	0.586165
8	3	Carbs	0.885070	0.174194	0.139173
9	3	Sugar	0.885070	0.267489	0.156898

Les résultats correspondent en tous points à ceux de SAS, au moins sur cet exemple.

2.5.5 VARCLUS avec TANAGRA

J'ai beaucoup travaillé sur le clustering de variables (TUTO 11). Je trouve qu'il n'est pas assez mis en avant dans les références francophones. J'essaie d'y sensibiliser mes étudiants, d'autant plus que nous travaillons de plus en plus sur des données à forte dimensionnalité dans les applications de machine learning. Plusieurs algorithmes de classification autour de composantes latentes ont été implémentées dans TANAGRA : ascendante, descendante, et une variante des K-Means. Seules les techniques fondées sur le carré de la corrélation (r^2) sont disponibles, nous retrouvons ainsi l'interprétation classique des composantes principales.

Pour ce qui est de VARCLUS en particulier, deux modifications ont été introduites : une rotation VARIMAX (section 2.2) est utilisée pour faire pivoter les facteurs de l'ACP dans le plan ; il n'y a pas de procédé de réaffectation des variables à chaque subdivision, il se révèle gourmand en calculs sans être réellement décisif sur les très grandes bases de données. De fait, le dendrogramme de TANAGRA retrace simplement le processus de découpage récursif. Les paliers ne correspondent pas à des proportions de variance expliquée.

ACP + ROTATION VARIMAX. Pour apprécier pleinement les résultats de notre variante de VARCLUS, voici les résultats de l'ACP avec une rotation VARIMAX sur les 3 premiers axes.

Rotated Factor Loadings

Attribute	Axis_1		Axis_2		Axis_3	
-	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)
Fat_Cal	0.93396	*87 % (87 %)*	0.08324	*1 % (88 %)*	0.20124	*4 % (92 %)*
Fat	0.93103	*87 % (87 %)*	0.06634	*0 % (87 %)*	0.20229	*4 % (91 %)*
Calories	0.91490	*84 % (84 %)*	0.28997	*8 % (92 %)*	0.23363	*5 % (98 %)*
Sodium	0.75901	*58 % (58 %)*	-0.48265	*23 % (81 %)*	-0.07393	*1 % (81 %)*
Sugar	0.01040	*0 % (0 %)*	0.93011	*87 % (87 %)*	0.15759	*2 % (89 %)*
Carbs	0.36132	*13 % (13 %)*	0.84721	*72 % (85 %)*	-0.00692	*0 % (85 %)*
Protein	0.46658	*22 % (22 %)*	-0.70900	*50 % (72 %)*	0.43672	*19 % (91 %)*
Fiber	0.08106	*1 % (1 %)*	-0.08945	*1 % (1 %)*	-0.85190	*73 % (74 %)*
Sat_Fat	0.38179	*15 % (15 %)*	0.03288	*0 % (15 %)*	0.77864	*61 % (75 %)*
Chol	0.48680	*24 % (24 %)*	-0.15433	*2 % (26 %)*	0.59759	*36 % (62 %)*
Var. Expl.	3.88991	*39 % (39 %)*	2.44681	*24 % (63 %)*	2.04622	*20 % (84 %)*

Figure 60 – ACP + Rotation VARIMAX (q = 3 facteurs) – Données "Burger King"

Il y a 3 principales « dimensions » dans les variables. Les mêmes que celles que nous avions perçues avec l'ACP usuelle (sous SAS PRINCOMP, TANAGRA, R FACTOMINER ; section 1.7), mais de manière autrement plus tranchée grâce à la rotation des facteurs : la graisse calorique salée, les glucides opposés aux protéines, la grasse saturée opposée aux fibres.

VARCLUS. Nous lançons ensuite la procédure VARCLUS.

Cluster summary

Cluster	# Members	Variation Explained	Proportion Explained
1	4	3.2899	0.8225
2	3	2.1283	0.7094
3	3	1.8822	0.6274
Total		7.3004	0.7300

Cluster members and R-square values

Cluster	Members	Own Cluster	Next Closest	1-R^2 ratio
1	Calories	0.8962	0.2686	0.1419
	Fat_Cal	0.9492	0.2591	0.0685
	Fat	0.9463	0.2592	0.0724
	Sodium	0.4982	0.0346	0.5198
2	Protein	0.5162	0.3260	0.7177
	Carbs	0.7319	0.0083	0.2703
	Sugar	0.8801	0.0239	0.1228
3	Sat_Fat	0.7671	0.2497	0.3104
	Chol	0.5848	0.2849	0.5806
	Fiber	0.5302	0.0112	0.4751

Figure 61 – VARCLUS – TANAGRA – Données "Burger King"

TANAGRA produit 3 groupes d'effectifs (4, 3, 3) correspondants respectivement à (Calories, Fat_Cal, Fat, Sodium), (Protein, Carbs, Sugar), (Sat_Fat, Chol, Fiber), avec une proportion de variance expliquée de 73%.

Au regard du dendrogramme sous SAS, pour une solution analogue à 3 classes, la PROC VARCLUS rapporte 72.74% de la variance avec les groupes (Calories, Fatc_Cal, Fat, Sodium, Protein), (Carbs, Sugar), (Sat_Fat, Chol, Fibder). Seule la situation de « Protein » serait modifiée. On notera d'ailleurs que c'est la variable la plus litigieuse avec un (1 – R^2 ratio = 0.7177) (Figure 61).

Sinon, les résultats sont très proches globalement. L'appartenance aux groupes est plutôt bien dessinée comme le confirme le « tableau des structures de classes » (Figure 62).

Cluster correlations -- Structure

Attribute	# membership	Cluster 1	Cluster 2	Cluster 3
Calories	1	0.9467	0.2571	0.5183
Fat_Cal	1	0.9743	0.0316	0.5090
Protein	1	0.4905	-0.7185	0.5709
Fat	1	0.9728	0.0141	0.5092
Sat_Fat	1	0.4997	-0.0441	0.8758
Chol	1	0.5338	-0.1773	0.7647
Sodium	1	0.7058	-0.4406	0.1860
Carbs	1	0.3566	0.8555	0.0908
Fiber	1	-0.1057	-0.0118	-0.7282
Sugar	1	0.0344	0.9382	0.1547

Figure 62 – Structures de classes – VARCLUS – TANAGRA – Données "Burger King"

Nous disposons des corrélations de chaque variable avec l'ensemble des classes. Lorsque que la corrélation est supérieure à 0.7 (ou inférieure à –0.7), ce paramètre est modifiable, elle est mise en surbrillance (orange ou jaune selon le signe) et elle est recensée dans la colonne MEMBERS. Dans l'idéal, chaque variable ne devrait être significativement corrélée qu'avec une et une seule classe, c'est le cas ici.

Variables illustratives. Chaque groupe de variables est représenté par une variable synthétique correspondant à la première composante de l'ACP (sur les variables du groupe). Elle nous autorise

tout type de post-traitement, y compris l'interprétation des résultats à l'aide des variables illustratives (section 1.6).

Pour « Serving_Size » (taille du produit), nous constatons qu'elle surtout liée au 2nd cluster.

Y	X	r	r²	t	Pr(>|t|)
Serving_size	VarClus_1_1	-0.0860	0.0074	-0.3344	0.7427
Serving_size	VarClus_1_2	-0.5096	0.2597	-2.2938	0.0367
Serving_size	VarClus_1_3	0.1872	0.0350	0.7381	0.4719

Figure 63 – Corrélation – Variable illustrative quantitative – VARCLUS – Données "Burger King"

Pour « Meat » (présence de viande dans le produit), elle caractérise aussi le 2nd cluster.

Attribute_Y	Attribute_X	Description				Statistical test		
		Value	Examples	Average	Std-dev	Variance decomposition		
		yes	12	0.5742	1.1072	Source	Sum of square	d.f.
		no	5	-1.3781	2.6922	BSS	13.4524	1
		All	17	0.0000	1.8696	WSS	42.4767	15
VarClus_1_1	Meat					TSS	55.9291	16
						Significance level		
						Statistics	Value	Proba
						Fisher's F	4.750502	0.045639
		Value	Examples	Average	Std-dev	Variance decomposition		
		yes	12	-0.6825	0.5674	Source	Sum of square	d.f.
		no	5	1.6380	1.8462	BSS	19.0048	1
		All	17	0.0000	1.5038	WSS	17.1765	15
VarClus_1_2	Meat					TSS	36.1813	16
						Significance level		
						Statistics	Value	Proba
						Fisher's F	16.596677	0.000998
		Value	Examples	Average	Std-dev	Variance decomposition		
		yes	12	0.2111	1.5090	Source	Sum of square	d.f.
		no	5	-0.5066	1.1324	BSS	1.8176	1
		All	17	0.0000	1.4141	WSS	30.1792	15
VarClus_1_3	Meat					TSS	31.9968	16
						Significance level		
						Statistics	Value	Proba
						Fisher's F	0.903418	0.356931

Figure 64 – ANOVA – Variable illustrative qualitative– VARCLUS – Données "Burger King"

2.6 Analyse en facteurs principaux

2.6.1 Restituer la variance totale vs. la variance commune

L'analyse en composantes principales est une technique factorielle populaire. Elle est largement décrite dans l'abondante littérature en langue française consacrée à l'analyse de données.

La situation est un peu différente lorsqu'on s'intéresse aux ouvrages en langue anglaise. L'ACP y est présente certes, mais d'autres approches sont aussi mises en avant, notamment l'analyse en facteurs principaux [AFP] (Principal Factor Analysis en anglais) que l'on dit même préférable à l'ACP. Malgré tout, la plupart des auteurs concèdent que cette dernière reste la plus utilisée.

Qu'est ce qui les distingue, qu'est-ce qui les réunit ? Ce sont des techniques factorielles, raison pour laquelle on les confond bien souvent. Mais l'ACP cherche à résumer de manière la plus efficace possible l'information disponible en s'intéressant à la variabilité totale portée par chaque variable de la base. On peut la voir comme une technique de compression. En revanche, l'AFP cherche à structurer l'information en s'intéressant à la variabilité commune aux variables. L'idée est de mettre en avant des facteurs sous-jacents (variables latentes) qui associent deux ou plusieurs colonnes des données. L'influence des variables qui font cavalier seul, indépendantes des autres, devrait être écartée.

Elles sont donc différentes de par la nature des informations qu'elles exploitent. Mais la nuance n'est pas évidente. D'autant plus qu'elles sont souvent regroupées dans le même outil dans certains logiciels (ex. « PROC FACTOR » dans SAS, « ANALYZE / DATA REDUCTION / FACTOR » dans SPSS, etc.), que les tableaux des résultats sont identiques, et que les interprétations sont finalement très proches.

2.6.2 ACP sur les données « Autos bruitées »

Nous travaillons sur une variante des données « Autos » dans cette section. Aux 6 variables originelles, nous adjoignons 6 variables générées aléatoirement selon une loi normale centrée et réduite (RND1, ..., RND6), d'où l'idée d'une base « bruitée ». Ces variables additionnelles, sans aucun rapport avec les véhicules, risquent de peser de manière indue dans les résultats et

masquer les relations réellement existantes entre les autres. C'est tout l'enjeu de l'analyse en facteurs principaux qui s'intéresse aux informations communes aux variables.

Remarque : Notons que « bruit » n'est peut-être pas le terme le plus approprié pour qualifier ces variables additionnelles. Une variable peut être tout à fait légitime mais n'a aucun rapport avec le sujet de l'étude. Elle peut aussi, et ça devient particulièrement compliqué à ce stade, être parfaitement pertinente mais représenter une « dimension » à elle seule. Il est évidemment hors de question de tenter de l'évacuer dans ce cas.

Matrice des corrélations. Nous réalisons l'ACP normée sous TANAGRA. Elle travaille à partir de la matrice des corrélations (Figure 65). Hors du bloc des variables originelles (vert), les corrélations supérieure à (0.473, la plus petite corrélation chez les variables originelles) en valeur absolue ont été mis en évidence. On sait que ces corrélations sont purement fortuites.

	CYL	PUISS	LONG	LARG	POIDS	V.MAX	RND1	RND2	RND3	RND4	RND5	RND6
CYL	1	0.797	0.701	0.630	0.789	0.665	-0.282	0.236	-0.060	0.567	-0.550	-0.126
PUISS	0.797	1	0.641	0.521	0.765	0.844	-0.377	0.349	-0.108	0.432	-0.576	0.071
LONG	0.701	0.641	1	0.849	0.868	0.476	-0.289	0.218	-0.014	0.309	-0.482	0.309
LARG	0.630	0.521	0.849	1	0.717	0.473	-0.190	0.246	-0.192	0.484	-0.262	0.111
POIDS	0.789	0.765	0.868	0.717	1	0.478	-0.355	0.274	-0.012	0.353	-0.619	0.094
V.MAX	0.665	0.844	0.476	0.473	0.478	1	-0.314	0.266	-0.194	0.376	-0.453	0.112
RND1	-0.282	-0.377	-0.289	-0.190	-0.355	-0.314	1	-0.441	0.021	-0.176	0.372	-0.020
RND2	0.236	0.349	0.218	0.246	0.274	0.266	-0.441	1	0.083	0.429	-0.249	0.012
RND3	-0.060	-0.108	-0.014	-0.192	-0.012	-0.194	0.021	0.083	1	-0.028	-0.383	-0.019
RND4	0.567	0.432	0.309	0.484	0.353	0.376	-0.176	0.429	-0.028	1	-0.139	-0.241
RND5	-0.550	-0.576	-0.482	-0.262	-0.619	-0.453	0.372	-0.249	-0.383	-0.139	1	-0.015
RND6	-0.126	0.071	0.309	0.111	0.094	0.112	-0.020	0.012	-0.019	-0.241	-0.015	1

Figure 65 – Matrice des corrélations – Données "Autos bruitées"

Bien évidemment, la corrélation d'une variable avec elle-même est égale à 1. C'est ce que nous indique la diagonale principale de la matrice. Cette remarque est tout sauf anodine. En effet, l'ACP va en tenir compte lorsqu'elle diagonalisera la matrice de corrélation. Elle traitera la variabilité totale des variables en leur accordant la même importance.

Significance of Principal Components

Global critical values	
Kaiser-Guttman	1
Karlis-Saporta-Spinaki	2.6088

Eigenvalue table - Test for significance

Eigenvalues - Significance		
Axis	Eigenvalue	Broken-stick critical values
1	5.401106	3.103211
2	1.427143	2.103211
3	1.366423	1.603211
4	1.090817	1.269877
5	0.912116	1.019877
6	0.695369	0.819877
7	0.394765	0.653211
8	0.277786	0.510354
9	0.222135	0.385354
10	0.132830	0.274242
11	0.047628	0.174242
12	0.031884	0.083333

Figure 66 – ACP – Données "Autos bruitées" – Valeurs propres

Valeurs propres. Si l'on s'en tient à la règle de Kaiser-Guttman pour l'ACP normée, nous devrions sélectionner les 4 premiers facteurs. Avec les autres règles de détection (Karlis – Saporta – Spinaki) et « bâtons brisés », nous n'en retiendrons qu'un seul (Figure 66).

Le « **Scree plot** » nous laisse assez perplexe. Le coude, très marqué, est au niveau (q = 2) facteurs. Mais surtout une décroissance en pente linéaire (et non pas concave) de la courbe des valeurs propres n'est jamais bon signe généralement. Elle est caractéristique de la présence de variables qui n'ont aucun rapport entre eux dans la base. Nous avions déjà observé cette forme de courbe sur un autre jeu de données où nous avions également ajouté des variables générées aléatoirement (TUTO 9).

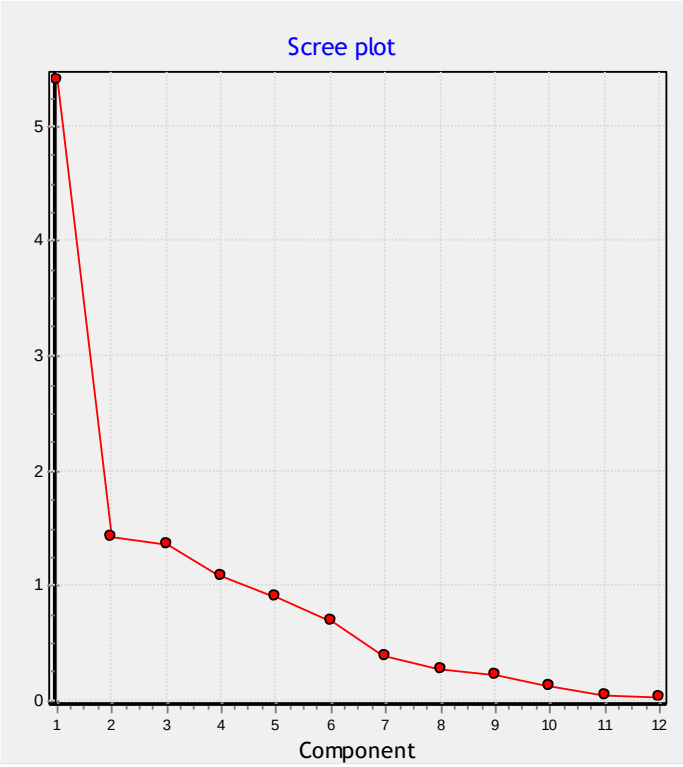


Figure 67 – Eboulis des valeurs propres – ACP – Données "Autos bruitées"

Redondance entre les descripteurs. Lorsque nous nous intéressons au degré de redondance des variables (indice KMO, section 2.3.3), mis à part CYL qui joue un rôle particulier (MSA_{CYL} = 0.8082), les autres variables originelles ne se démarquent pas. Elles sont noyées au milieu des descripteurs générés aléatoirement (RND1, ..., RND6). L'indice KMO global laisse penser à une certaine disparité entre les variables néanmoins (Overall MSA = 0.6735).

Kaiser's Measure of Sampling Adequacy (MSA)

Overall MSA = 0.6735478									
CYL	0.8082399	PUISS	0.7039763	LONG	0.7135369	LARG	0.6245607	POIDS	0.7194188
V.MAX	0.5823891	RND1	0.7455337	RND2	0.7167359	RND3	0.2946817	RND4	0.6496068
RND5	0.7308624	RND6	0.213933						

Figure 68 – Données "Autos bruitées" – Indices KMO global et par variable

Corrélations des variables avec les facteurs. Si l'on sélectionne les 4 premiers facteurs, les corrélations des variables avec les axes regroupent les variables originelles sur le 1er facteur.

Factor Loadings [Communality Estimates]

Attribute	Axis_1		Axis_2		Axis_3		Axis_4	
-	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)	Corr.	% (Tot. %)
PUISS	-0.89120	79 % (79 %)	-0.01566	0 % (79 %)	-0.02169	0 % (79 %)	0.03559	0 % (80 %)
POIDS	-0.89050	79 % (79 %)	-0.04878	0 % (80 %)	0.18506	3 % (83 %)	-0.15363	2 % (85 %)
CYL	-0.88465	78 % (78 %)	-0.02946	0 % (78 %)	-0.14629	2 % (80 %)	-0.26809	7 % (88 %)
LONG	-0.84520	71 % (71 %)	-0.22900	5 % (77 %)	0.31665	10 % (87 %)	-0.09108	1 % (88 %)
LARG	-0.77022	59 % (59 %)	-0.36959	14 % (73 %)	-0.01912	0 % (73 %)	-0.10238	1 % (74 %)
V.MAX	-0.75412	57 % (57 %)	-0.11451	1 % (58 %)	-0.08197	1 % (59 %)	0.13179	2 % (61 %)
RND5	0.64680	42 % (42 %)	-0.49954	25 % (67 %)	-0.34220	12 % (78 %)	0.17182	3 % (81 %)
RND4	-0.55916	31 % (31 %)	0.07304	1 % (32 %)	-0.60583	37 % (69 %)	-0.08726	1 % (69 %)
RND3	0.04741	0 % (0 %)	0.75480	57 % (57 %)	0.37444	14 % (71 %)	-0.32634	11 % (82 %)
RND6	-0.08106	1 % (1 %)	-0.32463	11 % (11 %)	0.71317	51 % (62 %)	0.44043	19 % (81 %)
RND1	0.46948	22 % (22 %)	-0.35500	13 % (35 %)	0.02929	0 % (35 %)	-0.57526	33 % (68 %)
RND2	-0.44129	19 % (19 %)	0.40681	17 % (36 %)	-0.26295	7 % (43 %)	0.53795	29 % (72 %)
Var. Expl.	5.40111	45 % (45 %)	1.42714	12 % (57 %)	1.36642	11 % (68 %)	1.09082	9 % (77 %)

Figure 69 – ACP – Corrélations des variables avec les composantes – Données "Autos bruitées"

Certes. Mais nous notons que RND5 et RND4 se sont sournoisement faufilés parmi les variables fortement contributives au 1er facteur. Les autres, facteurs n°2 à n°4 sont de la poudre aux yeux. Nous les savons parce que nous avons tripatouillé une base artificielle. Dans une étude réelle, avec une base dont on ne maîtrise pas la teneur, nous n'avons pas vraiment la possibilité d'identifier ces situations. Nous considérons que chaque variable pratiquement correspond à une dimension à elle seule. En soi, ces conclusions ne sont pas fausses.

Factor Scores – Coefficients de projection. Les coefficients de projection ont un rôle opérationnel. Ils permettent de calculer les coordonnées des individus dans le repère factoriel. Mais ils peuvent aussi servir à l'interprétation. En effet, ils s'appliquent sur des variables standardisées (centrées et réduites), il est par conséquent possible de les comparer pour identifier les variables qui ont le plus fort impact dans la définition de chaque facteur. On peut les voir comme une autre forme de la « contribution des variables aux axes », différentes de la définition usuelle certes (section 1.3.2), mais tout aussi intéressante parce qu'elle traduit l'influence de la variable dans le positionnement relatif des individus dans le repère factoriel.

Factor Score Coefficients

Attribute	Mean	Std-dev	Axis_1	Axis_2	Axis_3	Axis_4
CYL	1631.6666667	363.3944903	-0.3806551	-0.0246606	-0.1251477	-0.2566894
PUISS	84.6111111	19.8021853	-0.3834706	-0.0131069	-0.0185532	0.0340733
LONG	433.5000000	21.4844905	-0.3636804	-0.1916903	0.2708880	-0.0872028
LARG	166.6666667	5.1639778	-0.3314154	-0.3093798	-0.0163566	-0.0980265
POIDS	1078.8333333	133.0990650	-0.3831717	-0.0408324	0.1583140	-0.1470997
V.MAX	158.2777778	11.7983311	-0.3244878	-0.0958535	-0.0701209	0.1261808
RND1	0.2026111	0.9125674	0.2020098	-0.2971605	0.0250590	-0.5507921
RND2	0.4241111	0.5716308	-0.1898819	0.3405342	-0.2249467	0.5150718
RND3	-0.4291667	0.7927809	0.0203991	0.6318230	0.3203271	-0.3124620
RND4	-0.2902778	0.9949314	-0.2405999	0.0611392	-0.5182716	-0.0835478
RND5	-0.3491111	0.9778135	0.2783081	-0.4181558	-0.2927477	0.1645082
RND6	0.2138333	1.2347276	-0.0348807	-0.2717413	0.6100989	0.4216983

Figure 70 – Factor score coefficients – ACP – Données "Autos bruitées"

Le carré de ces coefficients se comprennent comme des contributions dans le sens où la somme des carrés par axe est égale à 1, du moins dans les logiciels de l'école française de l'analyse des données (la somme est différente dans les logiciels anglo-saxons, SAS notamment, mais le carré rapporté à la somme donne bien la même contribution).

	Axis_1	Axis_2	Axis_3	Axis_4
CYL	0.145	0.001	0.0157	0.0659
PUISS	0.147	0.000	0.0003	0.0012
LONG	0.132	0.037	0.0734	0.0076
LARG	0.110	0.096	0.0003	0.0096
POIDS	0.147	0.002	0.0251	0.0216
V.MAX	0.105	0.009	0.0049	0.0159
RND1	0.041	0.088	0.0006	0.3034
RND2	0.036	0.116	0.0506	0.2653
RND3	0.000	0.399	0.1026	0.0976
RND4	0.058	0.004	0.2686	0.0070
RND5	0.077	0.175	0.0857	0.0271
RND6	0.001	0.074	0.3722	0.1778

Figure 71 – Contributions aux facteurs – Données "Autos bruitées"

Nous avons calculé le carré des coefficients pour chaque facteur, puis nous avons mis en surbrillance les valeurs supérieures à la moyenne des contributions ($\frac{1}{p} = \frac{1}{12} = 0.0833$). La configuration apparaît plus clairement maintenant sur le 1er facteur. Notons que la somme des contributions des variables RND sur le premier facteur est 21.4% pour 6 variables.

Carte des individus. L'effet des variables RND pèse bien évidemment sur la carte des individus. Dans le premier plan factoriel, la dimension « cylindrée – encombrement » est perceptible sur le 1er facteur. La performance est totalement noyée en revanche. Ainsi, certaines proximités observées sont très questionnables : entre la Lada 1300 et l'Alfasud TI par exemple, ou encore entre Alfetta 1.66 et l'Audi 100. En réalité, si l'on poursuit l'exploration, la dimension « sportivité » n'est visible que sur le 5ème facteur… que nous n'avions pas retenu avec les critères usuels (COURS ACP, page 60).

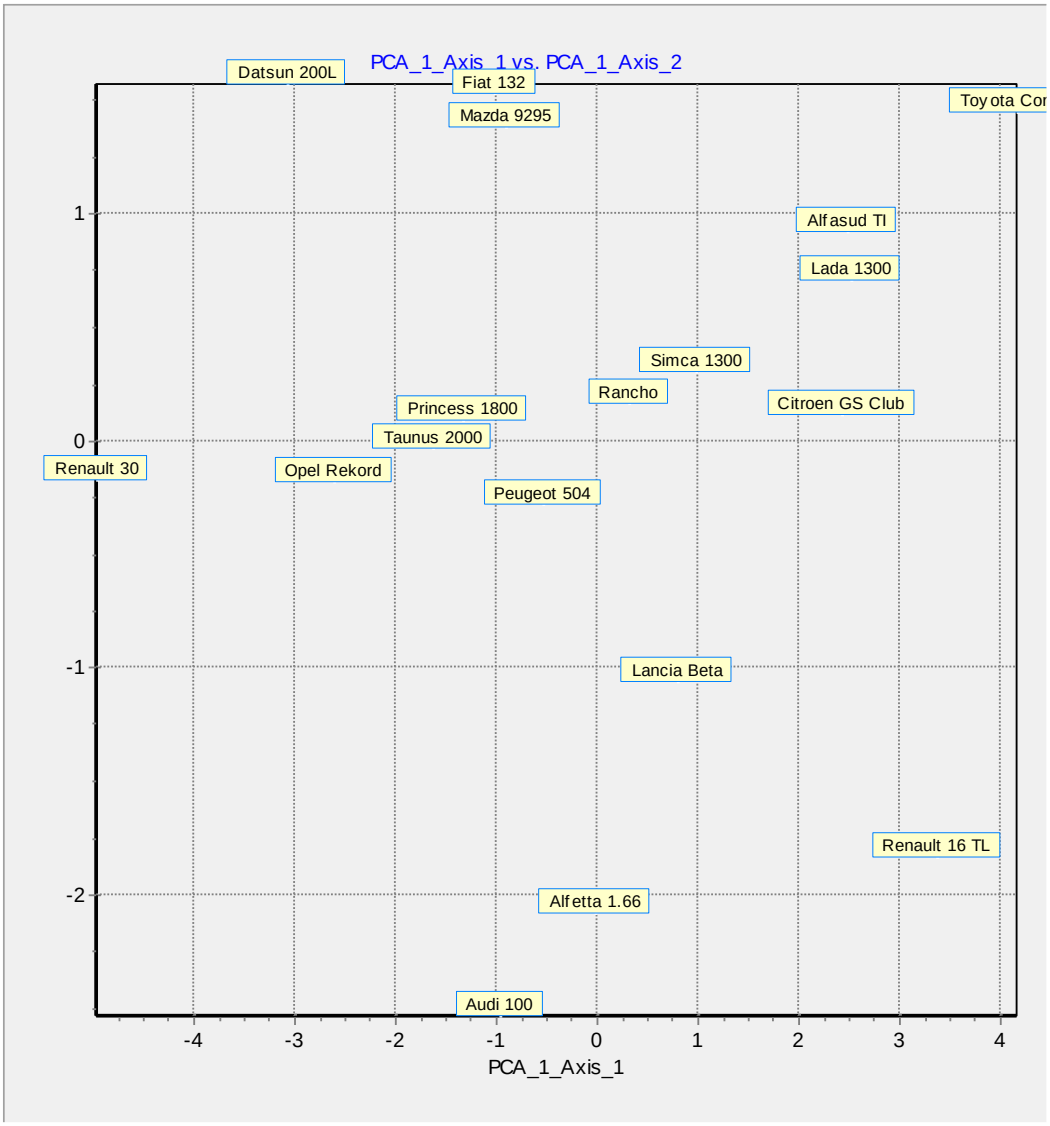


Figure 72 – Carte des individus – ACP – Données "Autos bruitées"

2.6.3 Principe de l'analyse en facteurs principaux (AFP)

L'analyse en facteurs principaux (factor analysis, principal factor analysis, common factor analysis, principal axis factoring) cherche à identifier les facteurs (les variables latentes) qui structurent les données. Ils sont associés à deux ou plusieurs variables de la base. L'analyse s'intéresse exclusivement à la variabilité partagée entre les variables. Elle correspond à une démarche de modélisation. On cherche à construire (q) facteurs communs (F1, F2, …, Fq ; $q \leq p$) qui permettent de reproduire au mieux les variables initiales :

$$\begin{cases} x_1 = a_{11}F_1 + a_{12}\,F_2 + \cdots + a_{1q}F_q + e_1 \\ \cdots \\ x_p = a_{p1}F_1 + a_{p2}\,F_2 + \cdots + a_{pq}\,F_q + e_p \end{cases}$$

Les (e_j) sont des termes d'erreur puisqu'une modélisation n'est jamais parfaite. Les (a_{jk}) sont les coefficients de la « factor loadings matrix », qui ne sont pas des corrélations des variables avec les axes, mais plutôt des coefficients standardisés de la régression de chaque variable (x_j) avec les facteurs. Dans les faits, l'usage du tableau des loadings est le même qu'en ACP, il sert à interpréter le sens et l'intensité de la liaison des variables avec les facteurs.

Nouvelle version de la matrice des corrélations. Plusieurs algorithmes existent. Le plus simple consiste à réaliser une analyse en composantes principales à partir d'une variante (R') de la matrice des corrélations (R) qui met en exergue la variance partagée entre les variables. Pour chaque variable (X_j), nous remplaçons les valeurs de la diagonale principale (qui est égale à 1 lorsque l'on tient compte de toute la variabilité disponible) par la part de variance expliquée par les autres colonnes. Concrètement, il s'agit du coefficient de détermination $(\mathcal{R}_j^2)$ de la régression linéaire multiple de X_j sur les (p-1) autres variables. On parle de « prior communalities » (ou « initial estimates of communalities »). Nous utiliserons le terme de « communalités initiales » (ex. la matrice R' pour les données « Autos bruitées », Figure 73).

A priori, nous devrions mener « p » régressions pour calculer les communalités. Ce qui peut s'avérer très lourd si la base est volumineuse. En pratique, nous les extrayons à partir de l'inverse R^{-1} de la matrice des corrélations R. Nous avons :

$$\mathcal{R}_j^2 = 1 - \frac{1}{r_{jj}^{-1}}$$

Où (r_{jj}^{-1}) est le j[ème] élément diagonal de la matrice (R^{-1}).

L'inertie totale à reproduire est égale à la trace de la matrice (R'), soit :

$$I_p = \sum_{j=1}^{p} \mathcal{R}_j^2$$

L'analyse en facteur principaux revient à diagonaliser la matrice R'.

	CYL	PUISS	LONG	LARG	POIDS	V_MA	RND1	RND2	RND3	RND4	RND5	RND6
CYL	**0.84**	0.80	0.70	0.63	0.79	0.66	-0.28	0.24	-0.06	0.57	-0.55	-0.13
PUISS	0.80	**0.92**	0.64	0.52	0.77	0.84	-0.38	0.35	-0.11	0.43	-0.58	0.07
LONG	0.70	0.64	**0.93**	0.85	0.87	0.48	-0.29	0.22	-0.01	0.31	-0.48	0.31
LARG	0.63	0.52	0.85	**0.88**	0.72	0.47	-0.19	0.25	-0.19	0.48	-0.26	0.11
POIDS	0.79	0.77	0.87	0.72	**0.92**	0.48	-0.36	0.27	-0.01	0.35	-0.62	0.09
V_MA	0.66	0.84	0.48	0.47	0.48	**0.88**	-0.31	0.27	-0.19	0.38	-0.45	0.11
RND1	-0.28	-0.38	-0.29	-0.19	-0.36	-0.31	**0.34**	-0.44	0.02	-0.18	0.37	-0.02
RND2	0.24	0.35	0.22	0.25	0.27	0.27	-0.44	**0.39**	0.08	0.43	-0.25	0.01
RND3	-0.06	-0.11	-0.01	-0.19	-0.01	-0.19	0.02	0.08	**0.45**	-0.03	-0.38	-0.02
RND4	0.57	0.43	0.31	0.48	0.35	0.38	-0.18	0.43	-0.03	**0.61**	-0.14	-0.24
RND5	-0.55	-0.58	-0.48	-0.26	-0.62	-0.45	0.37	-0.25	-0.38	-0.14	**0.69**	-0.02
RND6	-0.13	0.07	0.31	0.11	0.09	0.11	-0.02	0.01	-0.02	-0.24	-0.02	**0.51**

Figure 73 – AFP – Matrice R' à diagonaliser, comprenant les communalités initiales – "Autos bruitées"

Voyons tout cela sous Python. Nous importons la feuille « **AUTOS_AFP** » de notre fichier de données « **Data_Methodes_Factorielles.xlsx** ». Pour tracer nos résultats, nous mettons en parallèle les sorties de la PROC FACTOR de SAS. L'option (PRIORS = SMC) est primordiale. Elle spécifie l'algorithme que nous décrivons dans cette section.

```
PROC FACTOR DATA = AUTOS_NOISY
      METHOD = PRINCIPAL
      PRIORS = SMC
      NFACTORS = 2
      SCORE
      PLOTS = ALL;
      VAR CYL PUISS LONG LARG POIDS VMAX RND1 RND2 RND3 RND4 RND5 RND6;
RUN;
```

Sous Python, nous chargeons et inspectons les données. Nous avons (n = 18) observations et (p = 12) variables, dont les RND1 à RND6.

```
#chargement - index_col = 0 pour indiquer que la colonne n°0 est un label
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="AUTOS_AFP",index_col=0)

#affichage des caractéristiques
print(D.info())

#nombre de variables
p = D.shape[1]
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18 entries, Alfasud TI to Lada 1300
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   CYL     18 non-null     int64
 1   PUISS   18 non-null     int64
 2   LONG    18 non-null     int64
 3   LARG    18 non-null     int64
 4   POIDS   18 non-null     int64
 5   VMAX    18 non-null     int64
 6   RND1    18 non-null     float64
 7   RND2    18 non-null     float64
 8   RND3    18 non-null     float64
 9   RND4    18 non-null     float64
 10  RND5    18 non-null     float64
 11  RND6    18 non-null     float64
dtypes: float64(6), int64(6)
memory usage: 1.8+ KB
None
```

2.6.3.1 Matrice des corrélations

Le calcul de la matrice des corrélations ne pose aucune difficulté. Nous retrouvons la matrice ci-dessus (Figure 65), les valeurs de la diagonale principale sont toutes égales à 1.

```
#matrice des corrélations
R = numpy.corrcoef(D.values,rowvar=False)
numpy.set_printoptions(precision=2)
print(R)
```

```
[[ 1.    0.8   0.7   0.63  0.79  0.66 -0.28  0.24 -0.06  0.57 -0.55 -0.13]
 [ 0.8   1.    0.64  0.52  0.77  0.84 -0.38  0.35 -0.11  0.43 -0.58  0.07]
 [ 0.7   0.64  1.    0.85  0.87  0.48 -0.29  0.22 -0.01  0.31 -0.48  0.31]
 [ 0.63  0.52  0.85  1.    0.72  0.47 -0.19  0.25 -0.19  0.48 -0.26  0.11]
 [ 0.79  0.77  0.87  0.72  1.    0.48 -0.36  0.27 -0.01  0.35 -0.62  0.09]
 [ 0.66  0.84  0.48  0.47  0.48  1.   -0.31  0.27 -0.19  0.38 -0.45  0.11]
 [-0.28 -0.38 -0.29 -0.19 -0.36 -0.31  1.   -0.44  0.02 -0.18  0.37 -0.02]
 [ 0.24  0.35  0.22  0.25  0.27  0.27 -0.44  1.    0.08  0.43 -0.25  0.01]
 [-0.06 -0.11 -0.01 -0.19 -0.01 -0.19  0.02  0.08  1.   -0.03 -0.38 -0.02]
 [ 0.57  0.43  0.31  0.48  0.35  0.38 -0.18  0.43 -0.03  1.   -0.14 -0.24]
```

```
 [-0.55 -0.58 -0.48 -0.26 -0.62 -0.45  0.37 -0.25 -0.38 -0.14  1.   -0.02]
 [-0.13  0.07  0.31  0.11  0.09  0.11 -0.02  0.01 -0.02 -0.24 -0.02  1.  ]]
```

2.6.3.2 Communalités initiales

Pour calculer les communalités initiales, la part de la variance de chaque variable qui est expliquée par les autres, nous inversons la matrice des corrélations et nous appliquons la formule ci-dessus afin d'extraire ($\mathcal{R}_j^2$).

$$communalité\_init_j = \mathcal{R}_j^2$$

```
#inverse de la matrice de corrélations
invR = numpy.linalg.inv(R)
print(invR)
```

```
[[ 6.37  0.2  -3.04  1.95 -2.45 -2.25 -0.38  0.54  0.32 -1.87  0.29  1.54]
 [ 0.2  12.98 -1.04  4.63 -8.61 -8.69 -0.37 -0.8   0.12 -1.1  -1.13  0.41]
 [-3.04 -1.04 13.5  -7.51 -4.11  1.47  0.73  0.44 -1.43  1.69 -0.14 -3.04]
 [ 1.95  4.63 -7.51  8.32 -2.76 -4.03 -0.71 -0.45  0.96 -1.99 -0.97  1.54]
 [-2.45 -8.61 -4.11 -2.76 13.15  6.98  0.41  0.06  0.86  0.91  2.62  0.15]
 [-2.25 -8.69  1.47 -4.03  6.98  8.5   0.48  0.33  0.55  0.82  1.8  -1.03]
 [-0.38 -0.37  0.73 -0.71  0.41  0.48  1.51  0.57 -0.29  0.09 -0.31 -0.23]
 [ 0.54 -0.8   0.44 -0.45  0.06  0.33  0.57  1.65 -0.18 -0.66  0.15 -0.17]
 [ 0.32  0.12 -1.43  0.96  0.86  0.55 -0.29 -0.18  1.82 -0.47  1.29  0.16]
 [-1.87 -1.1   1.69 -1.99  0.91  0.82  0.09 -0.66 -0.47  2.56 -0.46 -0.02]
 [ 0.29 -1.13 -0.14 -0.97  2.62  1.8  -0.31  0.15  1.29 -0.46  3.21 -0.22]
 [ 1.54  0.41 -3.04  1.54  0.15 -1.03 -0.23 -0.17  0.16 -0.02 -0.22  2.03]]
```

```
#R2 des régressions de X avec les autres - communalités initiales
R2Det = 1.0 - 1.0/numpy.diag(invR)
print(pandas.DataFrame(R2Det,index=D.columns))
```

```
              0
CYL    0.843053
PUISS  0.922937
LONG   0.925902
LARG   0.879876
POIDS  0.923971
VMAX   0.882288
RND1   0.336690
RND2   0.394369
RND3   0.450844
RND4   0.610068
RND5   0.688055
RND6   0.506320
```

Le premier groupe des variables originelles se distingue immédiatement (CYL...VMAX). Nous observons aussi des valeurs élevées (RND4, RND5) qui sont purement fortuites, mais pas exceptionnelles pour un échantillon d'aussi petite taille (n = 18) où les artefacts sont légions.

2.6.3.3 Formation et diagonalisation de R'

Nous insérons ces valeurs dans la diagonale de la matrice R pour former R'.

```
#insérer ces valeurs dans la diagonale de la matrice R
Rprim = R.copy()
numpy.fill_diagonal(Rprim,R2Det)
print(Rprim)
```

```
[[ 0.84  0.8   0.7   0.63  0.79  0.66 -0.28  0.24 -0.06  0.57 -0.55 -0.13]
 [ 0.8   0.92  0.64  0.52  0.77  0.84 -0.38  0.35 -0.11  0.43 -0.58  0.07]
 [ 0.7   0.64  0.93  0.85  0.87  0.48 -0.29  0.22 -0.01  0.31 -0.48  0.31]
 [ 0.63  0.52  0.85  0.88  0.72  0.47 -0.19  0.25 -0.19  0.48 -0.26  0.11]
 [ 0.79  0.77  0.87  0.72  0.92  0.48 -0.36  0.27 -0.01  0.35 -0.62  0.09]
 [ 0.66  0.84  0.48  0.47  0.48  0.88 -0.31  0.27 -0.19  0.38 -0.45  0.11]
 [-0.28 -0.38 -0.29 -0.19 -0.36 -0.31  0.34 -0.44  0.02 -0.18  0.37 -0.02]
 [ 0.24  0.35  0.22  0.25  0.27  0.27 -0.44  0.39  0.08  0.43 -0.25  0.01]
 [-0.06 -0.11 -0.01 -0.19 -0.01 -0.19  0.02  0.08  0.45 -0.03 -0.38 -0.02]
 [ 0.57  0.43  0.31  0.48  0.35  0.38 -0.18  0.43 -0.03  0.61 -0.14 -0.24]
 [-0.55 -0.58 -0.48 -0.26 -0.62 -0.45  0.37 -0.25 -0.38 -0.14  0.69 -0.02]
 [-0.13  0.07  0.31  0.11  0.09  0.11 -0.02  0.01 -0.02 -0.24 -0.02  0.51]]
```

La trace de la matrice est égale à $I_p = 8.364$.

```
#trace de la matrice
print(numpy.trace(Rprim))
```

```
8.364373071752006
```

C'est la quantité totale d'information, la variance partagée entre les variables, qu'il faudra décomposer sur les facteurs de l'analyse en facteurs principaux.

Nous diagonalisons R'. Nous affichons dans un premier temps les valeurs propres (λ_k).

```
#diagonalisation de la matrice Rprim
sol1 = numpy.linalg.eigh(Rprim)
numpy.set_printoptions(precision=4)

#valeurs propres - ordre décroissant
print(numpy.flip(sol1[0]))
```

```
[ 5.2295  1.0115  0.9756  0.7232  0.5386  0.2634  0.0959  0.0059 -0.0438
 -0.0914 -0.152  -0.192 ]
```

Valeurs propres de la matrice de corrélation réduite: Total = 8.36437307 Moyenne = 0.69703109				
	Valeur propre	Différence	Proportion	Cumulé
1	5.22952071	4.21804975	0.6252	0.6252
2	1.01147097	0.03588598	0.1209	0.7461
3	0.97558499	0.25234689	0.1166	0.8628
4	0.72323810	0.18463259	0.0865	0.9492
5	0.53860551	0.27519543	0.0644	1.0136
6	0.26341008	0.16755804	0.0315	1.0451
7	0.09585205	0.08995268	0.0115	1.0566
8	0.00589936	0.04972695	0.0007	1.0573
9	-.04382759	0.04758184	-0.0052	1.0521
10	-.09140943	0.06059926	-0.0109	1.0411
11	-.15200869	0.03995431	-0.0182	1.0230
12	-.19196299		-0.0230	1.0000

Première surprise, certaines valeurs propres sont négatives. Ça n'en est pas vraiment une en réalité. La matrice **R'**, contrairement à **R**, n'est pas semi-définie positive. Concrètement, jusqu'au 4ème axe, la somme des valeurs propres n'excède pas la trace de la matrice R', nous traitons bien la variance partagée. A partir du 5ème, la variance intrinsèque aux variables pèse sur les facteurs, au point qu'il faut retrancher de l'information (à partir du 9ème axe) pour que la somme totale des valeurs propres soit égale à la trace de la matrice que nous avons diagonalisée.

Pour identifier le nombre de facteurs à retenir, une règle très simple serait de sélectionner ceux dont la valeur propre est supérieure à la moyenne, qui est égale à 0.697 pour nos données (SAS affiche cette information automatiquement). Dans ce cas, nous retiendrons les 4 premiers axes pour notre analyse.

2.6.3.4 Diagramme des valeurs propres

L'autre piste consiste à étudier l'éboulis des valeurs propres.

```
#librairie graphique
import matplotlib.pyplot as plt

#préparer le graphique – scree plot
fig, ax = plt.subplots(figsize=(5,5))
ax.plot(range(1,p+1),numpy.flip(sol1[0]),".-")
ax.set_xlabel("Nb. facteurs")
ax.set_ylabel("Val. propres")
plt.title("Eboulis des valeurs propres")
plt.show()
```

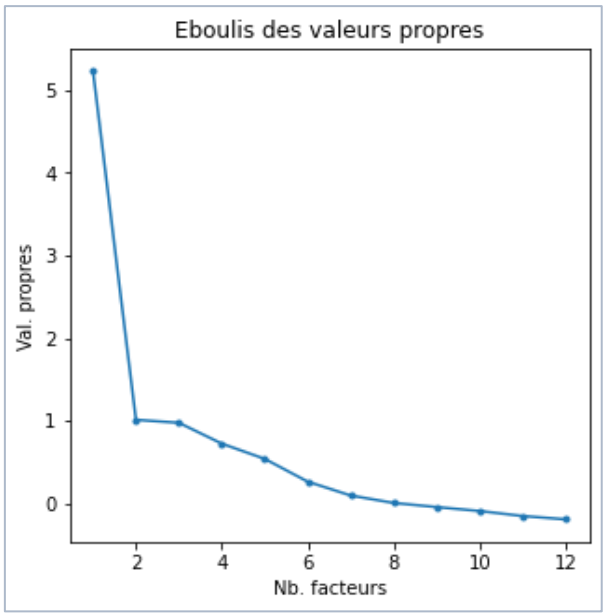


Figure 74 – Eboulis des valeurs propres – AFP – Données "Autos bruitées"

On serait plutôt enclin à retenir (q = 2) facteurs au regard de ce critère. Nous nous en tiendrons à ce choix pour la suite. Il a été spécifié dans la commande SAS (option NFACTORS = 2).

2.6.3.5 Loadings – Représentation des facteurs

Les « loadings » (factor pattern, « représentation du facteur » en français) ne correspondent pas aux corrélations des variables avec les axes dans le cas de l'analyse en facteurs principaux. Il s'agirait plutôt des coefficients standardisés $(a_{jk}\,; j = 1, \ldots, p\,; k = 1, \ldots, q)$ de la régression de chaque variable avec les facteurs. Dans les faits, l'usage du tableau des « loadings » reste le même. Il sert à interpréter les axes par le sens et l'intensité des relations avec les variables.

Pour les obtenir, nous multiplions les coefficients des vecteurs propres (v_{jk}) par la racine carrée de leurs valeurs propres respectives.

$$a_{jk} = v_{jk} \times \sqrt{\lambda_k}$$

Puisque les vecteurs propres sont normés (c.-à-d. $\sum_{j=1}^{p} v_{jk}^2 = 1$), la somme des carrés des « loadings » par facteur correspond à la variance restituée :

$$\sum_{j=1}^{p} a_{jk}^2 = \lambda_k$$

```
#vecteurs propres - ordre décroissant des v.p.
print(numpy.flip(sol1[1],axis=0))
```

```
[[-0.1128  0.1206 -0.2931  0.3053 -0.1429  0.3271 -0.293  -0.3636  0.3391 0.4155 -0.3991  0.0339]
 [-0.2721 -0.5556  0.2305 -0.0876 -0.0628  0.3186  0.0208 -0.1636 -0.013 -0.4373 -0.3992 -0.2703]
 [-0.2396  0.5003  0.1276  0.1505 -0.0529  0.2824 -0.3462 -0.1981 -0.3164 -0.4758  0.1807  0.2293]
 [-0.3175 -0.4172  0.1055 -0.0699  0.1027  0.0673 -0.4822  0.0053 -0.2957 0.4491  0.4148 -0.0201]
 [ 0.5987 -0.2866 -0.0276  0.0268  0.0451  0.0812  0.0371 -0.6542 -0.0474 -0.0918  0.2827  0.1714]
 [ 0.5854  0.002   0.0458  0.092   0.2617  0.2752 -0.4602  0.4328 -0.1209 -0.0566 -0.2298 -0.1811]
 [-0.0129 -0.1672  0.4052  0.2387  0.0032 -0.3508 -0.3466  0.086   0.5936 -0.1919  0.0581  0.328 ]
 [-0.0535 -0.1043  0.3172  0.5207  0.2928  0.2723  0.4392  0.1306 -0.1994 0.2074 -0.0944  0.3911]
 [-0.1099 -0.1353 -0.3735 -0.0203  0.3985 -0.4671 -0.1683 -0.1194 -0.322 -0.1269 -0.4214  0.336 ]
 [ 0.1584  0.1055  0.452  -0.488  -0.3609 -0.0178 -0.0628 -0.0547 -0.1993 0.2691 -0.3652  0.372 ]
 [-0.0987 -0.0102 -0.2006 -0.5253  0.3981  0.44    0.0529  0.1213  0.3663 -0.0643  0.1164  0.3906]
 [ 0.0627 -0.3193 -0.4265  0.1405 -0.5988  0.1252 -0.0088  0.3607 -0.1037 -0.1549  0.0845  0.3824]]
```

```
#loadings des variables avec les facteurs F1, F2
cor1 = sol1[1][:,-1]*numpy.sqrt(sol1[0][-1])
cor2 = sol1[1][:,-2]*numpy.sqrt(sol1[0][-2])
```

```
#loadings
loadings = numpy.transpose([cor1,cor2])
print(pandas.DataFrame(loadings,index=D.columns))
```

```
              0         1
CYL    0.874463  0.084990
PUISS  0.893121  0.117049
LONG   0.850778 -0.367295
LARG   0.768420 -0.423811
POIDS  0.894371 -0.094914
VMAX   0.749973  0.058408
RND1  -0.414234 -0.231072
RND2   0.391899  0.284307
RND3  -0.046057  0.417174
RND4   0.524426  0.181777
RND5  -0.618223 -0.401492
RND6   0.077452 -0.401383
```

Représentation du facteur			
		Factor1	Factor2
CYL	CYL	0.87446	-0.08499
PUISS	PUISS	0.89312	-0.11705
LONG	LONG	0.85078	0.36730
LARG	LARG	0.76842	0.42381
POIDS	POIDS	0.89437	0.09491
VMAX	VMAX	0.74997	-0.05841
RND1	RND1	-0.41423	0.23107
RND2	RND2	0.39190	-0.28431
RND3	RND3	-0.04606	-0.41717
RND4	RND4	0.52443	-0.18178
RND5	RND5	-0.61822	0.40149
RND6	RND6	0.07745	0.40138

SAS

Variance expliquée par chaque facteur	
Factor1	Factor2
5.2295207	1.0114710

```
#variance restituée par chaque facteur
print(numpy.sum(loadings**2,axis=0))
```

```
[5.2295 1.0115]
```

2.6.3.6 Communalités initiales et estimées

Les loadings passés au carré permettent de calculer la qualité de représentation des variables sur les facteurs, laquelle peut être cumulée d'un facteur à l'autre, à la manière des COS2 de l'ACP. Pour un nombre de facteurs déterminé, nous disposons ainsi des « communalités estimées » pour chaque variable, que l'on peut confronter avec les communalités initiales pour mesurer la part de variance restituée de la variable dans le repère factoriel.

$$communalité\_est_j = \sum_{k=1}^{q} a_{jk}^2$$

```
#communalités estimées pour chaque variable sur les 2 premiers facteurs
estim_comm = numpy.sum(loadings**2,axis=1)

#affichage
print(pandas.DataFrame(estim_comm,index=D.columns))
```

```
              0
CYL    0.771910
PUISS  0.811365
LONG   0.858729
LARG   0.770085
POIDS  0.808908
VMAX   0.565872
RND1   0.224983
RND2   0.234415
RND3   0.176156
RND4   0.308066
RND5   0.543395
RND6   0.167107
```

SAS

Valeurs estimées finales des facteurs communs : Total = 6.240992

CYL	PUISS	LONG	LARG	POIDS	VMAX	RND1	RND2	RND3	RND4	RND5	RND6
0.77190958	0.81136470	0.85872919	0.77008512	0.80890830	0.56587163	0.22498350	0.23441523	0.17615562	0.30806588	0.54339549	0.16710745

Le total des « valeurs estimée finales des facteurs communs » caractérise la qualité de restitution du repère factoriel. Elle correspond : à l'addition des valeurs propres des facteurs concernés, soit $(5.2295207 + 1.0114710 = 6.240992)$; mais aussi à la somme des communalités estimées : $(0.7719 + 0.8113 + 0.8587 + \cdots + 0.1671 = 6.240992)$.

Pour en revenir aux variances restituées par variable, nous l'exprimons en ratio de la variance initiale. Il sera plus facile d'identifier les variables bien représentées dans le repère factoriel.

```
#pourcentage expl. par variables
pourcent = estim_comm / R2Det

#qualité de représentation par variables
print(pandas.DataFrame(pourcent,index=D.columns))
```

```
              0
CYL    0.915612
PUISS  0.879111
LONG   0.927452
LARG   0.875220
POIDS  0.875469
VMAX   0.641368
RND1   0.668221
RND2   0.594406
RND3   0.390724
RND4   0.504970
RND5   0.789756
RND6   0.330043
```

De manière assez incongrue, la variable RND5 est très bien représentée sur les (q = 2) premiers facteurs. Avant de tirer des conclusions hâtives, intéressons-nous aux contributions calculées à partir des fonctions scores.

2.6.3.7 Coefficients de la fonction score

Nous obtenons les coefficients de fonctions de projection en multipliant l'inverse la matrice des corrélations avec les « loadings ».

```
#factor scores
fscores = numpy.dot(invR,loadings)
print(pandas.DataFrame(fscores,index=D.columns))
```

```
              0         1
CYL    0.098976  0.257383
PUISS  0.200054  0.268495
LONG   0.217457 -0.737828
LARG   0.101000 -0.188740
POIDS  0.240592 -0.041291
VMAX   0.190469 -0.152789
RND1  -0.035438 -0.128374
RND2   0.054243  0.146876
RND3   0.027593  0.218099
RND4   0.068207  0.151815
RND5  -0.035918 -0.417231
RND6  -0.031482 -0.088133
```

Coefficients du score normalisés

		Factor1	Factor2
CYL	CYL	0.09898	-0.25738
PUISS	PUISS	0.20005	-0.26850
LONG	LONG	0.21746	0.73783
LARG	LARG	0.10100	0.18874
POIDS	POIDS	0.24059	0.04129
VMAX	VMAX	0.19047	0.15279
RND1	RND1	-0.03544	0.12837
RND2	RND2	0.05424	-0.14688
RND3	RND3	0.02759	-0.21810
RND4	RND4	0.06821	-0.15181
RND5	RND5	-0.03592	0.41723
RND6	RND6	-0.03148	0.08813

Nous utiliserons ces coefficients plus loin pour déterminer les coordonnées des individus dans le repère factoriel. Pour l'heure, nous nous intéressons à la contribution des variables dans la projection.

2.6.3.8 Contribution des variables au sens de la fonction score

Nous calculons tout d'abord la somme des carrées des coefficients des fonctions scores par facteur.

```
#somme des carrés des coefficients
contrib = numpy.sum(fscores**2,axis=0)
print(contrib)
```

```
[0.2134 1.0339]
```

Pour obtenir les contributions des variables aux facteurs, nous faisons le ratio entre le carré des coefficients et les sommes ci-dessus.

```
#contributions des variables par facteur
contrib_var = fscores**2 / contrib
print(pandas.DataFrame(contrib_var,index=D.columns))
```

```
              0         1
CYL    0.045914  0.064073
PUISS  0.187575  0.069725
LONG   0.221630  0.526533
LARG   0.047810  0.034454
POIDS  0.271297  0.001649
VMAX   0.170033  0.022579
RND1   0.005886  0.015939
RND2   0.013790  0.020865
RND3   0.003568  0.046007
RND4   0.021804  0.022292
RND5   0.006047  0.168372
RND6   0.004645  0.007513
```

Résultat très intéressant durant le calcul de cette forme de contribution, on se rend compte que les variables RND, au moins sur le 1er facteur, voient leur impact très amoindri lors du positionnement des individus, lequel est surtout déterminé par un mix d'encombrement (LONG, POIDS) et de performances (PUISS, VMAX).

2.6.3.9 Fidélité des facteurs

En théorie, la variance des facteurs vaut 1. Sauf que nous ne disposons pas des « vrais » coefficients définis sur la population, mais des estimations obtenues à partir d'un échantillon. La variance observée des coordonnées des individus sur les axes donne une indication sur leur fiabilité (des axes). Nous les obtenons en effectuant la somme du produit des coefficients de projection « factor scores » avec les « loadings ».

```
#fidelité des facteurs
fidelite = numpy.sum(fscores*loadings,axis=0)
print(fidelite)
```

```
[0.9761 0.7922]
```

Plusieurs corrélations au carré des variables avec chaque facteur	
Factor1	Factor2
0.97606980	0.79218070

SAS utilise l'appellation « carré du coefficient de corrélation multiple des variables avec chaque facteur ». En effet, cette variance correspond également au carré de la corrélation entre la variable latente théorique (définie sur la population) et son estimation par le facteur (obtenue sur l'échantillon). L'utiliser comme indicateur de crédibilité est dès lors tout à fait approprié. Une valeur élevée (≥ 0.7 selon certaines références) indique une bonne stabilité, le facteur reflète une réalité qui existe dans la population.

Pour les données « Autos bruitées », si nous sélectionnons les 8 premiers facteurs correspondant aux valeurs propres positives, nous obtenons aux indications suivantes :

Facteur	Fidelité
Axis_1	0.9761
Axis_2	0.7922
Axis_3	0.8541
Axis_4	0.8118
Axis_5	0.5759
Axis_6	0.5371
Axis_7	0.3701
Axis_8	0.0538

Avec ce nouveau critère, nous pencherons pour le choix de (q = 4) facteurs pour l'analyse en facteurs principaux.

2.6.3.10 Carte des individus

Reste à projeter maintenant des observations dans le repère factoriel pour situer la nouvelle appréciation des proximités entre les véhicules. Les coordonnées s'obtiennent par le produit des données centrées réduites par les coefficients des « factor scores ».

```
#moyennes des variables
moyennes = numpy.mean(D.values,axis=0)
print(moyennes)

[ 1.6317e+03  8.4611e+01  4.3350e+02  1.6667e+02  1.0788e+03  1.5828e+02
  2.0261e-01  4.2411e-01 -4.2917e-01 -2.9028e-01 -3.4911e-01  2.1383e-01]

#écarts-type des variables
stdev = numpy.std(D.values,axis=0)
print(stdev)


[363.3945  19.8022  21.4845   5.164  133.0991  11.7983   0.9126   0.5716
   0.7928   0.9949   0.9778   1.2347]


#données centrées et réduites – vérification des dimensions
Z = (D.values - moyennes)/stdev
print(Z.shape)
(18, 12)

#coordonnées des individus dans le plan
coord = numpy.dot(Z,fscores)
print(pandas.DataFrame(coord,index=D.index))

                         0         1
Modele
```

```
Alfasud TI      -0.986199  1.119263
Audi 100         0.523302 -2.582756
Simca 1300      -0.472979 -0.182853
Citroen GS Club -1.081071  0.098957
Fiat 132         0.368924  0.960006
Lancia Beta     -0.198631 -0.755777
Peugeot 504      0.183672 -0.602165
Renault 16 TL   -1.270892 -1.110566
Renault 30       2.096801  0.904822
Toyota Corolla  -1.787926  0.814933
Alfetta 1.66     0.069678 -0.654304
Princess 1800    0.454170  0.045963
Datsun 200L      1.499793  0.440714
Taunus 2000      0.659626  0.217548
Rancho          -0.211494  0.056303
Mazda 9295       0.299261  0.510391
Opel Rekord      1.038765 -0.153442
Lada 1300       -1.184801  0.872964
```

Nous affichons la carte des individus enfin (Figure 75).

```
#carte des individus
fig, ax = plt.subplots(figsize=(10,10))
ax.axis([-3,+3,-3,+3])
ax.plot([-3,+3],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-3,+3],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Carte des individus")

for i in range(D.shape[0]):
    ax.text(coord[i,0],coord[i,1],D.index[i])

plt.show()
```

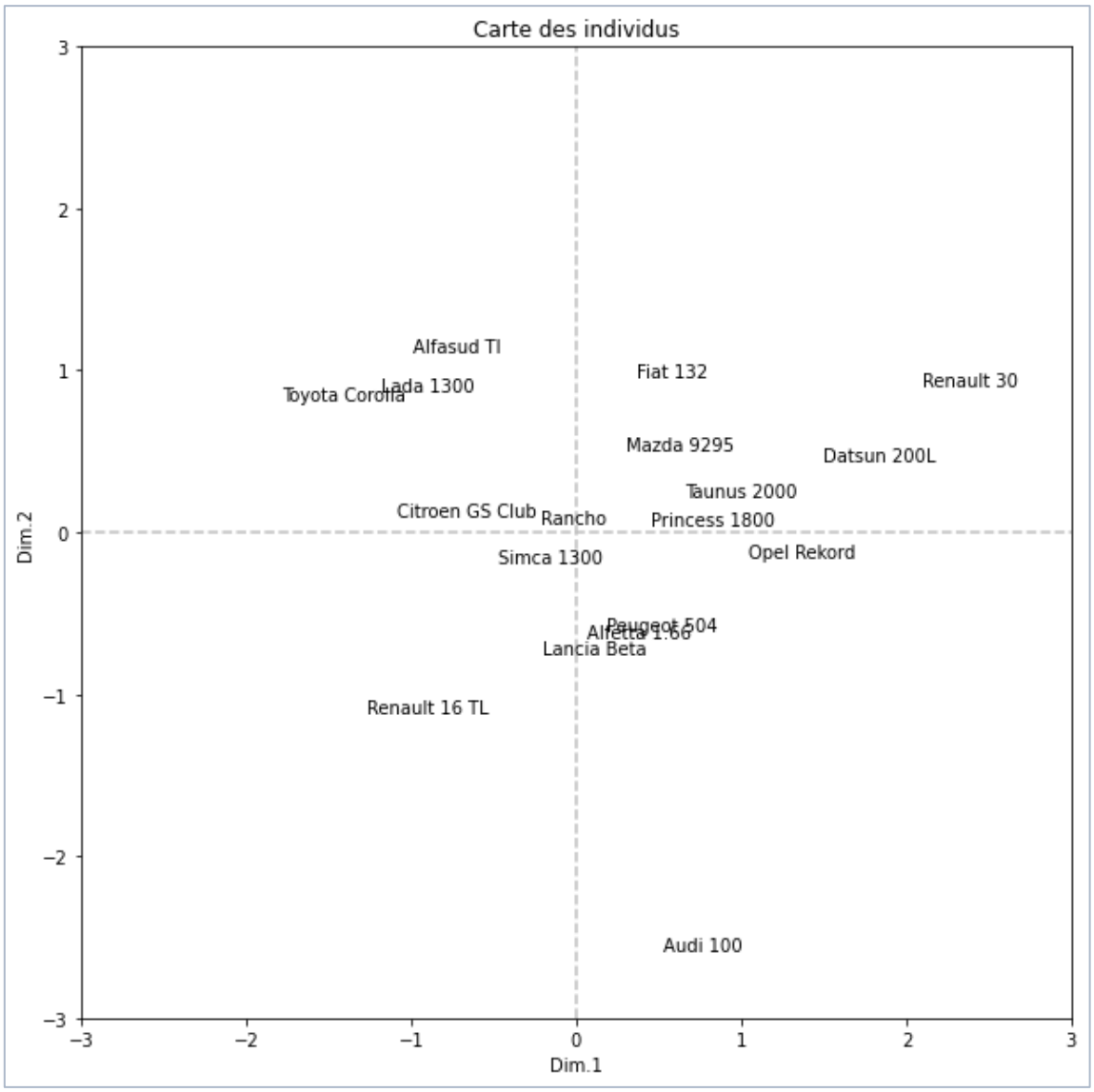


Figure 75 – Carte des individus – AFP – Données "Autos bruitées"

Même si les perspectives ont quelque peu changé par rapport à l'ACP (Figure 72), il reste que le premier plan factoriel est dominé par la dimension « encombrement ». La vraie différence ici le changement de position de l'Alfetta 1.66.

2.6.3.11 Une stratégie itérative pour l'analyse en facteurs principaux

Il existe une variante itérative de l'analyse en facteurs principaux censée fournir une meilleure qualité de projection. La procédure précédente sert de première étape. Ensuite, nous remplaçons la diagonale de la matrice R' par les communalités estimées. Nous relançons alors l'analyse. On procède ainsi jusqu'à convergence du système c.-à-d. jusqu'à les communalités soient stables (ex. SAS). On peut aussi fixer arbitrairement le nombre maximum d'itérations (Ex. SPSS).

Attention, il se peut que les communalités d'une des variables soit supérieure à 1 dans certaines situations. On parle de « problème de Heywood ». Cela indique clairement une incohérence dans le processus. Les causes sont diverses, entres autres parce qu'il est possible que l'on ait sélectionné un nombre inapproprié de facteurs.

2.6.4 L'analyse de Harris

Mon intérêt pour l'analyse en facteurs principaux tient surtout à l'idée de l'exploitation de la variabilité communes aux données, avec l'espoir de disposer d'un outil capable d'évacuer les variables non pertinentes qui n'ont aucun rapport avec le sujet de l'étude. Je pensais que l'AFP pourrait dépasser cet écueil, et j'avais monté plusieurs expérimentations sur le même modèle que ci-dessus (le TUTO 9 rapporte les résultats sur le dataset « Bières ») pour étudier le comportement de l'approche lorsqu'on lui soumet des bases bruitées, au sens où l'on rajoute plusieurs variables n'ayant aucun rapport avec le sujet étudié. Au final, nous disposons certes d'un point de vue différent sur les données par rapport à l'ACP, mais pour ce qui est de la capacité à neutraliser les variables inopportunes, j'étais assez déçu.

En continuant mes investigations (j'ai quand-même passé pas mal de temps sur la documentation de SAS pour être honnête), mon attention a été attirée par l'analyse de Harris. Elle procède de la même logique que l'analyse en facteur principaux. Elle cherche à décomposer l'information partagée par les variables en travaillant à partir d'une version modifiée de la matrice de corrélations. Ici, l'idée consiste à exacerber les corrélations entre deux variables si elles (l'une des deux ou les deux simultanément) sont fortement liées aux autres.

Une nouvelle notion, « uniqueness » (u_j), est mise en avant. Elle correspond à la part de variance (résiduelle) de X_j non-expliquée par les (p-1) autres variables. Elle est définie comme suit :

$$u_j = 1 - \mathcal{R}_j^2$$

La nouvelle matrice R'' à diagonaliser est alors déduite de R' avec la formule suivante :

$$r''_{lj} = \frac{r'_{lj}}{\sqrt{u_l \times u_j}}$$

Pour les données « Autos bruitées », voici la matrice qui sera diagonalisée avec la méthode de Harris (Figure 76). Nous remarquerons que les corrélations relatives aux variables originelles de la base (CYL, ..., VMAX) sont très fortement accentuées avec cette nouvelle transformation.

	CYL	PUISS	LONG	LARG	POIDS	V_MA	RND1	RND2	RND3	RND4	RND5	RND6
CYL	5.35	7.24	6.50	4.59	7.22	4.89	-0.88	0.76	-0.20	2.29	-2.48	-0.45
PUISS	7.24	11.94	8.49	5.41	10.00	8.87	-1.67	1.61	-0.53	2.49	-3.72	0.37
LONG	6.50	8.49	12.55	9.00	11.57	5.10	-1.30	1.03	-0.07	1.82	-3.17	1.62
LARG	4.59	5.41	9.00	7.33	7.50	3.98	-0.67	0.91	-0.75	2.23	-1.35	0.46
POIDS	7.22	10.00	11.57	7.50	12.10	5.05	-1.58	1.28	-0.06	2.05	-4.02	0.48
V_MAX	4.89	8.87	5.10	3.98	5.05	7.48	-1.12	1.00	-0.76	1.75	-2.37	0.46
RND1	-0.88	-1.67	-1.30	-0.67	-1.58	-1.12	0.51	-0.70	0.03	-0.35	0.82	-0.04
RND2	0.76	1.61	1.03	0.91	1.28	1.00	-0.70	0.64	0.14	0.88	-0.57	0.02
RND3	-0.20	-0.53	-0.07	-0.75	-0.06	-0.76	0.03	0.14	0.82	-0.06	-0.93	-0.04
RND4	2.29	2.49	1.82	2.23	2.05	1.75	-0.35	0.88	-0.06	1.56	-0.40	-0.55
RND5	-2.48	-3.72	-3.17	-1.35	-4.02	-2.37	0.82	-0.57	-0.93	-0.40	2.21	-0.04
RND6	-0.45	0.37	1.62	0.46	0.48	0.46	-0.04	0.02	-0.04	-0.55	-0.04	1.03

Figure 76 – Matrice à diagonaliser – Méthode de Harris – Données "Autos bruitées"

Une fois cette matrice définie, tout le dispositif subséquent est identique à celui de l'analyse en facteurs principaux. Dans ce qui suit, nous formons R'' sous Python puis nous la diagonalisons pour obtenir le jeu des valeurs propres. Nous poursuivrons ensuite directement avec les résultats de TANAGRA. **Remarque :** Sous SAS, nous aurions paramétré la PROC FACTOR de SAS avec l'option (METHOD = HARRIS) pour spécifier la technique à utiliser.

```
PROC FACTOR DATA = AUTOS_NOISY
       METHOD = HARRIS
       NFACTORS = 2
       SCORE
       PLOTS = ALL;
       VAR CYL PUISS LONG LARG POIDS VMAX RND1 RND2 RND3 RND4 RND5 RND6;
RUN;
```

Nous calculons tout d'abord les « uniqueness » des variables. Puis nous formons R'' avec une double boucle, très scolaire, mais qui a le mérite de la clarté.

```
#uniqueness par variable
u = 1 - R2Det

#matrice R'' pour Analyse de Harris
Rsecond = Rprim.copy()

#double boucle pour faire simple
for l in range(p):
```

```
    for j in range(p):
        Rsecond[l,j] = Rprim[l,j]/numpy.sqrt(u[l]*u[j])

#affichage
numpy.set_printoptions(precision=2)
print(Rsecond)
```

```
[[ 5.37  7.24  6.5   4.59  7.22  4.89 -0.88  0.76 -0.2   2.29 -2.48 -0.45]
 [ 7.24 11.98  8.49  5.41 10.    8.87 -1.67  1.61 -0.53  2.49 -3.72  0.37]
 [ 6.5   8.49 12.5   9.   11.57  5.1  -1.3   1.03 -0.07  1.82 -3.17  1.62]
 [ 4.59  5.41  9.    7.32  7.5   3.98 -0.67  0.91 -0.75  2.23 -1.35  0.46]
 [ 7.22 10.   11.57  7.5  12.15  5.05 -1.58  1.28 -0.06  2.05 -4.02  0.48]
 [ 4.89  8.87  5.1   3.98  5.05  7.5  -1.12  1.   -0.76  1.75 -2.37  0.46]
 [-0.88 -1.67 -1.3  -0.67 -1.58 -1.12  0.51 -0.7   0.03 -0.35  0.82 -0.04]
 [ 0.76  1.61  1.03  0.91  1.28  1.   -0.7   0.65  0.14  0.88 -0.57  0.02]
 [-0.2  -0.53 -0.07 -0.75 -0.06 -0.76  0.03  0.14  0.82 -0.06 -0.93 -0.04]
 [ 2.29  2.49  1.82  2.23  2.05  1.75 -0.35  0.88 -0.06  1.56 -0.4  -0.55]
 [-2.48 -3.72 -3.17 -1.35 -4.02 -2.37  0.82 -0.57 -0.93 -0.4   2.21 -0.04]
 [-0.45  0.37  1.62  0.46  0.48  0.46 -0.04  0.02 -0.04 -0.55 -0.04  1.03]]
```

Il ne nous reste plus qu'à diagonaliser R''.

```
#diagonalisation de la matrice Rsecond
sol2 = numpy.linalg.eigh(Rsecond)

#valeurs propres - ordre décroissant
numpy.set_printoptions(precision=2,suppress=True)
print(numpy.flip(sol2[0]))
```

```
[48.36  8.07  3.73  2.69  1.31  0.97  0.36  0.04 -0.28 -0.37 -0.58 -0.70]
```

Eboulis des valeurs propres. Il y a toujours des valeurs propres négatives puisque nous nous intéressons uniquement à la variabilité partagée. Mais la nouveauté ici, par rapport à l'AFP, est que l'éboulis des valeurs propres retrouve cette forme concave bien sympathique.

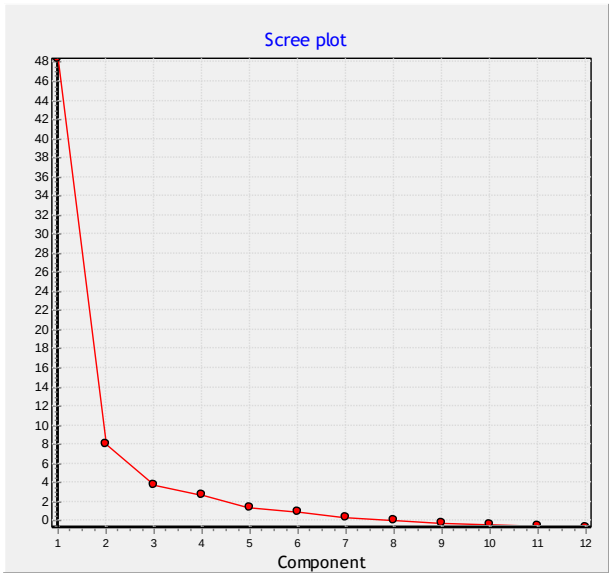


Figure 77 – Méthode de Harris – Eboulis des valeurs propres – Données "Autos bruitées"

Loadings. Si l'on s'en tient aux (q = 2) premiers facteurs. L'analyse restitue 72% de l'information disponible (8.3647 = somme des carrés des communalités initiales vs. 6.03354 = somme des carrés des communalités estimées dans le repère factoriel). VMAX et PUISS semblent à nouveau jouer un rôle important sur le 2nd facteur, même s'il restent corrélés toujours avec le 1er facteur.

Factor Loadings [Communality Estimates]

Attribute	Communality Estimates		Axis_1		Axis_2	
-	Prior	Final	Corr.	Sq. (Cumul.)	Corr.	Sq. (Cumul.)
POIDS	0.92397	0.87330	0.91588	*0.84 (0.84)*	0.18563	*0.03 (0.87)*
LONG	0.92590	0.92109	0.88809	*0.79 (0.79)*	0.36384	*0.13 (0.92)*
PUISS	0.92294	0.92234	0.87969	*0.77 (0.77)*	-0.38533	*0.15 (0.92)*
CYL	0.84305	0.76902	0.86673	*0.75 (0.75)*	-0.13341	*0.02 (0.77)*
LARG	0.87988	0.75899	0.79279	*0.63 (0.63)*	0.36122	*0.13 (0.76)*
V.MAX	0.88229	0.81191	0.72424	*0.52 (0.52)*	-0.53609	*0.29 (0.81)*
RND5	0.68805	0.38490	-0.60269	*0.36 (0.36)*	0.14717	*0.02 (0.38)*
RND6	0.50632	0.05529	0.12942	*0.02 (0.02)*	0.19632	*0.04 (0.06)*
RND1	0.33669	0.15944	-0.37559	*0.14 (0.14)*	0.13554	*0.02 (0.16)*
RND2	0.39437	0.12657	0.32900	*0.11 (0.11)*	-0.13539	*0.02 (0.13)*
RND4	0.61007	0.23796	0.47121	*0.22 (0.22)*	-0.12616	*0.02 (0.24)*
RND3	0.45084	0.01274	-0.07587	*0.01 (0.01)*	0.08354	*0.01 (0.01)*
Unweighted Var. Expl.	8.36437	6.03354	5.16276	*62 % (62 %)*	0.87078	*10 % (72 %)*

Figure 78 – Méthode de Harris – Tableau des "loadings" – Données "Autos bruitées"

Cercle des « corrélations ». Le cercle des corrélations (Figure 79) (cercle des loadings en vrai, nommée « Représentation de facteur initiale » sous SAS) ressemble à s'y méprendre à celui de l'ACP sur exclusivement les variables initiales (section 1.3.2). L'influence néfaste des RND est annihilé, au moins dans le premier plan factoriel.

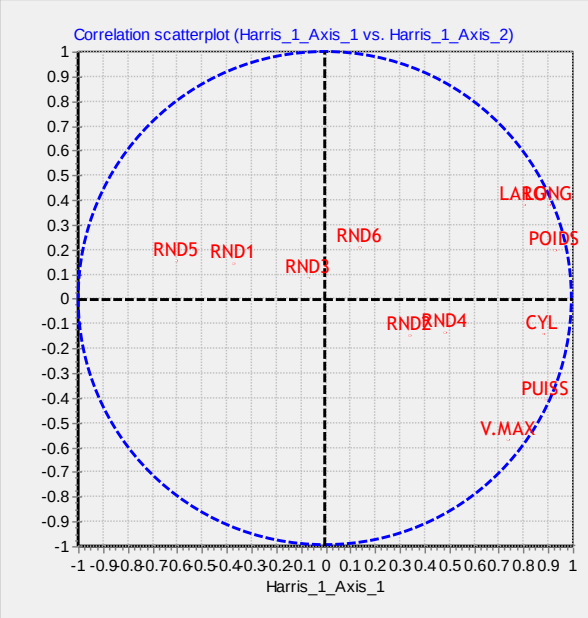


Figure 79 – Méthode de Harris – Cercle des « loadings » – Données "Autos bruitées"

Factor score coefficients. Nous affichons les coefficients des fonctions scores pour le premier plan factoriel en faisant figurer les contributions des variables au positionnement des individus. (PUISS, LONG, POIDS) sont les plus influentes sur le 1er facteur, (PUISS, LONG, VMAX) sur le second. Les 2 facteurs présentent tous deux une forte crédibilité, proche de 1.

Factor Score Coefficients

Squared Multiple Corr. of the Variables with Each Factor			0.9797421		0.8897044	
Attribute	Mean	Std-dev	Axis_1	CTR	Axis_2	CTR
CYL	1631.667	363.394	0.111873	0.06	-0.093756	0.01
PUISS	84.611	19.802	0.231250	0.24	-0.551502	0.29
LONG	433.500	21.484	0.242798	0.27	0.541575	0.28
LARG	166.667	5.164	0.133697	0.08	0.331664	0.10
POIDS	1078.833	133.099	0.244036	0.27	0.269300	0.07
VMAX	158.278	11.798	0.124639	0.07	-0.502310	0.24
RND1	0.203	0.913	-0.011471	0.00	0.022537	0.00
RND2	0.424	0.572	0.011005	0.00	-0.024656	0.00
RND3	-0.429	0.793	-0.002799	0.00	0.016780	0.00
RND4	-0.290	0.995	0.024481	0.00	-0.035684	0.00
RND5	-0.349	0.978	-0.039139	0.01	0.052035	0.00
RND6	0.214	1.235	0.005311	0.00	0.043861	0.00

Figure 80 – Méthode de Harris – Coefficients des fonctions scores – Données "Autos bruitées"

Carte des individus. Ces coefficients appliquées sur les données centrées et réduites nous permettent de retrouver les coordonnées factorielles des individus (Figure 81).

Quelques véhicules « emblématiques » retrouvent leur place, en particulier l'Alfasud TI, l'Alfetta 166,.... La carte des individus reflète les proximités initiales que nous avons identifiées sur la base « Autos » non-polluée[1] (section 1.3.3).

[1] Si on peut dire, pour des voitures qui roulaient à l'essence plombée avec des carburateurs gavés par des pompes de reprise. Mais la sonorité de certaines était diabolique, mes oreilles s'en souviennent encore.

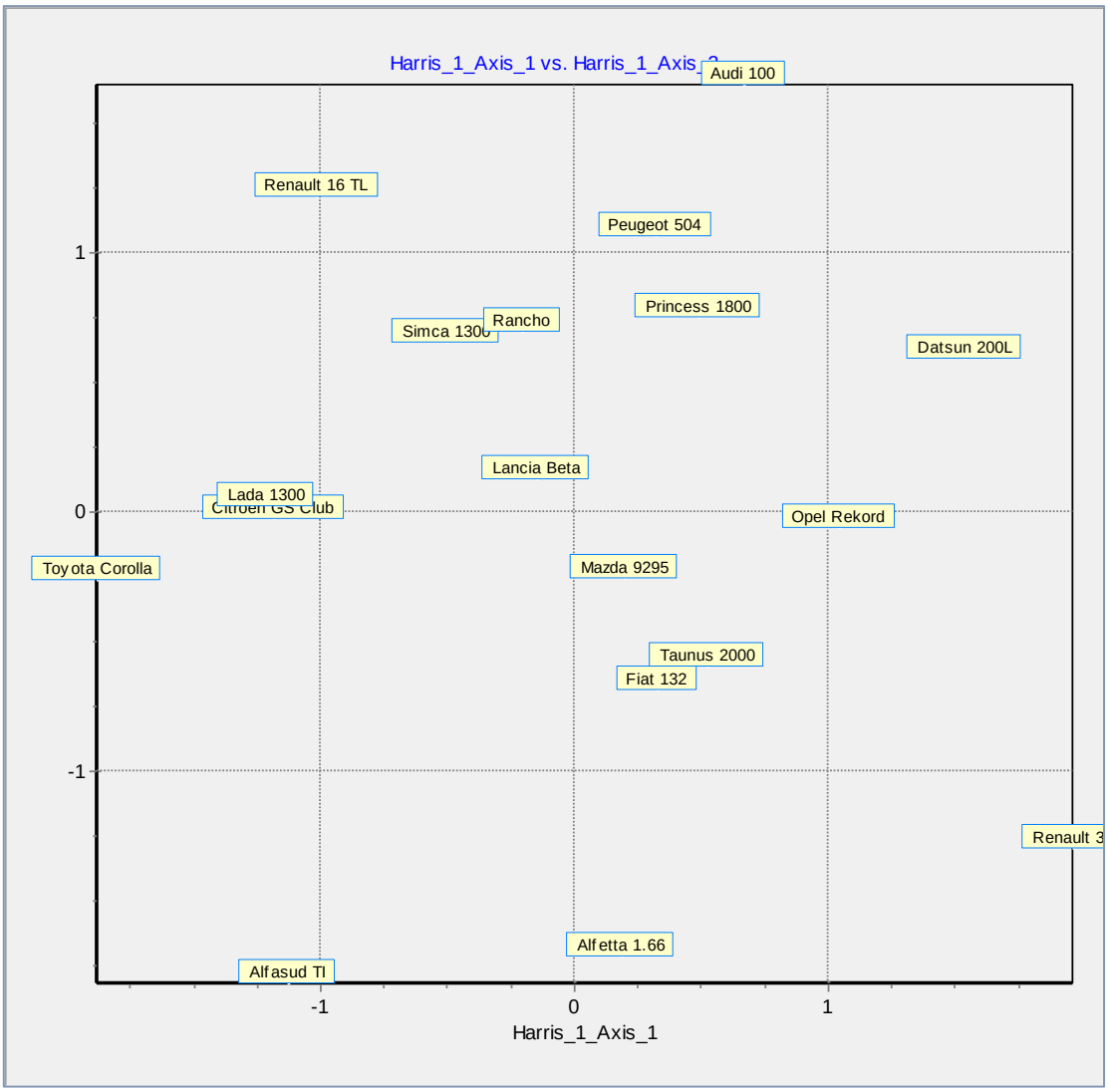


Figure 81 - Méthode de Harris - Carte des individus - Données "Autos bruitées"

3 Positionnement multidimensionnel (MDS)

Le positionnement multidimensionnel (multidimensional scaling en anglais, MDS) est une technique de représentation qui permet de visualiser dans un repère euclidien (à q = 2 ou 3 dimensions le plus souvent) les positions relatives d'objets décrits par une matrice de distances, de dissimilarités, ou de similarités. Curieusement, la technique est peu décrite dans la littérature francophone. En cherchant un peu, on la retrouve néanmoins dans les ouvrages qui font référence. On parle plus volontiers d'analyse factorielle sur tableau de distances et de dissimilarités (Saporta, 2006, section 7.5 ; Diday et al., 1982, section 2.3). Le MDS peut avoir plusieurs finalités (voir Borg I., Groenen P.J., « Modern Multidimensional Scaling – Theory and Applications », 2nd Edition, Springer, 2005). Nous nous focaliserons sur son utilisation en tant que méthode exploratoire dans ce chapitre.

3.1 Position du problème

Parfois les données s'expriment intrinsèquement sous la forme de similarités ou dissimilarités entre objets. Cela peut être les proximités entre individus dans un réseau social, des distances entre les villes d'un pays, des associations entre produits exprimées par des notes composites attribuées par un panel de consommateurs, etc. L'objectif le plus évident du positionnement multidimensionnel est de rendre compte dans une représentation graphique des proximités dans un espace de dimension réduite en disposant d'un critère de qualité de l'approximation.

Mais, au-delà de ça, il permet d'aller plus loin dans notre démarche exploratoire. Il nous aide déjà à comprendre les principales structures qui régissent les données en révélant les oppositions

et proximités. Si nous disposons des connaissances sur les propriétés des objets par exemple, leurs positions relatives dans le nouveau repère permettent d'interpréter les « dimensions » qu'il met en évidence. En exprimant les données sous la forme d'un tableau individus – variables (qui sont les dimensions), il ouvre la porte à l'utilisation de techniques de machine learning inapplicables directement sur des matrices de distance.

Pour illustrer notre propos, nous reprenons un variante des données « Autos » de notre ouvrage de référence (Saporta, 2006, page 428). Nous restreignons la base aux véhicules étrangers. Les distances entre paires de véhicules sont représentées dans le tableau ci-dessous (Figure 82). Les objets proches (distance faible) sont sur fond bleu, éloignes sur fond rouge.

	Toyota Corolla	Lada 1300	Alfasud TI	Lancia Beta	Mazda 9295	Fiat 132	Alfetta 1.66	Princess 1800	Audi 100	Taunus 2000	Opel Rekord	Datsun 200L
Toyota Corolla												
Lada 1300	190.2											
Alfasud TI	195.3	106.0										
Lancia Beta	299.3	130.1	219.8									
Mazda 9295	667.2	497.4	477.9	472.4								
Fiat 132	513.6	331.6	336.1	289.8	184.9							
Alfetta 1.66	477.8	301.1	294.5	275.6	204.3	51.0						
Princess 1800	722.4	546.2	536.4	507.6	72.0	220.9	251.8					
Audi 100	521.1	339.9	346.7	295.3	184.2	35.2	73.8	217.2				
Taunus 2000	870.9	712.1	678.2	696.3	225.1	408.8	423.9	211.7	407.5			
Opel Rekord	872.3	708.3	680.5	684.2	213.3	395.0	414.8	187.4	391.8	47.8		
Datsun 200L	1004.6	821.4	822.8	760.4	360.5	492.0	530.4	292.9	486.5	292.3	251.7	

Figure 82 - Données "Autos-MDS"

Si on connaît un peu les voitures des années 70, on comprend aisément qu'une « Audi 100 » soit proche d'une « Fiat 132 », tous deux se positionnent sur le segment des berlines familiales. Etc.

Mais on se rend compte également que l'exploration approfondie de ce type de matrice en l'état se révèlera très rapidement inextricable à mesure que le nombre d'objets augmente. Le passage par une représentation graphique est indispensable pour disposer d'une vision globale des données, en espérant que l'information initiale soit préservée.

Pour la base « Autos-MDS », voici le nuage de points dans un repère à (q = 2) dimensions issu du positionnement multidimensionnel. Nous retrouvons plus facilement les rapprochements (fond bleu) et oppositions entre les véhicules (rouge) (Figure 83). Avec un peu d'expertise, nous

constatons également que l'axe horizontal les distingue selon leur segment, d'ouest en est : « grosses » (cylindrée, encombrement...) vs. « petites » voitures.

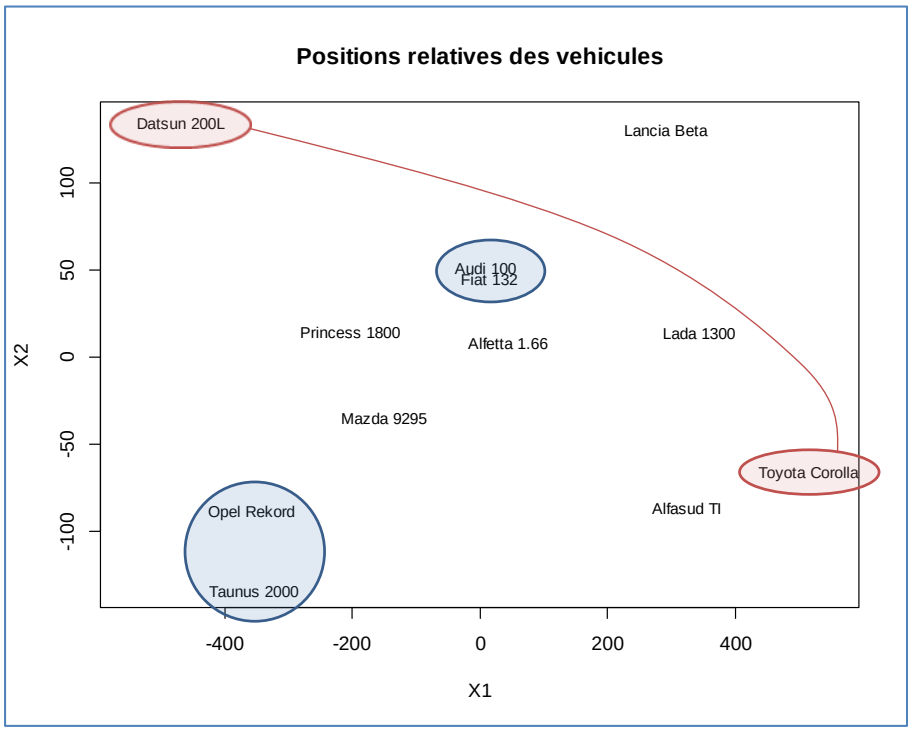


Figure 83 - Données "Autos-MDS" - Représentation dans le plan

Remarque : La matrice des similarités / dissimilarités (Figure 82) est l'unique source d'information habituellement en MDS. Mais comme nous sommes dans une démarche pédagogique, voici le tableau initial utilisé pour le calcul les distances entre paires de véhicules, trié de manière décroissante selon la cylindrée (Figure 84). Nous comprenons aisément le positionnement relatif des véhicules dans la représentation graphique.

Modele	CYL	PUISS	LONG	LARG	POID	VMA
Datsun 200L	1998	115	469	169	1370	160
Taunus 2000	1993	98	438	170	1080	167
Opel Rekord	1979	100	459	173	1120	173
Princess 1800	1798	82	445	172	1160	158
Mazda 9295	1769	83	440	165	1095	165
Audi 100	1588	85	468	177	1110	160
Fiat 132	1585	98	439	164	1105	165
Alfetta 1.66	1570	109	428	162	1060	175
Alfasud TI	1350	79	393	161	870	165
Lancia Beta	1297	82	429	169	1080	160
Lada 1300	1294	68	404	161	955	140
Toyota Corolla	1166	55	399	157	815	140

Figure 84 - Données "Autos-MDS" - Tableau source

Notre propos dans ce chapitre est de montrer comment partir de la matrice des dissimilarités (Figure 82) pour parvenir à une représentation dans un espace à « q » dimensions (Figure 83) à l'aide du positionnement multidimensionnel. Une approche possible pour apprécier la qualité des résultats sera alors de confronter les distances initiales ($\delta_{ii'}$) avec celles reproduites dans le nouveau repère ($\hat{\delta}_{ii'}$). La restitution sera parfaite si ($\delta_{ii'} = \hat{\delta}_{ii'}$).

3.2 Positionnement multidimensionnel classique

3.2.1 Matrice de distances euclidiennes

Le positionnement multidimensionnel classique part du principe que la matrice source est constituée de distances euclidiennes. On parle aussi d'analyse factorielle sur tableau de distances ou de Principal Coordinates Analysis (PCoA). Un petit rappel sur les notions de dissimilarités et distances nous permet de délimiter le champ d'application de la méthode.

On parle de dissimilarité lorsqu'une mesure de proximité (δ) entre deux observations (i) et (i') répond aux propriétés suivantes :

- $(\delta_{ii} = 0)$, la dissimilarité d'un objet avec lui-même est nulle.
- $(\delta_{ii'} \geq 0\ ;\ \forall\ i \neq i')$, la mesure est positive ou nulle.
- $(\delta_{ii'} = \delta_{i'i})$, elle est symétrique.

Elle devient une distance si elle remplit la condition d'inégalité triangulaire, c.-à-d.

- $(\delta_{ii'} \leq \delta_{ii''} + \delta_{i'i''})$, il y a égalité si et seulement si les 3 points sont alignés dans l'espace de représentation.

Enfin, on parle de distance euclidienne si elle (la distance) peut être exprimée à travers un produit scalaire de l'écart entre les vecteurs de description des objets :

- $\delta^2_{ii'} = \langle x_i - x_{i'}, x_i - x_{i'} \rangle$

La technique présentée ici s'applique à cette dernière configuration. Mais nous verrons plus loin qu'elle peut s'appliquer aux autres situations moyennant des menus aménagements.

3.2.2 Transformation des données – Formule de Torgerson

L'objectif est de projeter les individus dans un espace factoriel X (x_{ij}) avec i = 1, ..., n (nombre d'observations) et j = 1, ..., q (dimension que l'on spécifie, souvent q = 2 ou 3). Nous utilisons le terme « factoriel » par commodité. Il n'est pas très judicieux dans notre contexte puisque les nouvelles variables de description ne sont en rien des combinaisons linéaires de variables initiales, qui ne sont pas censées exister, dont nous n'avons pas accès en tous les cas.

Ces facteurs sont centrés c.-à-d.

$$\frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0\ ;\ \forall\, j = 1, \dots, q$$

Le positionnement multidimensionnel classique consiste à déterminer les coordonnées des individus (x_{ij}) de manière à minimiser la quantité :

$$\sum_{i,i'} (\langle x_i, x_{i'} \rangle - b_{ii'})^2$$

Où B = ($b_{ii'}$), de taille (n x n), est la matrice des produits scalaires déduite de la matrice des distances via la formule de Torgerson (Saporta, 2006, page 182 ; Diday et al., 1982, page 212) :

$$b_{ii'} = -\frac{1}{2}\left(\delta^2_{ii'} - \delta^2_{i.} - \delta^2_{.i} + \delta^2_{..}\right)$$

Avec :

- $\delta^2_{i.} = \frac{1}{n}\sum_{i'=1}^{n} \delta^2_{ii'}$
- $\delta^2_{.i'} = \frac{1}{n}\sum_{i=1}^{n} \delta^2_{ii'}$
- $\delta^2_{..} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i'=1}^{n} \delta^2_{ii'}$

L'objectif du MDS classique consiste donc à approximer dans l'espace (X) les produits scalaires déduits des distances (B) en minimisant la somme des écarts au carré. Cela passe par la diagonalisation de la matrice B ci-dessus.

Traitements sous Python. Fidèle à notre démarche, nous tentons de reproduire les calculs sous Python. Nous importons et visualisons la matrice des distances.

```
#chargement - index_col = 0 pour indiquer que la colonne n°0 est un label
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="AUTOS_MDS",index_col=0)
print(D)

#effectifs
n = D.shape[0]
```

```
                Toyota Corolla   Lada 1300  Alfasud TI  Lancia Beta  \
Autos
Toyota Corolla        0.000000  190.247208  195.279287   299.264097
Lada 1300           190.247208    0.000000  105.962257   130.073056
Alfasud TI          195.279287  105.962257    0.000000   219.779435
Lancia Beta         299.264097  130.073056  219.779435     0.000000
Mazda 9295          667.205366  497.380136  477.940373   472.410838
Fiat 132            513.599065  331.572013  336.059519   289.784403
Alfetta 1.66        477.753074  301.137842  294.492784   275.559431
Princess 1800       722.387015  546.226144  536.364615   507.612057
Audi 100            521.123786  339.855852  346.678525   295.254128
Taunus 2000         870.874273  712.103925  678.247742   696.277962
Opel Rekord         872.332505  708.330431  680.548308   684.202455
Datsun 200L        1004.635755  821.424981  822.839596   760.388059

                Mazda 9295    Fiat 132  Alfetta 1.66  Princess 1800  \
Autos
Toyota Corolla  667.205366  513.599065    477.753074     722.387015
Lada 1300       497.380136  331.572013    301.137842     546.226144
Alfasud TI      477.940373  336.059519    294.492784     536.364615
Lancia Beta     472.410838  289.784403    275.559431     507.612057
Mazda 9295        0.000000  184.886452    204.340402      72.041655
Fiat 132        184.886452    0.000000     50.950957     220.904957
Alfetta 1.66    204.340402   50.950957      0.000000     251.775694
Princess 1800    72.041655  220.904957    251.775694       0.000000
Audi 100        184.236261   35.185224     73.824115     217.179649
Taunus 2000     225.075543  408.815362    423.884418     211.695536
Opel Rekord     213.314322  394.981012    414.786692     187.368621
Datsun 200L     360.516296  491.968495    530.353656     292.878815

                  Audi 100  Taunus 2000  Opel Rekord  Datsun 200L
Autos
Toyota Corolla  521.123786   870.874273   872.332505  1004.635755
Lada 1300       339.855852   712.103925   708.330431   821.424981
Alfasud TI      346.678525   678.247742   680.548308   822.839596
Lancia Beta     295.254128   696.277962   684.202455   760.388059
Mazda 9295      184.236261   225.075543   213.314322   360.516296
Fiat 132         35.185224   408.815362   394.981012   491.968495
Alfetta 1.66     73.824115   423.884418   414.786692   530.353656
Princess 1800   217.179649   211.695536   187.368621   292.878815
Audi 100          0.000000   407.543863   391.755025   486.482271
Taunus 2000     407.543863     0.000000    47.812132   292.275555
Opel Rekord     391.755025    47.812132     0.000000   251.735973
Datsun 200L     486.482271   292.275555   251.735973     0.000000
```

```
#seaborn
import seaborn as sns
```

```
#heatmap des distances
sns.heatmap(D,cmap='coolwarm',linewidth=0.5)
```

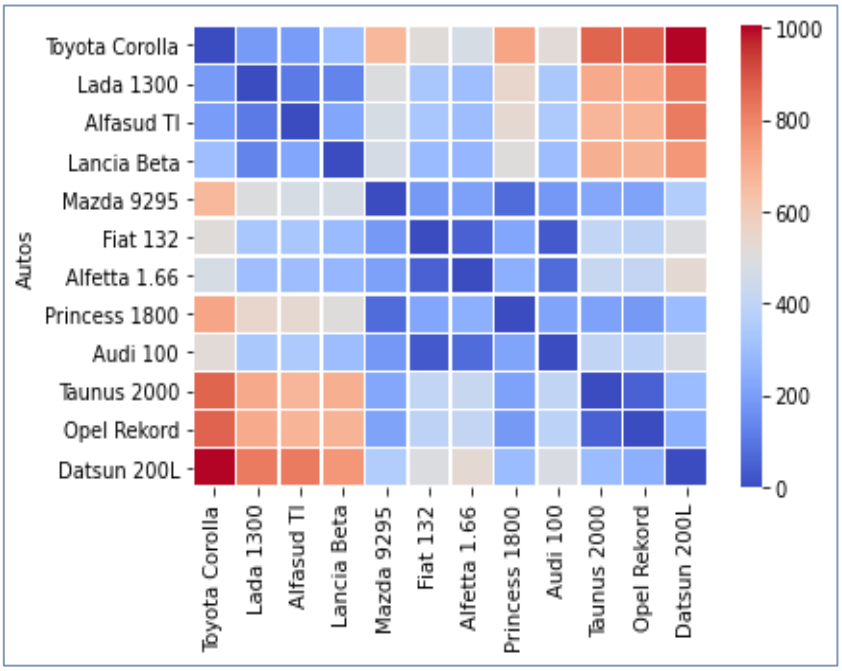


Figure 85 – "Heatmap" des distances entre paires d'individus – Données "Autos-MDS"

Nous calculons facilement la matrice B que nous devrons diagonaliser par la suite.

```
#librairy numpy
import numpy

#matrice des distances au format numpy
delta = D.values

#calcul des différentes sommes -- delta_point
d1 = numpy.sum(delta**2,axis=0)/n
d2 = numpy.sum(delta**2,axis=1)/n
d3 = numpy.sum(delta**2)/(n**2)

#construction de la matrice B
B = delta**2

for i in range(n):
    for iprim in range(n):
        B[i,iprim] = B[i,iprim] - d1[i] - d2[iprim] + d3

#
B = -0.5 *B

#matrice B à diagonaliser
numpy.set_printoptions(precision=2,suppress=True)
print(B)
```

```
[[ 269144.10  175489.01  171355.60  140642.01  -75846.57    3895.85
    21878.85 -105417.57     662.68 -173264.90 -178062.99 -250476.07]
 [ 175489.01  118027.93  109250.51  101403.93  -52516.65    5259.76
```

```
   15102.76  -69235.65    3138.60 -123157.99 -124005.07 -158757.15]
 [ 171355.60  109250.51  111701.10   82548.51  -46200.07     598.35
   13918.35  -67061.07   -2366.82 -102785.40 -107875.49 -163083.57]
 [ 140642.01  101403.93   82548.51  101698.93  -48573.65   10077.76
   14313.76  -57053.65    9137.60 -120177.99 -115370.07 -118647.15]
 [ -75846.57  -52516.65  -46200.07  -48573.65   24325.76   -3712.82
   -7283.82   30499.76   -2932.99   58207.43   57258.35   66775.26]
 [   3895.85    5259.76     598.35   10077.76   -3712.82    2431.60
    1348.60   -2251.82    2472.43  -10975.15   -8942.24    -202.32]
 [  21878.85   15102.76   13918.35   14313.76   -7283.82    1348.60
    2861.60   -9332.82     581.43  -17034.15  -16746.24  -19608.32]
 [-105417.57  -69235.65  -67061.07  -57053.65   30499.76   -2251.82
   -9332.82   41863.76    -775.99   69898.43   71225.35   97641.26]
 [    662.68    3138.60   -2366.82    9137.60   -2932.99    2472.43
     581.43    -775.99    3751.26   -9796.32   -7013.40    3141.51]
 [-173264.90 -123157.99 -102785.40 -120177.99   58207.43  -10975.15
  -17034.15   69898.43   -9796.32  142748.10  138078.01  148259.93]
 [-178062.99 -124005.07 -107875.49 -115370.07   57258.35   -8942.24
  -16746.24   71225.35   -7013.40  138078.01  135693.93  155759.85]
 [-250476.07 -158757.15 -163083.57 -118647.15   66775.26    -202.32
  -19608.32   97641.26    3141.51  148259.93  155759.85  239196.76]]
```

3.2.3 Diagonalisation

La solution du problème de minimisation ci-dessus passe par la diagonalisation de la matrice B.

Nous obtenons à la sortie :

- Les valeurs propres (λ_j), j = 1, ..., q. Dans le cas des distances euclidiennes, B est semi-définie positive, les valeurs propres sont positives ou nulles ($\lambda_j \geq 0, \forall j$).
- Les vecteurs propres (v_j), j = 1, ..., q ; chacun étant de taille (n).

Les coordonnées de l'individu n°i sur la dimension n°j est obtenue avec

$$x_{ij} = \sqrt{\lambda_j} \times v_{ij}$$

Traitements sous Python. Nous faisons appel aux routines de diagonalisation de la librairie « Numpy » pour traiter la matrice B.

```
#diagonalisation de B
sol = numpy.linalg.eig(B)

#valeurs propres
numpy.set_printoptions(precision=5,suppress=True)
print(sol[0])
```

```
[1110687.72015   79216.45647    1823.00709    1440.37536     212.74987
      64.52439      -0.00000       0.00000       0.00000       0.00000
      -0.00000      -0.00000]
```

Elles sont toutes positives. Celles qui semblent négatives sont simplement dues à la précision des calculs.

Voici les vecteurs propres correspondants.

```
#vecteurs propres
numpy.set_printoptions(precision=2,suppress=True)
print(sol[1])
```

```
[[-0.49 -0.23 -0.33  0.08  0.37  0.30 -0.42  0.32 -0.03 -0.13 -0.27 -0.01]
 [-0.33  0.05 -0.16  0.42  0.24 -0.28  0.55 -0.29  0.26 -0.21 -0.24 -0.08]
 [-0.31 -0.30  0.37  0.07 -0.30 -0.18 -0.23 -0.43 -0.45 -0.23 -0.26 -0.05]
 [-0.28  0.47  0.08 -0.10 -0.45 -0.20 -0.20  0.19  0.35  0.23 -0.36  0.03]
 [ 0.14 -0.12 -0.13  0.09 -0.36  0.68  0.14 -0.28  0.17  0.04 -0.23 -0.40]
 [-0.01  0.16  0.21 -0.12  0.17  0.26  0.47  0.20 -0.51  0.13 -0.51  0.24]
 [-0.04  0.03  0.62 -0.21  0.19  0.12  0.09  0.28  0.21 -0.48 -0.06 -0.38]
 [ 0.19  0.05 -0.29  0.26 -0.43 -0.10  0.13  0.47 -0.32 -0.59 -0.17 -0.03]
 [-0.01  0.19 -0.42 -0.66  0.12 -0.21  0.04 -0.20 -0.24 -0.18 -0.19 -0.42]
 [ 0.34 -0.47  0.04  0.13  0.09 -0.39 -0.02  0.28  0.01  0.32 -0.33 -0.40]
 [ 0.34 -0.31 -0.06 -0.31  0.02 -0.03  0.00 -0.14  0.33 -0.28 -0.33  0.53]
 [ 0.45  0.48  0.07  0.35  0.34  0.03 -0.40 -0.22 -0.11 -0.14 -0.25 -0.09]]
```

Nous en déduisions les coordonnées des individus dans le premier plan « factoriel ».

```
#calcul des coordonnées dans le premier plan
coord = numpy.sqrt(sol[0][:2])*sol[1][:,:2]

#affichage
print(pandas.DataFrame(numpy.round(coord,2),index=D.index,columns=['X1','X2']))
```

```
                    X1      X2
Autos
Toyota Corolla -514.37  -65.80
Lada 1300      -342.77   14.88
Alfasud TI     -322.67  -85.48
Lancia Beta    -290.43  131.45
Mazda 9295      151.93  -33.80
Fiat 132        -13.72   46.16
Alfetta 1.66    -44.53    9.76
Princess 1800   203.30   15.42
Audi 100         -7.65   52.42
Taunus 2000     353.77 -132.49
Opel Rekord     357.45  -88.16
Datsun 200L     469.67  135.64
```

La visualisation graphique – nuage de points – est un des atouts fort du positionnement multidimensionnel (Figure 86).

```
#représentation des individus dans le plan
#importer la librairie graphique
import matplotlib.pyplot as plt

#préparer le graphique
fig, ax = plt.subplots(figsize=(10,10))
```

```
ax.axis([-550,+550,-550,+550])
ax.plot([-550,+550],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-550,+550],color='silver',linestyle='--')
ax.set_xlabel("X1")
ax.set_ylabel("X2")

#ajouter les labels des véhicules
for i in range(n):
    ax.text(coord[i,0],coord[i,1],D.index[i])

#faire afficher
plt.show()
```

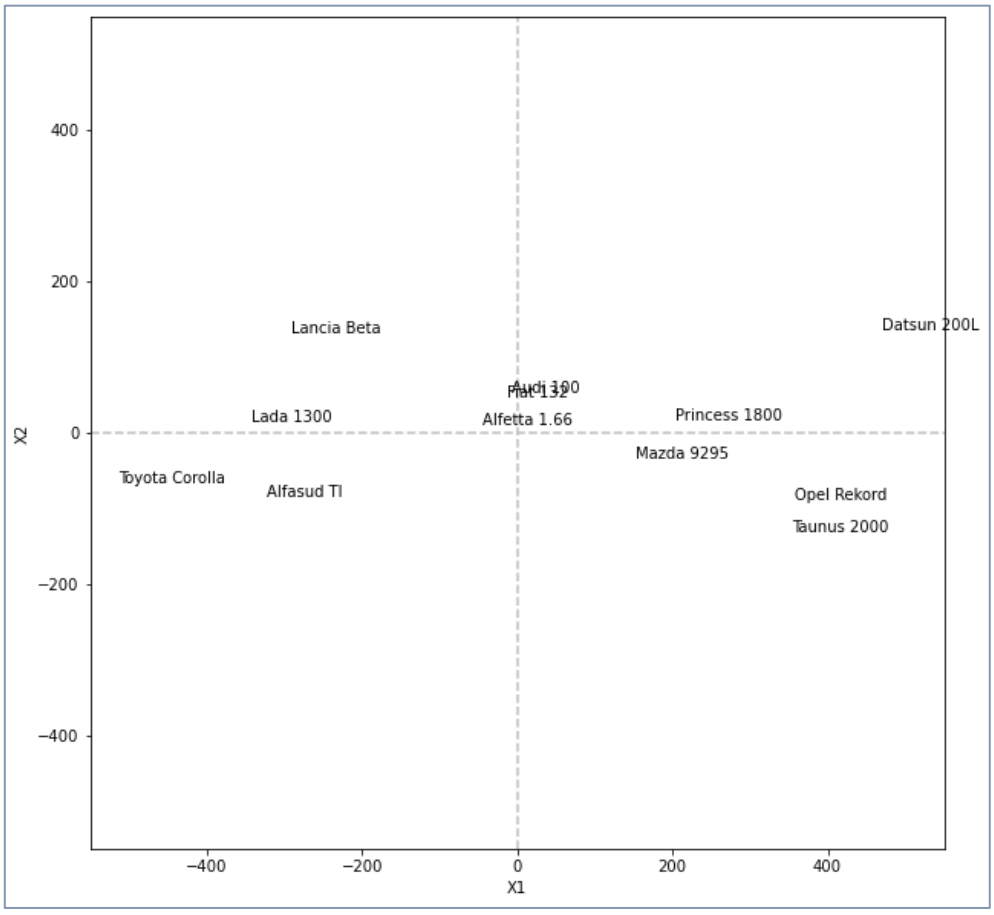


Figure 86 – Représentation des individus dans le plan – Données "Autos-MDS"

Deux commentaires par rapport au graphique que nous avions mis en avant en introduction de ce chapitre (section 3.2.1, Figure 83) :

– Les signes sont inversés sur le premier axe. Il s'agit d'un artefact de calcul simplement. Ce sont les positions relatives entre les objets qui importent en analyse factorielle. Nous retrouvons bien les proximités (ex. Audi 100 vs. Fiat 132, ...) et les oppositions (ex. Toyota Corolla vs. Datsun 200L, ...).

– En imposant les mêmes échelles en abscisse et ordonnée, un réflexe à prendre s'agissant des représentations graphiques en analyse factorielle, nous constatons que la dispersion des individus est avant tout tributaire de la 1ère dimension. Elle est porteuse de l'essentiel de l'information. Il faudra pouvoir quantifier cet aspect de la représentation.

Notons enfin que si les calculs sont simples dans leur principe, la diagonalisation d'une matrice (n x n) reste une opération complexe, ardue à mener lorsque l'effectif (n) devient important. Les implémentations usuelles de la MDS classique cantonnent la méthode aux matrices de distances de taille modérée.

3.2.4 Qualité de la représentation

3.2.4.1 Pourcentage d'information restituée

La quantité totale d'information véhiculée par les données est égale à la trace de la matrice B, soit la somme de ses valeurs propres :

$$Tr(B) = \sum_{i=1}^{n} \lambda_i$$

De fait, la qualité de l'approximation sur les « q » premiers facteurs peut être quantifiée par le ratio représentant la part cumulée d'information restituée :

$$\tau_q = \frac{\sum_{i=1}^{q} \lambda_i}{Tr(B)}$$

Pour les données « Autos-MDS », nous constatons que le premier plan (q = 2 premiers axes) restitue 99.7% de l'information disponible.

```
#trace de la matrice B
Tr_B = numpy.trace(B)
print(Tr_B)

1193444.8333333335

#vérification avec la somme des valeurs propres
print(numpy.sum(sol[0]))

1193444.833333333
```

```
#part d'information sur chaque axe
print(sol[0]/Tr_B)

[ 0.931  0.066  0.002  0.001  0.     0.    -0.     0.     0.     0.
 -0.    -0.   ]

#part d'information cumulée
numpy.set_printoptions(precision=3,suppress=True)
print(numpy.cumsum(sol[0])/Tr_B)

[0.931 0.997 0.999 1.    1.    1.    1.    1.    1.    1.    1.    1.   ]
```

La courbe montrant l'évolution de la part cumulée d'information restituée en fonction du nombre d'axes donne une bonne vision du processus d'approximation des distances initiales (Figure 87).

```
#courbe de la part d'information restituée
#en fonction du nombre de dimensions
fig, ax = plt.subplots(figsize=(5,5))
ax.plot(range(1,n+1),numpy.cumsum(sol[0])/Tr_B,".-")
ax.set_xlabel("Nb. axes")
ax.set_ylabel("Ratio cumulé")
plt.title("Part d'information restituée")
plt.show()
```

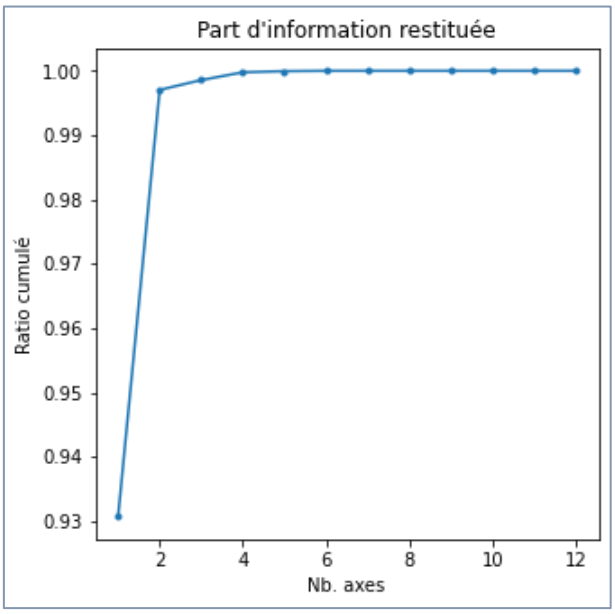


Figure 87 – MDS – Part d'information restituée par les "q" (q = 1 à 12) premiers axes – Données "Autos-MDS"

Manifestement, il n'y a rien à attendre au-delà du 2ème axe.

3.2.4.2 Identification du nombre de dimensions à conserver

Justement, à l'instar de l'analyse factorielle usuelle (ACP, AFC, ACM), le choix du nombre de dimensions à retenir est tout aussi crucial en positionnement multidimensionnel. Mais la finalité est un peu différente. Il ne s'agit pas tant de dégager des facteurs « pertinents » que l'on

souhaiterait interpréter, mais plutôt de disposer d'un espace de représentation restituant les véritables « formes » qui structurent les données, tout en respectant au mieux les distances initiales. Dans ce contexte, le diagramme ci-dessus (Figure 87) donne une bonne indication. Les axes situés après le « coude », surtout si ce dernier est suivi d'un plateau quasi-horizontal, restituent des informations résiduelles assimilables à des fluctuations d'échantillonnage.

Bien évidemment, il est également possible de produire un graphique de type « Eboulis des valeurs propres » (Figure 88), avec des règles de lecture similaires à celles de l'ACP.

```
#diagramme des valeurs propres
fig, ax = plt.subplots(figsize=(5,5))
ax.plot(range(1,n+1),sol[0],".-")
ax.set_xlabel("Nb. axes")
ax.set_ylabel("Val. propres")
plt.title("Diagramme des valeurs propres")
plt.show()
```

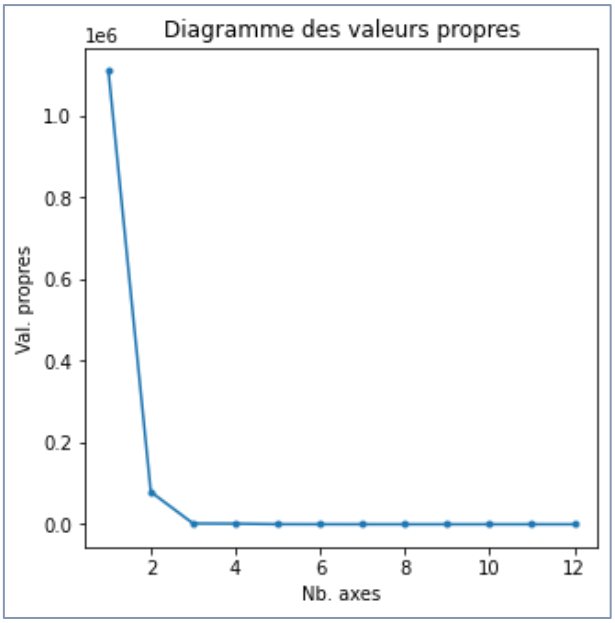


Figure 88 – Eboulis des valeurs propres – MDS – Données "Autos-MDS"

3.2.5 Matrice des distances restituées

La finalité principale du positionnement multidimensionnel est de projeter les individus dans un espace de représentation en respectant au mieux les distances entre les paires d'individus. Une approche simple pour évaluer la qualité du dispositif est de confronter les distances euclidiennes calculées dans le nouveau repère ($\hat{\delta}_{ii'}$), qui constituent les approximations, avec les distances initiales ($\delta_{ii'}$). La fonction de coût associée est appelée « stress »

$$Stress_D = \sqrt{\frac{\sum_{i,i'}\left(\hat{\delta}_{ii'} - \delta_{ii'}\right)^2}{\sum_{i,i'} \delta_{ii'}^2}}$$

Où

$$\hat{\delta}_{ii'} = \sqrt{\sum_{j=1}^{q}\left(x_{ij} - x_{i'j}\right)^2}$$

Si l'approximation est parfaite, le « stress » est égal à zéro.

Revenons aux données « Autos-MDS ». Nous souhaitons calculer les distances entre paires d'individus dans le premier plan (q = 2) pour les confronter aux distances initiales. Nous affichons pour rappel les coordonnées.

```
#rappel - coordonnées
print(coord)
```

```
[[-514.368  -65.796]
 [-342.767   14.875]
 [-322.67   -85.483]
 [-290.429  131.454]
 [ 151.928  -33.798]
 [ -13.717   46.156]
 [ -44.526    9.764]
 [ 203.304   15.42 ]
 [  -7.649   52.42 ]
 [ 353.774 -132.491]
 [ 357.453  -88.163]
 [ 469.666  135.641]]
```

Le calcul de la distance entre paires d'individus tient en une ligne de commande grâce à la magie de Python.

```
#distance estimée entre paires d'individus
delta_hat = numpy.sqrt(numpy.apply_along_axis(arr=coord,axis=1,func1d=lambda x:numpy.sum((x-coord)**2,axis=1)))
print(delta_hat)
```

```
[[   0.     189.618  192.707  298.423  667.064  513.016  475.879  722.253
   520.326  870.701  872.108 1004.44 ]
 [ 189.618    0.     102.351  127.789  497.083  330.533  298.284  546.071
   337.214  711.959  707.76   821.359]
 [ 192.707  102.351    0.     219.32   477.403  335.828  294.     535.565
   343.883  678.075  680.128  822.612]
 [ 298.423  127.789  219.32     0.     472.216  289.561  274.366  507.185
   293.617  696.179  684.093  760.107]
 [ 667.064  497.083  477.403  472.216    0.     183.932  201.226   71.148
   181.379  224.683  212.594  360.093]
 [ 513.016  330.533  335.828  289.561  183.932    0.      47.682  219.187
```

```
    8.721  408.613  394.726  491.596]
[ 475.879  298.284  294.     274.366  201.226   47.682    0.     247.895
   56.387  422.942  413.736  529.375]
[ 722.253  546.071  535.565  507.185   71.148  219.187  247.895    0.
  214.174  210.995  185.718  292.235]
[ 520.326  337.214  343.883  293.617  181.379    8.721   56.387  214.174
    0.     405.979  391.233  484.515]
[ 870.701  711.959  678.075  696.179  224.683  408.613  422.942  210.995
  405.979    0.      44.48   292.105]
[ 872.108  707.76   680.128  684.093  212.594  394.726  413.736  185.718
  391.233   44.48     0.     250.359]
[1004.44   821.359  822.612  760.107  360.093  491.596  529.375  292.235
  484.515  292.105  250.359    0.   ]]
```

Nous visualisons mieux les différences en mettant côte à côte les deux matrices ().

Matrice des distances initiales ($\delta_{ii'}$)

	Toyota Corolla	Lada 1300	Alfasud TI	Lancia Beta	Mazda 9295	Fiat 132	Alfetta 1.66	Princess 1800	Audi 100	Taurus 2000	Opel Rekord	Datsun 200L
Toyota Corolla												
Lada 1300	190.2											
Alfasud TI	195.3	106.0										
Lancia Beta	299.3	130.1	219.8									
Mazda 9295	667.2	497.4	477.9	472.4								
Fiat 132	513.6	331.6	336.1	289.8	184.9							
Alfetta 1.66	477.8	301.1	294.5	275.6	204.3	51.0						
Princess 1800	722.4	546.2	536.4	507.6	72.0	220.9	251.8					
Audi 100	521.1	339.9	346.7	295.3	184.2	35.2	73.8	217.2				
Taurus 2000	870.9	712.1	678.2	696.3	225.1	408.8	423.9	211.7	407.5			
Opel Rekord	872.3	708.3	680.5	684.2	213.3	395.0	414.8	187.4	391.8	47.8		
Datsun 200L	1004.6	821.4	822.8	760.4	360.5	492.0	530.4	292.9	486.5	292.3	251.7	

Matrice des distances restituées ($\hat{\delta}_{ii'}$)

	Toyota Corolla	Lada 1300	Alfasud TI	Lancia Beta	Mazda 9295	Fiat 132	Alfetta 1.66	Princess 1800	Audi 100	Taurus 2000	Opel Rekord	Datsun 200L
Toyota Corolla												
Lada 1300	189.6											
Alfasud TI	192.7	102.4										
Lancia Beta	298.4	127.8	219.3									
Mazda 9295	667.1	497.1	477.4	472.2								
Fiat 132	513.0	330.5	335.8	289.6	183.9							
Alfetta 1.66	475.9	298.3	294.0	274.4	201.2	47.7						
Princess 1800	722.3	546.1	535.6	507.2	71.1	219.2	247.9					
Audi 100	520.3	337.2	343.9	293.6	181.4	8.7	56.4	214.2				
Taurus 2000	870.7	712.0	678.1	696.2	224.7	408.6	422.9	211.0	406.0			
Opel Rekord	872.1	707.8	680.1	684.1	212.6	394.7	413.7	185.7	391.2	44.5		
Datsun 200L	1004.4	821.4	822.6	760.1	360.1	491.6	529.4	292.2	484.5	292.1	250.4	

Figure 89 – Confrontation des distances initiales et restituées dans le premier plan – Données "Autos-MDS"

La valeur du « stress » est de **0.00896**. Le premier plan qui restitue 99.7% de l'information disponible produit une approximation de qualité des distances initiales.

```
#calcul du stress
stress_D = numpy.sqrt(numpy.sum((delta_hat-delta)**2)/numpy.sum(delta**2))
print(stress_D)
```

```
0.008962132453455604
```

La confrontation des distances peut être aussi matérialisée dans un diagramme de Shepard où l'on place en abscisse les distances initiales, en ordonnée les distances estimées (Figure 90).

```
#diagramme de Shepard
fig, ax = plt.subplots(figsize=(5,5))
ax.plot(delta,delta,color='silver',linestyle='--')
ax.plot(delta,delta_hat,'go',markersize=3)
ax.set_xlabel("delta")
ax.set_ylabel("delta_hat")
plt.show()
```

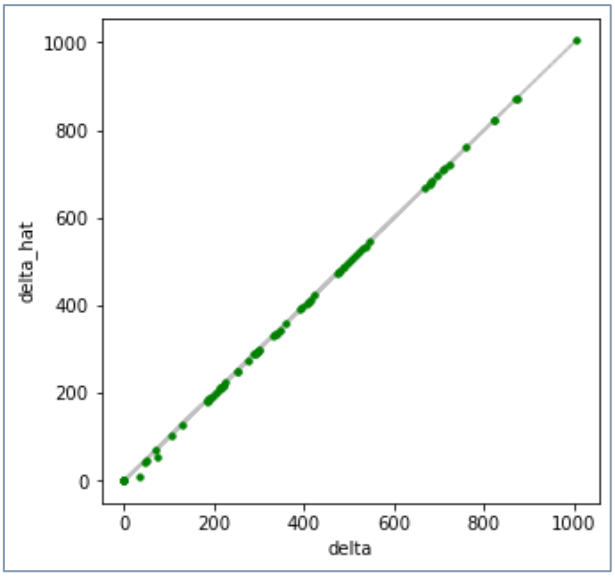


Figure 90 – Diagramme de Shepard – MDS – Données "Autos-MDS"

L'approximation est parfaite lorsque les points sont alignés sur la première bissectrice.

3.3 Traitement des individus supplémentaires

Nous pouvons appliquer le mécanisme des individus supplémentaires (ou illustratifs, cela dépend de la manière selon laquelle nous souhaitons les exploiter) au positionnement multidimensionnel. Il suffit pour cela de disposer, pour l'objet « i* » à positionner, du vecteur $\delta_{i^*} = (\delta_{i^*1}, \dots, \delta_{i^*n})$ comptabilisant les distances avec l'ensemble des objets qui constitue la base initiale (i = 1, …, n). Le calcul de ses coordonnées « factorielles » est réalisé en 2 étapes :

1. Transformer les distances $(\delta_{i^*i}\,; i = 1, \dots, n)$ en produit scalaire :

$$b_{i^*i} = -\frac{1}{2}\left(\delta_{i^*i}^2 - \delta_{i^*.}^2 - \delta_{.i}^2 + \delta_{..}^2\right)$$

Notons que les deux derniers termes (en gris) sont calculés une fois pour toutes sur la base initiale ayant servi à la construction du repère fourni par l'algorithme de MDS.

2. Déduire les coordonnées sur l'axe n°j à partir de celles des individus actifs

$$x_{i^*j} = \frac{1}{\lambda_j}\sum_{i=1}^{n} b_{i^*i} \times x_{ij}$$

Où (x_{ij}) est la coordonnée de l'individu actif n°i sur le facteur n°j.

Pour la base « Autos-MDS », nous souhaitons positionner 3 voitures françaises emblématiques des années 70 (Citroën GS Club, Renault 30, Peugeot 504 ; ceux de ma génération savent très bien ce que représentent ces véhicules). Nous disposons de leurs distances avec les n = 12 véhicules de notre base de référence (Figure 91).

Modele	Toyota Corolla	Lada 1300	Alfasud TI	Lancia Beta	Mazda 9295	Fiat 132	Alfetta 1.66	Princess 1800	Audi 100	Taunus 2000	Opel Rekord	Datsun 200L
Citroen GS Club	129.16	77.94	144.71	170.55	572.72	406.02	375.95	621.66	412.92	787.07	783.37	895.72
Renault 30	1583.99	1420.48	1391.17	1389.01	924.18	1100.84	1124.95	882.16	1097.45	713.52	714.22	668.53
Peugeot 504	720.65	544.46	535.10	505.81	72.03	219.44	250.81	7.35	215.08	214.15	189.76	294.34

Figure 91 – Véhicules supplémentaires à positionner – Données "Autos-MDS"

Les voisins les plus proches ont été mis en exergue (police bleue) pour chaque voiture supplémentaire. Le tableau est déchiffrable sur une base de taille réduite. Nous observons ici que : la Citroën GS est assez proche de la Lada 1300, plutôt une polyvalente ; la voisine immédiate de la Renault 30 est la Datsun 200L, une routière ; la Peugeot 504 est très proche de la Princess 1800, une familiale. Mais dès lors le nombre d'observations augmente, insérer les nouveaux individus dans la représentation graphique demeure la solution la plus lisible.

Sous Python, nous chargeons le tableau des distances pour les 3 véhicules supplémentaires.

```
#chargement des observations supplémentaires
DSupp = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="AUTOS_MDS_SUPP",index_col=0)
print(DSupp.transpose())
```

```
Modele          Citroen GS Club   Renault 30  Peugeot 504
Toyota Corolla       129.162688  1583.989583   720.653176
Lada 1300             77.942286  1420.483368   544.458447
Alfasud TI           144.710055  1391.167495   535.104663
Lancia Beta          170.552045  1389.010439   505.812218
Mazda 9295           572.718081   924.179636    72.027772
Fiat 132             406.016010  1100.836500   219.437918
Alfetta 1.66         375.948135  1124.952888   250.812679
Princess 1800        621.662288   882.159849     7.348469
Audi 100             412.922511  1097.450227   215.083705
Taunus 2000          787.067341   713.522950   214.151815
Opel Rekord          783.369006   714.217754   189.755105
Datsun 200L          895.715357   668.528234   294.339940
```

Nous les transformons avec la formule de Torgerson.

```
#transformation en produit scalaire (Torgerson)
dSupp = DSupp.values
```

```
BSupp = numpy.apply_along_axis(arr=dSupp,axis=1,func1d=lambda x:-0.5*(x**2-numpy.sum(x**2)-d2+d3))
print(BSupp)
```

```
[[1726201.681 1655947.597 1645351.181 1636276.597 1448131.014 1518762.431
  1530733.431 1427671.014 1516594.264 1361607.681 1360984.597 1318416.514]
 [6416191.681 6586258.597 6624308.181 6622305.597 7121240.014 6931426.431
  6904802.431 7167960.014 6935808.264 7352947.681 7348924.597 7432264.514]
 [ 669687.681  705582.597  707468.181  717712.597  804355.014  771925.431
   764763.431  815691.014  773531.264  843229.681  844629.597  871066.514]]
```

Nous appliquons la formule pour obtenir les coordonnées factorielles des individus supplémentaires, que nous exploitons pour les introduire dans la représentation graphique (Figure 92).

```
#calcul des coordonnées factorielles
coordSupp = numpy.dot(BSupp, coord)
coordSupp = coordSupp/sol[0][:2]
print(pandas.DataFrame(coordSupp,index=DSupp.index,columns=['X1','X2']))
```

```
                         X1          X2
Modele
Citroen GS Club  -418.260083   19.707020
Renault 30       1065.891142 -165.824064
Peugeot 504       201.478597   16.499780
```

```
#représentation des individus dans le plan
import matplotlib.pyplot as plt

#préparer le graphique
fig, ax = plt.subplots(figsize=(15,15))
ax.axis([-1100,+1100,-1100,+1100])
ax.plot([-1100,+1100],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-1100,+1100],color='silver',linestyle='--')
ax.set_xlabel("X1")
ax.set_ylabel("X2")

#ajouter les labels des véhicules actifs
for i in range(n):
    ax.text(coord[i,0],coord[i,1],D.index[i],c='dimgray')

#ajouter les labels des véhicules supplémentaires
for i in range(coordSupp.shape[0]):
    ax.text(coordSupp[i,0],coordSupp[i,1],DSupp.index[i],c='magenta')

#faire afficher
plt.show()
```

Encore une fois, il faut bien veiller à utiliser la même échelle en abscisse et ordonnée pour éviter les proximités fallacieuses. Dans notre représentation, la dispersion repose principalement sur le 1er axe et, dans cette optique, la Renault 30 est effectivement plus proche de la Datsun 200L que du tandem (Opel Rekord, Taunus 2000). La perception visuelle aurait été faussée si, pour plus de clarté dans la lecture des étiquettes, nous avions agrandi l'échelle en ordonnée.

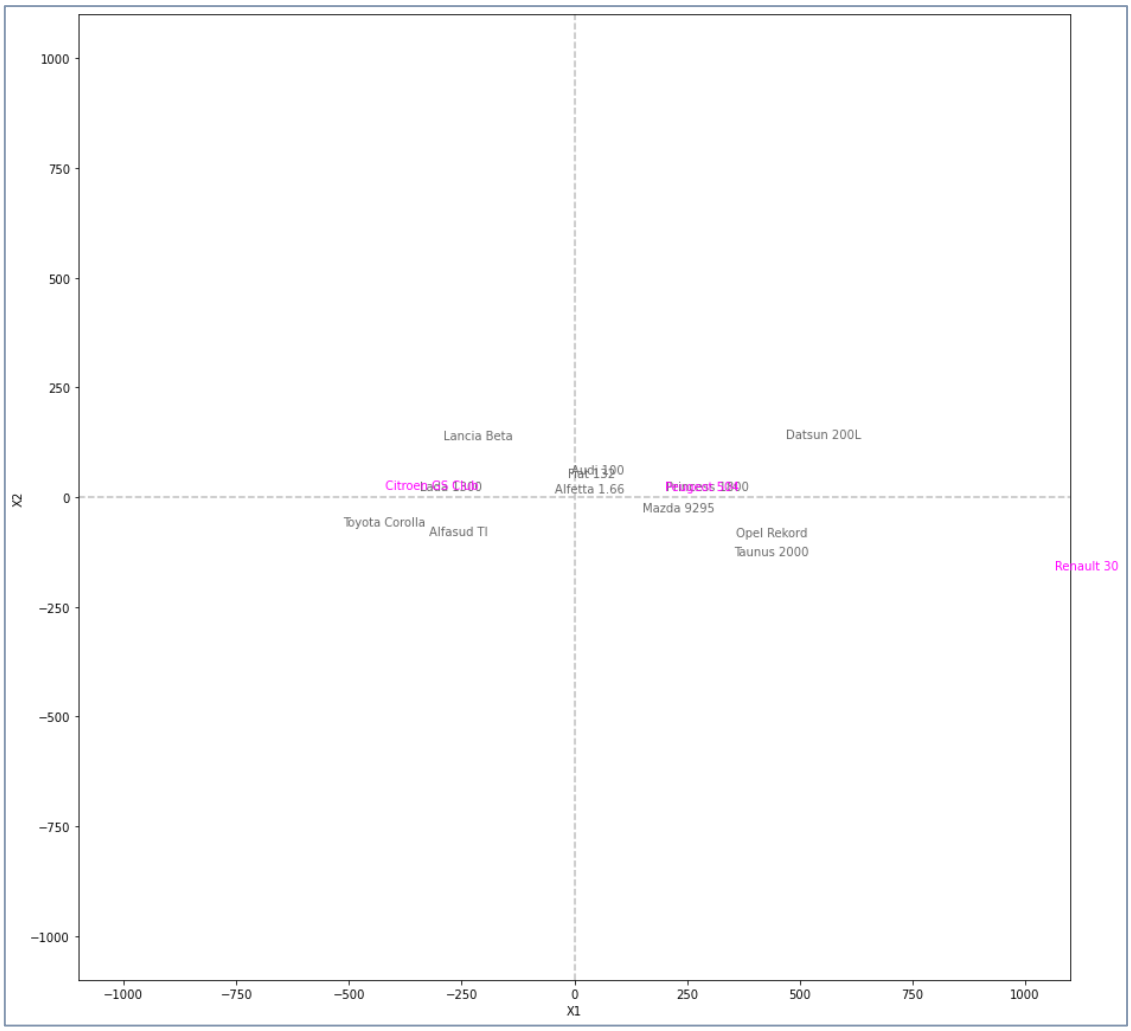


Figure 92 – Position des individus supplémentaires (magenta) – Données "Autos-MDS"

3.4 Positionnement multidimensionnel classique et ACP

L'ACP est une technique de visualisation qui travaille à partir d'un tableau « individus – variables ». Elle permet de projeter les observations dans un espace de dimensionnalité réduite en minimisant la perte d'information. Dans le cas où $(\delta_{ii'})$ est une distance euclidienne calculée sur des données initiales exprimées en « individus – variables », l'ACP et le positionnement multidimensionnel sont formellement équivalents (Debois D., « Une introduction au positionnement multidimensionnel », Revue Modulad, n°32, pages 1 à 28, 2005 ; voir page 17).

Pour les données « Autos-MDS », nous disposons des données sources (Figure 93) qui ont servi au calcul de la matrice des distances entre paires de véhicules. Nous souhaitons leur appliquer

une ACP non-normée et inspecter les coordonnées des individus dans le premier plan factoriel. Nous verrons s'il y a équivalence avec celles fournies par le MDS.

Modele	CYL	PUISS	LONG	LARG	POIDS	VMAX
Toyota Corolla	1166	55	399	157	815	140
Lada 1300	1294	68	404	161	955	140
Alfasud TI	1350	79	393	161	870	165
Lancia Beta	1297	82	429	169	1080	160
Mazda 9295	1769	83	440	165	1095	165
Fiat 132	1585	98	439	164	1105	165
Alfetta 1.66	1570	109	428	162	1060	175
Princess 1800	1798	82	445	172	1160	158
Audi 100	1588	85	468	177	1110	160
Taunus 2000	1993	98	438	170	1080	167
Opel Rekord	1979	100	459	173	1120	173
Datsun 200L	1998	115	469	169	1370	160

Figure 93 – Tableau source ayant servi au calcul des distances par paires de "Autos-MDS"

Sous Python, nous chargeons tout d'abord le tableau individus-variables décrivant les véhicules à partir de leurs caractéristiques (CYL, PUISS, …, VMAX).

```
#chargement des données sources – tableau individus - variables
DRaw = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="AUTOS_MDS_SOURCE",index_col=0)
print(DRaw)
```

```
                CYL  PUISS  LONG  LARG  POIDS  VMAX
Modele
Toyota Corolla  1166     55   399   157    815   140
Lada 1300       1294     68   404   161    955   140
Alfasud TI      1350     79   393   161    870   165
Lancia Beta     1297     82   429   169   1080   160
Mazda 9295      1769     83   440   165   1095   165
Fiat 132        1585     98   439   164   1105   165
Alfetta 1.66    1570    109   428   162   1060   175
Princess 1800   1798     82   445   172   1160   158
Audi 100        1588     85   468   177   1110   160
Taunus 2000     1993     98   438   170   1080   167
Opel Rekord     1979    100   459   173   1120   173
Datsun 200L     1998    115   469   169   1370   160
```

Nous centrons les variables.

```
#centrage des données
Z = DRaw.values - numpy.mean(DRaw.values,axis=0)
print(Z)
```

```
[[-449.583  -32.833  -35.25    -9.667 -253.333  -20.667]
 [-321.583  -19.833  -30.25    -5.667 -113.333  -20.667]
 [-265.583   -8.833  -41.25    -5.667 -198.333    4.333]
 [-318.583   -5.833   -5.25     2.333   11.667   -0.667]
 [ 153.417   -4.833    5.75    -1.667   26.667    4.333]
 [ -30.583   10.167    4.75    -2.667   36.667    4.333]
 [ -45.583   21.167   -6.25    -4.667   -8.333   14.333]
 [ 182.417   -5.833   10.75     5.333   91.667   -2.667]
```

```
 [ -27.583   -2.833   33.75    10.333   41.667   -0.667]
 [ 377.417   10.167    3.75     3.333   11.667    6.333]
 [ 363.417   12.167   24.75     6.333   51.667   12.333]
 [ 382.417   27.167   34.75     2.333  301.667   -0.667]]
```

Et nous faisons appel à la classe de calcul PCA de la librairie « scikit-learn » (TUTO 1).

```
#ACP avec scikit-learn
from sklearn.decomposition import PCA
acp = PCA(svd_solver='full',n_components=2)

#coord. fact.
fact = acp.fit_transform(Z)
```

Nous affichons enfin les coordonnées des individus dans le premier plan factoriel,....

```
#affichage des coordonnées
print(pandas.DataFrame(fact,index=DRaw.index,columns=['F1','F2']))
```

```
                      F1          F2
Modele
Toyota Corolla  514.368216  -65.795531
Lada 1300       342.766700   14.875213
Alfasud TI      322.669602  -85.483117
Lancia Beta     290.429211  131.454491
Mazda 9295     -151.927781  -33.798481
Fiat 132         13.716817   46.156247
Alfetta 1.66     44.526156    9.763934
Princess 1800  -203.304446   15.419721
Audi 100          7.649174   52.420252
Taunus 2000    -353.774482 -132.490668
Opel Rekord    -357.453350  -88.162572
Datsun 200L    -469.665817  135.640512
```

... elles sont en tous points conformes aux coordonnées du positionnement multidimensionnel appliqué à la matrice des distances euclidiennes entre paires d'observations (Figure 86) (au signe près pour le 1er facteur, mais nous savons maintenant que ce sont les positions relatives qui importent en analyse factorielle).

3.5 Plus loin avec le positionnement multidimensionnel

3.5.1 Travailler avec une matrice de similarité

Lorsque le tableau de données se présente sous la forme d'une matrice de similarité ($s_{ii'}$) avec,

$$\begin{cases} s_{ii} = max_s \\ s_{ii'} \leq max_s \end{cases}$$

Il est possible de le ramener à une matrice de dissimilarités via des transformations adaptées, en particulier ($\delta_{ii'} = max_s - s_{ii'}$). Nous pouvons dès lors appliquer l'algorithme MDS.

3.5.2 Cas de B non définie positive

La matrice issue de la transformation de Torgerson (B) peut comporter des valeurs propres négatives lorsque ($\delta_{ii'}$) n'est pas une distance euclidienne ou, pire, est une simple dissimilarité qui ne respecte pas l'inégalité triangulaire. Deux pistes sont le plus souvent évoquées pour remédier à cet écueil :

Solution 1 : Prendre en compte uniquement les facteurs associés aux valeurs propres positives. La solution est simple voire triviale, mais elle a le mérite de l'opérationnalité immédiate.

Solution 2 : Ajouter une constante positive « c* » aux éléments hors diagonale principale de la matrice B c.-à-d.

$$B^*: b^*_{ii'} = b_{ii'} + c^* \,, \forall\, i \neq i'$$

De manière à que les valeurs propres de (B*) soient positives. Attention, si « c* » est trop grand, l'information peut être détériorée et la représentation déformée. Il existe une formule analytique pour trouver le plus petit « c* » assurant ($\lambda^*_j \geq 0, \forall\, j$) (Saporta, 2006, section 7.5.1.2). Elle est implémentée dans des outils tels que cmdscale() du package « stats » de R par exemple (voir l'option « add »).

3.5.3 MDS métrique et non-métrique

MDS métrique. Le positionnement multidimensionnel métrique est une généralisation de l'approche classique où l'on cherche à optimiser directement la fonction coût « stress » via une heuristique.

$$stress = \frac{\sum_{i,i'} \left[\hat{\delta}_{ii'} - f(\delta_{ii'})\right]^2}{\sum_{i,i'} \delta^2_{ii'}}$$

Où f() est une fonction affine de la distance $f(\delta) = a + b \times \delta$

Remarque : si $f(\delta) = \delta$, la solution n'est pas pour autant identique au positionnement multidimensionnel classique puisque les approches (MDS classique vs. MDS métrique) n'optimisent pas les mêmes fonctions de coût.

MDS non-métrique. Dans le cas du MDS non-métrique, ce n'est pas tant les valeurs et les écarts entre les dissimilarités ($\delta_{ii'}$) qui nous intéressent mais plutôt l'ordonnancement qu'elles induisent (l'objet A est proche de B, éloigné de C, plus éloignée encore de D, etc.). L'objectif est de le restituer dans le repère factoriel. La démarche consiste toujours à optimiser la fonction « stress ». Mais f() sera dans ce cas une fonction monotone qui a pour propriété de préserver l'ordre c.-à-d. $\delta_{ii'} < \delta_{kk'} \Rightarrow f(\delta_{ii'}) < f(\delta_{kk'})$

3.5.4 Travailler sur les matrices des corrélations

Le principe du positionnement multidimensionnel peut être appliqué à l'étude des relations entre variables, lesquelles peuvent être quantifiées par la corrélation ($r_{jj'}$) par exemple. L'objectif toujours est de pouvoir les positionner, les structurer, selon l'intensité et le sens des liaisons qu'elles entretiennent. Pour convertir les corrélations en distances, plusieurs types de transformations sont possibles.

Solution 1 : Qui tient compte du sens de la relation

$$\delta_{jj'} = 1 - r_{jj'}$$

Solution 2 : Qui se focalise sur son intensité

$$\delta_{jj'} = \sqrt{1 - r_{jj'}^2}$$

Remarque : Ignorer le sens de la relation n'induit pas forcément à une perte d'information. En effet, il peut être artificiel et dépendre de la nature des mesures utilisées. Si l'on étudie la relation entre l'acuité visuelle et la vieillesse par exemple, la liaison est négative si nous utilisons l'âge des personnes, elle devient positive si nous exploitons l'année de naissance. Dans les deux cas, le carré de la corrélation correspond bien à la même valeur.

3.6 Un exemple – Distances routières entre villes de Madagascar

Le positionnement multidimensionnel est disponible dans la fameuse librairie « scikit-learn » pour Python. La classe de calcul MDS implémente les méthodes métriques et non-métriques. Le MDS classique n'est pas proposé mais, en pratique, j'ai constaté que les résultats des approches classiques et métriques étaient souvent convergents. Pour apprécier pleinement la teneur des résultats que nous mettrons en avant ici, la même étude a été menée sous R (TUTO 20).

3.6.1 Données – Distances routières entre villes

L'objectif est de représenter dans le plan les positions relatives de villes en prenant en entrée les distances kilométriques des routes qui les relient. C'est une application très parlante et/mais ressassée du positionnement multidimensionnel (ex. Desbois, section 3 ; Wikistat, Figure 1). J'essaie d'être original ici en traitant les villes de Madagascar.

Madagascar est une île superbe de l'Océan Indien. Elle est subdivisée en 6 provinces historiques dont les chefs-lieux sont mis en évidence dans la carte ci-dessous (Figure 95). En adoptant les dénominations de Google Maps, nous avons, dans le sens trigonométrique : Antsiranana, Majunga, Tuléar, Fianarantsoa, Tananarive (la capitale), Toamasina. Via l'application en ligne, j'ai collecté manuellement[2] les distances entre les villes pour obtenir le tableau suivant (Figure 94). La matrice est symétrique.

Ville	Antsiranana	Majunga	Tuléar	Fianarantsoa	Tananarive	Toamasina
Antsiranana	0	859	2038	1522	1109	1287
Majunga	859	0	1495	979	566	926
Tuléar	2038	1495	0	516	930	1266
Fianarantsoa	1522	979	516	0	414	750
Tananarive	1109	566	930	414	0	355
Toamasina	1287	926	1266	750	355	0

Figure 94 – Distances routières entre les principales villes de Madagascar

[2] Il est possible d'accéder à ces informations directement avec une API. Mais il fallait s'enregistrer sur Google et fournir des informations personnelles. C'est exclu en ce qui me concerne. L'autre piste était de passer par une API fournie par « OpenStreetMap », totalement libre, je montre son usage dans un tutoriel (« OpenStreetMap sous R », avril 2019). Malheureusement, certaines villes ne sont pas recensées par l'outil. D'où, fautes de grives, la collecte manuelle des informations.

Précision importante, certains trajets nécessitaient de prendre le ferry (plutôt un bac s'agissant des rivières malgaches). Je les ai exclus pour privilégier les solutions exclusivement routières. Ainsi, il faut passer par la capitale Tananarive si on veut utiliser uniquement la route pour aller de Fianarantsoa à Toamasina (voir TUTO 20, Figure 3).

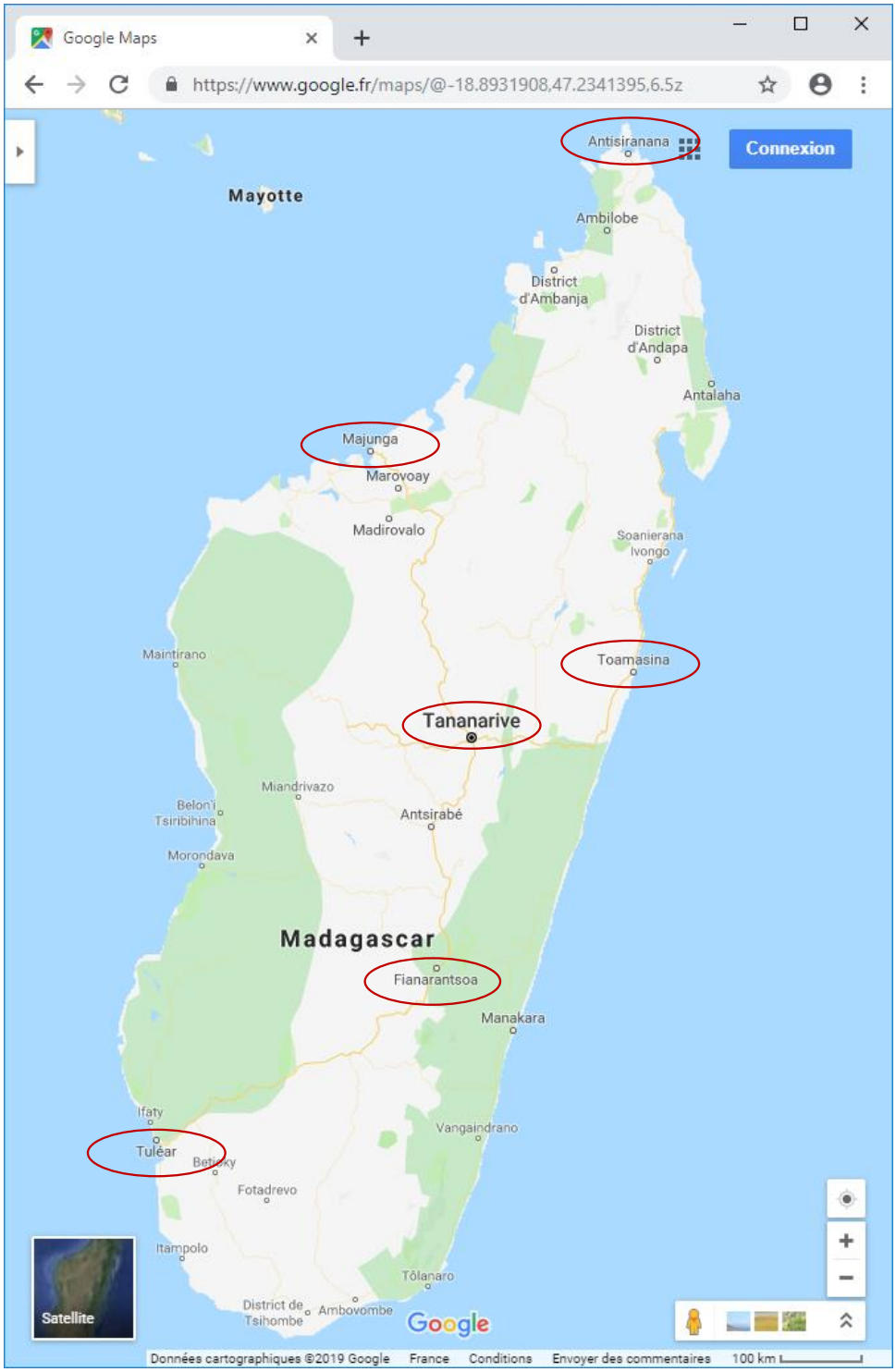


Figure 95 – Carte de Madagascar, avec les 6 villes principales

3.6.2 Pratique du positionnement multidimensionnel sous Python

3.6.2.1 Importation et inspection des données

Première étape toujours de tout traitement, nous importons et inspectons les données.

```
#chargement - index_col = 0 pour indiquer que la colonne n°0 est un label
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="MDS_MADAGASCAR",index_col=0)
print(D)

#nombre d'observations
n = D.shape[0]
```

```
             Antsiranana  Majunga  Tulear  Fianarantsoa  Tananarive  \
Ville
Antsiranana            0      859    2038          1522        1109
Majunga              859        0    1495           979         566
Tulear              2038     1495       0           516         930
Fianarantsoa        1522      979     516             0         414
Tananarive          1109      566     930           414           0
Toamasina           1287      926    1266           750         355

             Toamasina
Ville
Antsiranana        1287
Majunga             926
Tulear             1266
Fianarantsoa        750
Tananarive          355
Toamasina             0
```

3.6.2.2 Quantité d'information disponible – Inertie totale

Puisque nous disposons de la distance entre paires d'individus, nous pouvons calculer l'inertie du nuage de points (section 1.1.2.3).

```
#librairie numpy
import numpy

#inertie totale
I_tot = numpy.sum(D.values**2)/(2*(n**2))
print(I_tot)
```

```
501644.1666666667
```

Elle représente la quantité d'information totale portée par les données. Nous verrons comment elle sera décomposée sur les différents axes issus du MDS.

3.6.2.3 Réalisation du positionnement multidimensionnel

Nous pouvons lancer l'algorithme de positionnement multidimensionnel. Nous indiquons à la fonction que nous souhaitons produire les 2 premiers axes seulement (n_components = 2), et que les distances sont déjà précalculées (dissimilarity = 'precomputed'), il n'est pas nécessaire de les déduire de la matrice des données.

```
#importation de la classe de calcul
from sklearn.manifold import MDS

#instanciation
mds = MDS(n_components=2,dissimilarity='precomputed')

#lancement des calculs
mds.fit(D.values)
```

```
MDS(dissimilarity='precomputed', eps=0.001, max_iter=300, metric=True,
    n_components=2, n_init=4, n_jobs=None, random_state=None, verbose=0)
```

D'autres paramètres prennent leurs valeurs par défaut. Il faut lire attentivement la documentation pour apprécier leur éventuelle influence sur le déroulement des calculs et la teneur des résultats.

3.6.2.4 « Stress » et part d'information restituée

Le champ « stress_ » devrait nous donner une idée de la qualité de l'approximation dans le premier plan de représentation.

```
#valeur brute du stress (numérateur)
print(mds.stress_)
```

```
47646.60256584347
```

En réalité, la valeur correspond au numérateur – sans la racine carrée – de la formule usuelle (section 3.2.5 ; $Stress_D$). Nous devons la normaliser pour obtenir quelque chose de lisible.

```
#stress ramené à la dispersion initiale (cf. formule en section 3.2.5)
print(numpy.sqrt(mds.stress_/numpy.sum(D.values**2)))
```

```
0.036320501623653766
```

Manifestement, l'approximation des distances dans le plan est d'une très bonne tenue. $Stress_D$ serait égale à 0 si elle était parfaite.

3.6.2.5 Coordonnées des objets

Le champ « embedding_ » fournit les coordonnées factorielles.

```
#coordonnées
print(pandas.DataFrame(mds.embedding_,index=D.index,columns=['X1','X2']))
```

```
                     X1           X2
Ville
Antsiranana   127.541424 -1091.324109
Majunga       403.445790  -362.895625
Tulear         56.479148  1007.194885
Fianarantsoa   -8.496627   474.038401
Tananarive    -82.638593    32.604814
Toamasina    -496.331142   -59.618366
```

Avant de passer à la représentation graphique, nous procédons à quelques vérifications. Déjà, est-ce que les facteurs sont centrés ?

```
#est-ce que les axes sont centrés ? => oui
print(numpy.mean(mds.embedding_,axis=0))
```

```
[9.47390314e-15 3.78956126e-14]
```

La réponse est oui.

3.6.2.6 Informations restituées par les axes et par le repère

Quelle est la part d'information restituée par les deux premiers axes ?

Pour répondre à cette question, nous calculons tout d'abord les dispersions sur chaque axe.

```
#dispersion sur les 2 axes
I_axes = numpy.mean(mds.embedding_**2,axis=0)
print(I_axes)
```

```
[ 72578.52450422 427742.15193544]
```

Puis nous les ramenons à l'inertie totale.

```
#dispersion ramenée à l'inertie totale
print(I_axes/I_tot)
```

```
[0.14468129 0.85268041]
```

Le premier facteur restitue 14.46% de l'information, la seconde 85.26%. Le cumul des deux revient à 99.7%. Effectivement, comme le « stress » le suggérait précédemment, l'approximation est d'une excellente tenue.

3.6.2.7 Représentation graphique

Nous pouvons maintenant passer à la représentation graphique (Figure 96)

```
#représentation graphique
#importer la librairie graphique
import matplotlib.pyplot as plt

#préparer le graphique
fig, ax = plt.subplots(figsize=(10,10))
ax.axis([-1100,+1100,-1100,+1100])
ax.plot([-1100,+1100],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-1100,+1100],color='silver',linestyle='--')
ax.set_xlabel("X1")
ax.set_ylabel("X2")

#ajouter les labels des véhicules
for i in range(n):
    ax.text(-mds.embedding_[i,0],-mds.embedding_[i,1],D.index[i])

#faire afficher
plt.show()
```

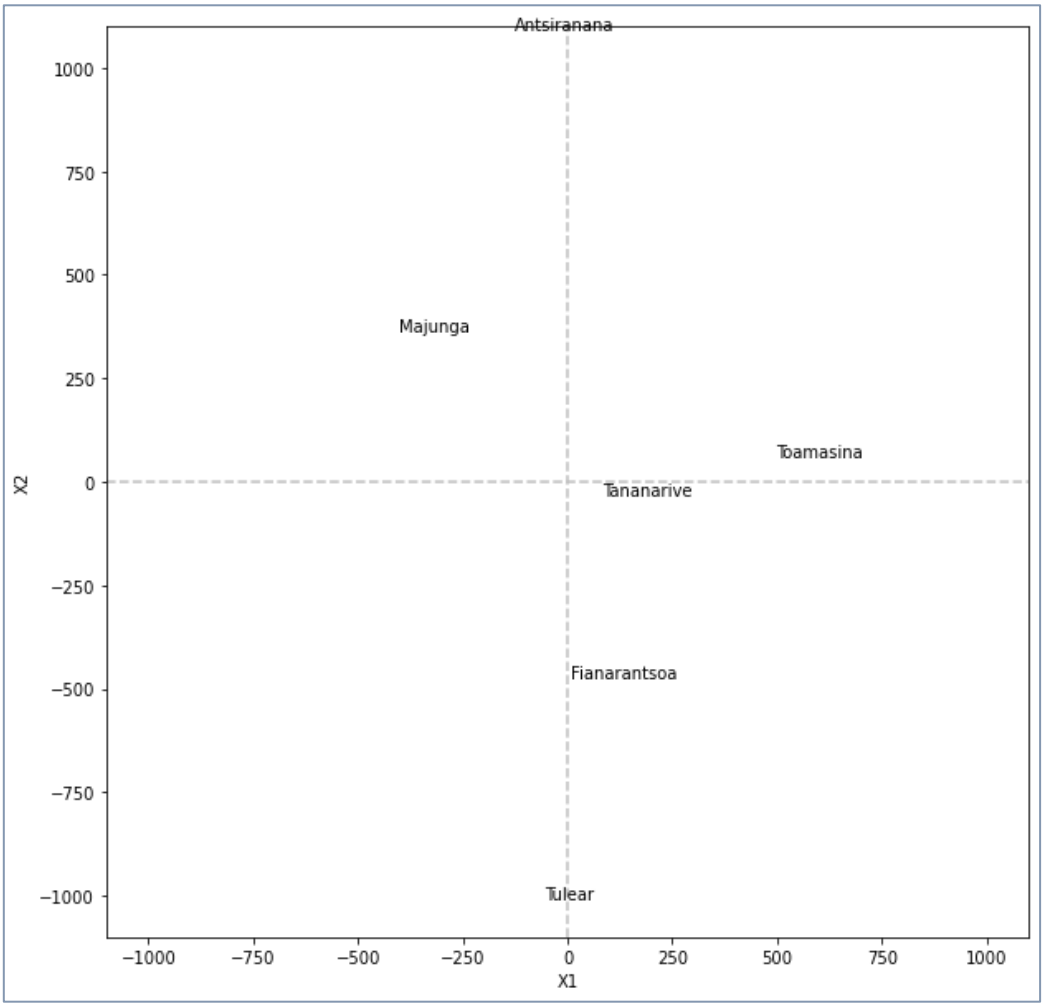


Figure 96 – Représentation des villes de Madagascar dans le plan – MDS

Nous avons modifié les signes des coordonnées pour que les positions relatives soient conformes à la réalité. La correspondance est quasi-parfaite si nous pivotons le graphique de quelques degrés dans le sens horaire (Figure 97).

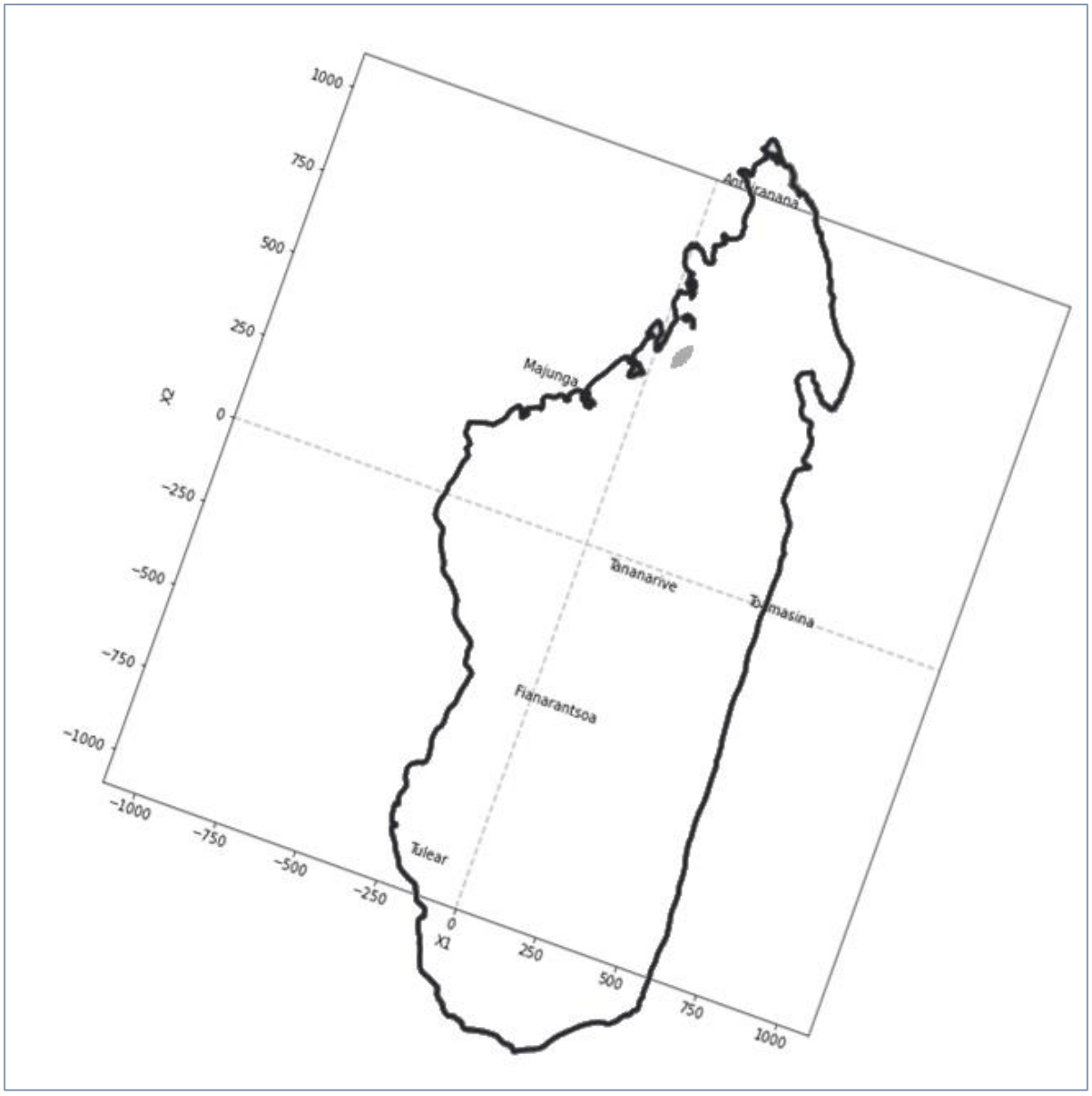


Figure 97 – Position des villes (MDS) et carte de Madagascar

J'avoue être assez content du résultat.

4 Analyse factorielle des correspondances (AFC)

L'analyse factorielle des correspondances (AFC) – ou analyse des correspondances pour simplifier - propose une vision synthétique des informations « saillantes » portées par un tableau de contingence. Elle permet de débroussailler rapidement les grands tableaux. Son pouvoir de séduction repose en grande partie sur les représentations graphiques qu'elle propose. Elles nous permettent de situer facilement les similarités (dissimilarités) entre les profils et les attractions (répulsions) entre les modalités. L'AFC est bien une technique factorielle. Les facteurs – les variables latentes – qui en sont issus sont des combinaisons linéaires des points modalités (lignes ou colonnes) exprimés par des profils (lignes ou colonnes).

Techniquement, on peut voir l'AFC comme une méthode d'ajustement des nuages de profils lignes et colonnes, permettant leur représentation simultanée, c.-à-d. dans le même repère. Nous pouvons aussi la voir sous l'angle de la décomposition orthogonale du χ^2 d'écart à l'indépendance. Ce prisme est intéressant parce qu'il montre clairement qu'avant de nous intéresser à la structure des écarts, il est très important de considérer leur importance et se poser la question : y a –t-il de l'information pertinente dans le tableau à analyser ? Si le χ^2 (ou le ϕ^2) est trop faible, l'étude subséquente est illusoire.

Enfin, si l'AFC s'applique en priorité sur les tableaux de contingence (tableau de comptage). Elle est en réalité valable dès lors que les valeurs sont positives, que les notions de marges (sommes en ligne et colonne) et profils (ratios en ligne et colonne) ont un sens dans le tableau que nous analysons.

4.1 Principe de l'analyse factorielle des correspondances

4.1.1 Tableau de contingence

Pour illustrer notre propos, nous utilisons la base « CSP Filières » qui est le fruit d'une enquête où l'on a croisé l'origine sociale des étudiants (à travers la CSP – catégorie social professionnelle, version très simplifiée - des parents) avec les choix de filières à l'Université. Elle est tirée de la page de cours de F-G. Carpentier de l'Université de Brest.

CSP\Filière	Droit	Sciences	Médecine	IUT	Total
Exp.agri	80	99	65	58	302
Patron	168	137	208	62	575
Cadre.sup	470	400	876	79	1825
Employé	145	133	135	54	467
Ouvrier	166	193	127	129	615
Total	1029	962	1411	382	3784

Figure 98 - CSP et Choix de filières avec les marges – Tableau des effectifs observés

Un tableau de contingence peut s'appréhender comme un croisement entre deux variables qualitatives (mais il peut être plus riche comme nous le verrons dans la section 4.5). On cherche à savoir si l'un apporte de l'information sur l'autre et inversement en caractérisant la relation de différentes manières :

1. Quelle est la structure des filières choisies selon la CSP ? On parle alors de profil ligne.
2. Est-ce que la structure est différente d'une CSP à l'autre ? Nous matérialisons les écarts à l'aide d'une distance adaptée à notre étude.
3. Nous pouvons mener la même étude mais sous l'angle des profils colonnes : les compositions des filières sont-elles différentes en termes de CSP ?
4. Nous pouvons enfin étudier les associations et répulsions entre CSP et filières : certaines CSP ont-elles une préférences pour certaines filières ? Des filières spécifiques attirent-elles des catégories particulières de CSP ?

Nous pouvons compléter les marges en ligne et colonnes en effectuant la somme des effectifs. Il s'agit de la partie grisée de notre tableau ci-dessus (Figure 98). Nous disposons de (n = 3874)

observations, 302 étaient des enfants d'exploitants agricoles (« Exp.Agri »), 1029 ont choisi la filière « Droit », etc. Dans ce chapitre, nous utiliserons les notations suivantes pour représenter les différents effectifs (Figure 99) :

Y / X	x_1	x_l	x_L	Σ
y_1				
		$\vdots$		
y_k	$\cdots$	n_{kl}	$\cdots$	$n_{k.}$
		$\vdots$		
y_K				
Σ		$n_{.l}$		n

Figure 99 – Tableau des effectifs – Notations – AFC

Nous pouvons utiliser les fréquences relatives pour estimer les différentes probabilités :

5. $P(Cadre.Sup) = \frac{1825}{3784} = 48.2\%$ est la proportion des enfants de « Cadre.Sup » ;
6. $P(Médecine) = \frac{1411}{3784} = 37.3\%$ est la proportion des étudiants qui ont fait le choix de la filière « Médecine ».
7. $P(Médecine \ \& \ Cadre.Sup) = \frac{876}{3784} = 23.2\%$ est la proportion de personnes qui ont choisi « Médecine » et qui sont des enfants de « Cadre.Sup ». On a une idée de concomitance, d'une association.
8. $P(Médecine \ / \ Cadre.Sup) = \frac{876}{1825} = 48.0\%$ est la proportion de personnes qui ont choisi « Médecine » parmi les enfants de « Cadres.Sup ». On serait plutôt sur une idée de causalité.

Dans ce qui suit, nous étudions les différentes manières d'explorer ces proportions, qui sont autant de points de vue sur l'analyse factorielle des correspondances.

Python. Comme de coutume, nous retraçons les calculs sous Python. Nous chargeons la feuille « **AFC_ETUDES** » du classeur « **Data_Methodes_Factorielles.xlsx** ».

```
#chargement - index_col = 0 pour indiquer que la colonne n°0 est un label
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="AFC_ETUDES",index_col=0)

#affichage des données
print(D)
```

```
                Droit  Sciences  Medecine  IUT
CSP_vs_Filiere
ExpAgri            80        99        65   58
Patron            168       137       208   62
CadreSup          470       400       876   79
Employe           145       133       135   54
Ouvrier           166       193       127  129
```

```
#librairie
import numpy

#calcul des totaux en ligne
tot_lig = numpy.sum(D.values,axis=1)
print(tot_lig)
```

```
[ 302  575 1825  467  615]
```

```
#calcul des totaux en colonne
tot_col = numpy.sum(D.values,axis=0)
print(tot_col)
```

```
[1029  962 1411  382]
```

4.1.2 Analyse des profils lignes

4.1.2.1 Profils lignes

Le premier prisme consiste à analyser les profils lignes – les proportions en ligne – du tableau de contingence. Ils sont calculés sous la forme du ratio :

$$P(X = l \,/\, Y \; = k) = \frac{n_{kl}}{n_{k.}}$$

Ils se présentent comme suit pour nos données « CSP Filières » :

CSP\Filière	Droit	Sciences	Médecine	IUT	Total
Exp.agri	0.265	0.328	0.215	0.192	1
Patron	0.292	0.238	0.362	0.108	1
Cadre.sup	0.258	0.219	0.480	0.043	1
Employé	0.310	0.285	0.289	0.116	1
Ouvrier	0.270	0.314	0.207	0.210	1
Total	0.272	0.254	0.373	0.101	1

Figure 100 – Profils lignes – Données "CSP Filières"

Nous lisons ainsi :

1. P(Sciences) = $\frac{962}{3784} = 25.4\%$, la proportion de personnes qui ont choisi la filière « Sciences ».

2. P(Sciences / Cadres.Sup) = $\frac{400}{1825} = 21.9\%$, la proportion de personnes qui ont choisi la filière « Sciences » parmi les enfants de « Cadres.Sup ».
3. P(Sciences / Ouvrier) = $\frac{193}{615} = 31.4\%$, la proportion de personnes qui ont choisi la filière « Sciences » parmi les enfants d'ouvriers. Etc.

Deux types de questions peuvent être posés au regard de ce mode de présentation des données : (1) Est-ce que les structures de choix sont les mêmes d'un CSP à l'autre, on parle de distances entre profils (section 4.1.2.2) ; (2) Est-ce que la structure de choix d'un CSP est différent de la structure globale, sans distinction de CSP, on parle de distance à l'origine (section 4.1.2.3).

Nous calculons ces proportions en ligne sous Python, que nous agrémentons à l'aide d'une représentation graphique en « barplot » empilés (Figure 101).

```
#profils lignes
prof_lig = numpy.apply_along_axis(arr=D.values,axis=1,func1d=lambda x:x/numpy.sum(x))
print(prof_lig)
```

```
[[0.26490066 0.32781457 0.21523179 0.19205298]
 [0.29217391 0.23826087 0.36173913 0.10782609]
 [0.25753425 0.21917808 0.48       0.04328767]
 [0.31049251 0.28479657 0.28907923 0.11563169]
 [0.2699187  0.31382114 0.20650407 0.2097561 ]]
```

```
#représentation graphique
import matplotlib.pyplot as plt
somme = numpy.zeros(shape=(prof_lig.shape[0]))
for i in range(prof_lig.shape[1]):
    plt.barh(range(prof_lig.shape[0]),prof_lig[:,i],left=somme)
    somme = somme + prof_lig[:,i]

plt.yticks(range(prof_lig.shape[0]),D.index)
plt.show()
```

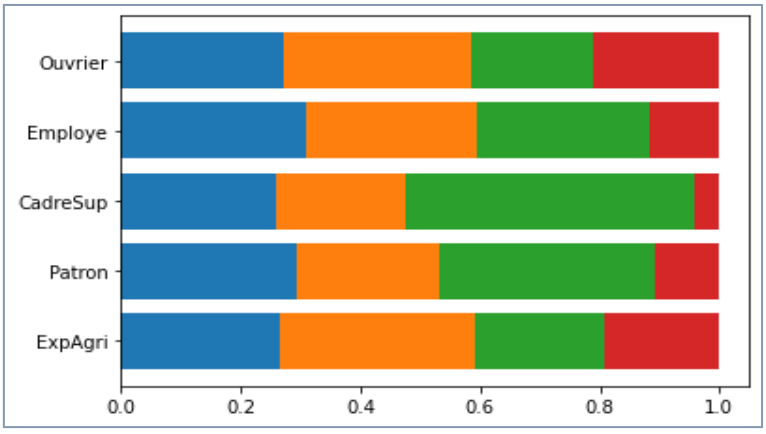


Figure 101 – Structure des choix de filières selon les CSP – Données "CSP – Filières"

Sans trop s'avancer, nous observons quand-même une préférence certaine des enfants de « Cadres.Sup » pour la « Médecine ». Par rapport aux autres catégories, les « ouvriers » et « exploitants agricoles » s'investissent plus dans les IUT. Etc. Le rôle de l'AFC justement sera de mettre en évidence les différences entre les profils et d'identifier les éventuelles accointances entre les CSP et les choix de filières.

4.1.2.2 Distance entre profils

Distance entre profils. La distance entre profils met en exergue les différences entre proportions en exacerbant celles qui ont trait aux modalités rares. Le même écart est d'autant plus marquant qu'il concerne une modalité peu représentée. On parle de distance du KHI-2 (χ^2) :

$$d^2(k,k') = \sum_{l=1}^{L} \frac{1}{\frac{n_{.l}}{n}} \left(\frac{n_{kl}}{n_{k.}} - \frac{n_{k'l}}{n_{k'.}} \right)^2$$

Voyons quelques exemples pour préciser les calculs :

$d^2(cadre, ouvrier)$

$$= \frac{1}{0.272}(0.258 - 0.270)^2 + \frac{1}{0.254}(0.219 - 0.314)^2 + \frac{1}{0.373}(0.480 - 0.207)^2$$
$$+ \frac{1}{101}(0.043 - 0.210)^2 = 0.5109$$

$d^2(cadre, patron)$

$$= \frac{1}{0.272}(0.258 - 0.292)^2 + \frac{1}{0.254}(0.219 - 0.238)^2 + \frac{1}{0.373}(0.480 - 0.362)^2$$
$$+ \frac{1}{101}(0.043 - 0.108)^2 = 0.0846$$

On en conclut que la structure des choix de filières des enfants de cadres est plus proche (similaire) de celle des cadres supérieurs qu'elle ne l'est de celle des enfants d'ouvrier.

Ces distances sont très faciles à produire sous Python, merci à la librairie Numpy.

```
#calul du profil marginal corresp. - ligne grisée de Figure 100
prof_marg_lig = tot_col/numpy.sum(tot_col)
print(prof_marg_lig)
```

```
[0.27193446 0.25422833 0.37288584 0.10095137]
```

```
#distance du KHI-2 entre cadre(2) et ouvrier(4)
print(numpy.sum((prof_lig[2,:]-prof_lig[4,:])**2/prof_marg_lig))

0.510900786777672

#distance du KHI-2 entre cadre(2) et patron(1)
print(numpy.sum((prof_lig[2,:]-prof_lig[1,:])**2/prof_marg_lig))

0.08461088232967051
```

Distances par paires des profils. Nous pouvons dès lors approfondir l'analyse en calculant les distances entre paires de modalités pour identifier les ressemblances et dissemblances.

```
#distance entre paires de modalités lignes
distPairesLig = numpy.zeros(shape=(prof_lig.shape[0],prof_lig.shape[0]))

#double boucle
for i in range(prof_lig.shape[0]-1):
    for j in range(i+1,prof_lig.shape[0]):
        distPairesLig[i,j] = numpy.sum((prof_lig[i,:]-prof_lig[j,:])**2/prof_marg_lig)
        #distPairesLig[j,i] = distPairesLig[i,j]

#affichage
print(pandas.DataFrame(distPairesLig,index=D.index,columns=D.index))
```

CSP_vs_Filiere	ExpAgri	Patron	CadreSup	Employe	Ouvrier
CSP_vs_Filiere					
ExpAgri	0.0	0.162117	0.453847	0.087400	0.004172
Patron	0.0	0.000000	0.084611	0.024514	0.191823
CadreSup	0.0	0.000000	0.000000	0.176847	0.510901
Employe	0.0	0.000000	0.000000	0.000000	0.115413
Ouvrier	0.0	0.000000	0.000000	0.000000	0.000000

Une représentation graphique sous forme de « heatmap » donne une vision globale des écarts.

```
#affichage sous forme de heatmap
import seaborn as sns
sns.heatmap(distPairesLig,vmin=0,vmax=numpy.max(distPairesLig),linewidth=0.1,cmap=
'Blues',xticklabels=D.index,yticklabels=D.index)
```

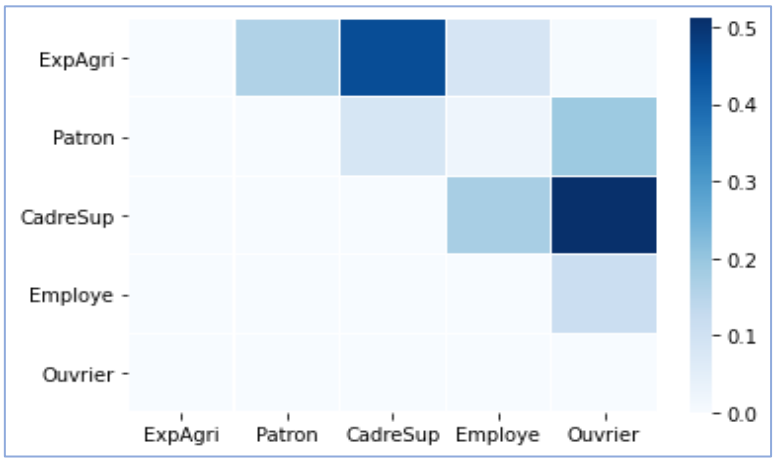


Figure 102 – Distances (au carré) entre les paires de modalités lignes – Données "CSP Filières"

Les enfants d'ouvriers et d'exploitants agricoles présentent des structures de choix assez proches (Figure 102). Il en est de même aussi, à un degré moindre cependant, entre les patrons et les employés. Finalement, il n'y a que les cadres supérieurs qui la ramènent avec des préférences qui les démarquent fortement des autres, en particulier des ouvriers et des exploitants agricoles.

Analyse factorielle des correspondances. Pour l'analyse des grands tableaux, lorsque le nombre de modalités lignes est élevé, comparer les profils (Figure 101) et inspecter les distances par paires (Figure 102) deviennent rapidement inextricables. L'AFC a pour objectif de produire un repère factoriel qui rend compte du positionnement relatif des modalités à l'aide de graphiques « nuages de points », en fournissant des indicateurs permettant d'apprécier la fidélité de la représentation par rapport à la matrice des distances initiale.

4.1.2.3 Distance à l'origine

Le profil marginal, la partie grisée dans le tableau de la Figure 100, correspond à la structure de choix dans distinction de CSP. Il représente le « profil moyen » c.-à-d. la moyenne pondérée des profils lignes. Par exemple, la P(droit) = $\frac{1029}{3784} = 0.272$ comme définit plus haut, mais elle peut être calculée également par la moyenne des probabilités conditionnelles associées :

$$P(droit) = \frac{302 \times 0.265 + 575 \times 0.292 + 1825 \times 0.258 + 467 \times 0.310 + 615 \times 0.270}{3784} = 0.272$$

Remarque : Cela veut dire aussi que si une modalité ligne est surreprésentée dans le fichier, elle pèsera énormément sur la définition du profil moyen.

La distance à l'origine est tout simplement la distance au profil moyen. Pour les enfants de cadres.sup par exemple :

$$d^2(cadre) = \frac{1}{0.272}(0.258 - 0.272)^2 + \frac{1}{0.254}(0.219 - 0.254)^2 + \frac{1}{0.373}(0.480 - 0.373)^2 + \frac{1}{0.101}(0.043 - 0.101)^2 = 0.0693$$

Nous obtenons le tableau suivant en calculant les distance à l'origine des CSP :

CSP\Filière	DISTO²
Exp.agri	**0.1703**
Patron	0.0033
Cadre.sup	0.0693
Employé	0.0301
Ouvrier	**0.2055**

Figure 103 – Distance à l'origine – AFC – Données "CSP Filières"

Nous remarquons par exemple que les enfants d'ouvriers et d'exploitants agricoles présentent des structures de choix (profils) les plus différentes de l'ensemble (la moyenne) des étudiants.

4.1.2.4 Inertie

L'inertie traduit la quantité d'information portée par une modalité. Elle est définie par le produit entre le poids de la modalité et sa distance à l'origine.

$$Inertie(y_k) = P(y_k) \times d^2(y_k)$$

Prenons l'exemple des CSP « cadres.sup » :

$$\begin{aligned} Inertie(cadre) &= P(cadre) \times d^2(cadre) \\ &= \frac{1825}{3784} \times 0.0693 \\ &= 0.4832 \times 0.0693 \\ &= 0.0334 \end{aligned}$$

Réalisons les calculs pour l'ensemble des modalités.

```
#distance à l'origine
distoLig = numpy.apply_along_axis(arr=prof_lig,axis=1,func1d=lambda x:numpy.sum((x
-prof_marg_lig)**2/prof_marg_lig))

#affichage
print(pandas.DataFrame(distoLig,index=D.index))

                        0
CSP_vs_Filiere
ExpAgri          0.170350
Patron           0.003311
CadreSup         0.069302
Employe          0.030113
Ouvrier          0.205492

#poids des lignes
poidsLig = tot_lig/numpy.sum(tot_lig)
```

```
#inertie des lignes
inertieLig = distoLig * poidsLig

#affichage
print(pandas.DataFrame(numpy.transpose([distoLig,poidsLig,inertieLig]),columns=['D
isto2','Poids','Inertie'],index=D.index))
```

	Disto2	Poids	Inertie
CSP_vs_Filiere			
ExpAgri	0.170350	0.079810	0.013596
Patron	0.003311	0.151956	0.000503
CadreSup	0.069302	0.482294	0.033424
Employe	0.030113	0.123414	0.003716
Ouvrier	0.205492	0.162526	0.033398

Nous notons par exemple que les enfants des exploitants agricoles ont certes un profil différent (distance à l'origine), mais sont peu représentés (poids), leur inertie ne se démarque pas vraiment. A contrario, les « cadres.sup » ne sont pas si différents, mais nombreux, ils pèseront beaucoup plus dans l'analyse.

Somme des inerties. La somme des inerties – l'inertie totale – représente la quantité d'information disponible dans les données. C'est un indicateur fondamental. Chaque facteur produit par l'AFC en représentera une partie.

Pour les données « CSP Filières », l'inertie totale – que nous noterons ϕ^2, nous comprendrons mieux plus loin – est égale à : $\phi^2 = 0.0846$.

```
#total inertie
tot_InertieLig = numpy.sum(inertieLig)
print(tot_InertieLig)
```

```
0.08463685828547157
```

4.1.2.5 Principe de l'analyse factorielle des correspondances – Version 1

Principe. L'AFC a pour mission de produire une successions de facteurs – des axes factoriels, combinaisons linéaires des profils – qui permettent de positionner les modalités de la manière la plus dispersée possible. Les facteurs sont deux à deux orthogonaux parce que le $q^{ème}$ facteur est élaboré à partir de la partie résiduelle des (q-1) qui le précèdent.

De ce point de vue, l'AFC se présente comme une ACP où les individus sont les modalités lignes, décrit par les profils lignes, et pondérés par les effectifs totaux des lignes.

Si on s'en tient au premier facteur (n°1), l'algorithme va maximiser la quantité λ_1 tel que :

$$\lambda_1 = \sum_{k=1}^{K} \frac{n_{k.}}{n} \times F_{k1}^2$$

- F_{k1} est la coordonnée de la modalité (y_k) sur le facteur n°1.
- La moyenne des points modalités est nulle c.-à-d. $(\sum_{k=1}^{K} F_{k1} = 0)$
- λ_1 représente par conséquent la variance des points modalités. Il s'agit de la valeur propre.

La quantité $\left(\frac{\lambda_1}{\phi^2}\right)$ indique la crédibilité du facteur. Il représente la part de d'inertie qu'il restitue.

Lorsque nous avons produit l'ensemble des facteurs, leur nombre maximal est égal à $\min(K-1, L-1)$, toute l'information est restituée c.-à-d.

$$\sum_{h=1}^{\min(K-1,\ L-1)} \lambda_h = \phi^2$$

Concrètement. Pour le facteur n°h, l'algorithme de l'AFC va produire un jeu de coefficients $(a_{h1}, a_{h2}, \ldots, a_{hL})$ – nous verrons comment dans la section 4.2 – qui permet de calculer la coordonnée de la modalité (y_k) à partir de son profil ligne. Pour le premier facteur par exemple, les coefficients fournis par l'AFC sont (0.098, 0.559, -1.056, 2.230). Sachant que les coordonnées – son profil ligne – de la modalité « Cadres.Sup » sont (0.258, 0.219, 0.480, 0.043) (Figure 100), sa coordonnée sur le premier facteur sera égale à :

$$0.098 \times 0.258 + 0.559 \times 0.219 - 1.056 \times 0.480 + 2.230 \times 0.043 = -0.263$$

Nous positionnons ainsi les différentes modalités lignes sur le premier facteur :

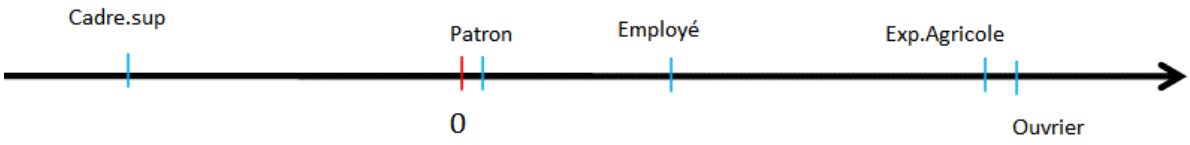


Figure 104 - Position des modalités lignes sur le 1er facteur de l'AFC - Données "CSP Filières"

Les positions relatives (Figure 104) corroborent ce que nous avions commencé à percevoir ci-dessus (Figure 101 et Figure 102) : les « Cadres.Sup » représentent un profil spécifique, différent (éloigné) des autres ; les enfants d'exploitants agricoles et d'ouvrier sont proches sur le 1er facteur, ils présentent des profils – structures de choix de filières – similaires.

Calculs sous Python. Nous réalisons une AFC avec le package « fanalysis » que nous présenterons plus en détail plus loin (section 4.3). Nous affichons les valeurs propres.

```
#importation de la librairie
from fanalysis.ca import CA

#lancer les calculs
afc = CA(row_labels=D.index,col_labels=D.columns)
afc.fit(D.values)

#information restituée sur les facteurs
print(afc.eig_)
```

```
[[8.23936026e-02 1.70344867e-03 5.39807038e-04]
 [9.73495522e+01 2.01265584e+00 6.37791913e-01]
 [9.73495522e+01 9.93622081e+01 1.00000000e+02]]
```

Nous observons pour chaque ligne de la matrice « eig_ »: les valeurs propres des [min(5-1, 4-1) = 3} facteurs, la fraction d'inertie restituée par facteur et cumulée. La variance du 1er facteur est égale à ($\lambda_1 = 0.08239$), elle représente déjà ($\frac{0.08239}{0.0846} = 97.35\%$) de l'inertie totale.

Nous affichons ensuite les coordonnées des modalités lignes par facteur.

```
#coordonnées des modalités lignes
print(pandas.DataFrame(afc.row_coord_,index=D.index))
```

```
                      0         1         2
CSP_vs_Filiere
ExpAgri        0.410115 -0.026253  0.038284
Patron         0.020151  0.026585 -0.046881
CadreSup      -0.262717 -0.015596  0.006199
Employe        0.142090  0.097326  0.021242
Ouvrier        0.451481 -0.039588 -0.009493
```

Pour le 1er facteur, vérifions que la moyenne pondérée des coordonnées est bien nulle.

```
#vérification de la moyenne pondérée des modalités - 1er facteur
print(numpy.sum(poidsLig * afc.row_coord_[:,0]))
```

```
6.938893903907228e-17
```

Et que sa variance est égale à la valeur propre.

```
#variance des modalités - 1er facteur
print(numpy.sum(poidsLig * afc.row_coord_[:,0]**2))

0.0823936025778933
```

Tout va pour le mieux.

Nous représentons les individus lignes dans le premier plan factoriel qui restitue **99.36%** de l'information disponible (Figure 105).

```
#affichage dans le premier plan factoriel
fig, ax = plt.subplots(figsize=(10,10))
ax.axis([-0.5,+0.5,-0.5,+0.5])
ax.plot([-0.5,+0.5],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-0.5,+0.5],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Carte des modalités lignes")

for i in range(D.shape[0]):
    ax.text(afc.row_coord_[i,0],afc.row_coord_[i,1],D.index[i])

plt.show()
```

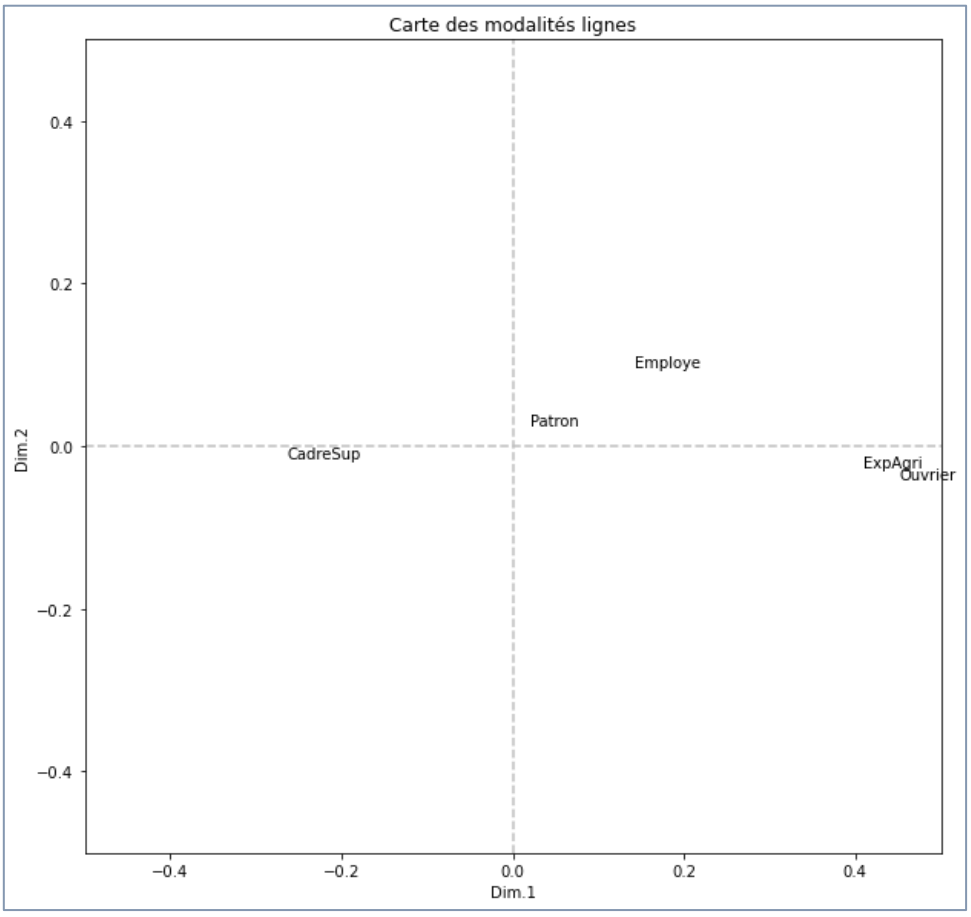


Figure 105 – Position des modalités lignes dans le plan – AFC – Données "CSP Filières"

En mettant la même échelle en abscisse et ordonnées, nous constatons que la différenciation des modalités lignes se joue quasi-exclusivement sur le premier axe factoriel.

4.1.2.6 Restitution de la distance entre profils

La distance du χ^2 entre les modalités lignes dans l'espace initial devient distance euclidienne dans l'espace factoriel. Et comme en ACP (concernant la distance entre individus), la restitution est parfaite lorsque nous utilisons tous les facteurs. Nous avons une approximation en revanche si nous n'en utilisons qu'une partie, sa précision dépend d'une part de la qualité de restitution du repère factoriel choisi, d'autre part de la qualité de représentation des modalités impliquées dans le calcul (on retrouve les fameux COS2, mais calculés différemment dans le cadre de l'AFC).

Pour les (q) premiers facteurs, la distance entre deux modalités lignes s'écrit :

$$d^2_{\{F_1,\dots,F_q\}}(k,k') = \sum_{h=1}^{q}(F_{kh} - F_{k'h})^2$$

Reprenons les distances de la section 4.1.2.2 sur le 1er facteur (F1) :

$$d^2_{\{F_1\}}(cadre, ouvrier) = (-0.26272 - 0.45148)^2 = 0.5101$$

$$d^2_{\{F_1\}}(cadre, patron) = (-0.26272 - 0.02015)^2 = 0.0800$$

Nous comparons ci-dessous les matrices des distances originelles et estimées :

CSP	Exp.agri	Patron	Cadre.sup	Employé	Ouvrier
Exp.agri	0	0.1621	0.4538	0.0874	0.0042
Patron		0	0.0846	0.0245	0.1918
Cadre.sup			0	0.1768	0.5109
Employé				0	0.1154
Ouvrier					0

Distance du KHI-2 entre modalités (tableau initial)

CSP	Exp.agri	Patron	Cadre.sup	Employé	Ouvrier
Exp.agri	0	0.1521	0.4527	0.0718	0.0017
Patron		0	0.0800	0.0149	0.1860
Cadre.sup			0	0.1639	0.5101
Employé				0	0.0957
Ouvrier					0

Distance euclidienne entre modalités (1er facteur)

Figure 106 - Comparaison des distances exactes et approchées entre modalités - Données "CSP Filières"

En fond bleu, les distances plutôt bien estimées, elles concernent les modalités situées aux extrémités du 1er facteur, celles qui sont bien représentées comme nous le constaterons plus loin (section 4.3.2). En fond rose, les distances mal restituées, elles concernent les modalités (les deux ou une des deux) proches de l'origine.

Sous Python, le calcul des distances entre modalités sur le 1er facteur ne pose aucune difficulté.

```
#distances euclidienne dans le 1er plan
distPairesLigF1 = numpy.zeros(shape=(prof_lig.shape[0],prof_lig.shape[0]))

#double boucle
for i in range(prof_lig.shape[0]-1):
    for j in range(i+1,prof_lig.shape[0]):
        distPairesLigF1[i,j] = numpy.sum((afc.row_coord_[i,0]-afc.row_coord_[j,0])**2)

#affichage
print(pandas.DataFrame(distPairesLigF1,index=D.index,columns=D.index))
```

```
CSP_vs_Filiere  ExpAgri    Patron  CadreSup   Employe   Ouvrier
CSP_vs_Filiere
ExpAgri            0.0  0.152072  0.452704  0.071837  0.001711
Patron             0.0  0.000000  0.080014  0.014869  0.186046
CadreSup           0.0  0.000000  0.000000  0.163869  0.510079
Employe            0.0  0.000000  0.000000  0.000000  0.095723
Ouvrier            0.0  0.000000  0.000000  0.000000  0.000000
```

4.1.3 Analyse des profils colonnes

4.1.3.1 Profils colonnes

Nous pouvons mener la même analyse sur les profils colonnes du tableau de données :

$$P(Y = k \,/\, X = l) = \frac{n_{kl}}{n_{.l}}$$

Les proportions se présentent comme suit :

CSP\Filière	Droit	Sciences	Médecine	IUT	Total
Exp.agri	0.078	0.103	0.046	0.152	0.080
Patron	0.163	0.142	0.147	0.162	0.152
Cadre.sup	0.457	0.416	0.621	0.207	0.482
Employé	0.141	0.138	0.096	0.141	0.123
Ouvrier	0.161	0.201	0.090	0.338	0.163
Total	1	1	1	1	1

Figure 107 – Tableau des profils colonnes – Données "CSP Filières"

Nous lisons entres autres :

1. P(Cadres.Sup) = 48.2%
2. P(Cadres.Sup / Droit) = 45.7%, etc.

La question en filigrane ici est : Observe-t-on les mêmes structures de CSP dans les filières ?

4.1.3.2 Distances entre profils, distance à l'origine, inertie

De fait, nous allons à l'essentiel dans cette section. La distance du χ^2 entre profils s'écrit :

$$d^2(l,l') = \sum_{k=1}^{K} \frac{1}{\frac{n_{k.}}{n}} \left(\frac{n_{kl}}{n_{.l}} - \frac{n_{kl'}}{n_{.l'}} \right)^2$$

Par exemple,

$$d^2(droit, sciences) = \frac{1}{0.080}(0.078 - 0.103)^2 + \frac{1}{0.152}(0.163 - 0.142)^2 + \cdots = 0.024$$

$$d^2(droit, médecine) = \frac{1}{0.080}(0.078 - 0.046)^2 + \frac{1}{152}(0.163 - 0.147)^2 + \cdots = 0.118$$

Le profil sociologique des étudiants en « droit » est plus proche de ceux en « sciences » qu'en « médecine ».

De la même manière que pour les profils lignes, nous pouvons calculer les distances à l'origine (représenté par la marge colonne du tableau de la Figure 107), les poids (des colonnes) et les inerties. Nous égrenons les calculs sous Python.

Nous calculons tout d'abord le profil « moyen » des filières.

```
#profil marginal des filières
prof_marg_col = tot_lig/numpy.sum(tot_lig)
print(prof_marg_col)
```

```
[0.07980973 0.1519556  0.48229387 0.12341438 0.16252643]
```

Puis les profils par filière.

```
#tableau des profils colonnes
prof_col = numpy.apply_along_axis(arr=D.values,axis=0,func1d=lambda x:x/numpy.sum(x))
print(pandas.DataFrame(prof_col,index=D.index,columns=D.columns))
```

```
                 Droit  Sciences  Medecine       IUT
CSP_vs_Filiere
ExpAgri       0.077745  0.102911  0.046067  0.151832
Patron        0.163265  0.142412  0.147413  0.162304
CadreSup      0.456754  0.415800  0.620836  0.206806
Employe       0.140914  0.138254  0.095677  0.141361
Ouvrier       0.161322  0.200624  0.090007  0.337696
```

Pour chaque filière, nous formons la distance à l'origine.

```
#distance**2 à l'orgine
distoCol = numpy.apply_along_axis(arr=prof_col,axis=0,func1d=lambda x:numpy.sum((x
-prof_marg_col)**2/prof_marg_col))

#affichage
print(pandas.DataFrame(distoCol,index=D.columns))

                0
Droit     0.004738
Sciences  0.027168
Medecine  0.092792
IUT       0.414466
```

La composition sociologique des étudiants en IUT est manifestement différente de la globalité. En comparant son profil avec le profil marginal, nous notons une surreprésentation des enfants d'exploitants agricoles et ouvrier, et une sous-représentation des cadres supérieurs.

Nous calculons ensuite le poids des filières.

```
#poids de chaque colonne
poidsCol = tot_col/numpy.sum(tot_col)
print(pandas.DataFrame(poidsCol,index=D.columns))

                0
Droit     0.271934
Sciences  0.254228
Medecine  0.372886
IUT       0.100951
```

Puis des inerties.

```
#inertie
inertieCol = distoCol*poidsCol
print(pandas.DataFrame(inertieCol,index=D.columns))

                0
Droit     0.001288
Sciences  0.006907
Medecine  0.034601
IUT       0.041841
```

Les filières IUT et Médecine sont celles qui pèseront le plus dans l'analyse, mais pas pour les mêmes raisons : la première parce que sa distance à l'origine est élevée, la seconde parce qu'elle rassemble un grand nombre d'étudiants (poids).

La somme des inerties des modalités colonnes (filières) est égale à $\phi^2 = 0.0486$,...

```
#somme des inerties
print(numpy.sum(inertieCol))

0.08463685828547156
```

… identique à la somme des inerties des modalités lignes (CSP). Ce n'est pas un hasard. Nous analysons le même tableau qui porte la même quantité totale d'information, mais avec un prisme différent, celui des filières.

4.1.3.3 Principe de l'analyse factorielle des correspondances – Version 2

Principe. L'objectif est toujours de produire une succession de facteurs, combinaisons linéaires des profils, qui étalent au mieux les modalités colonnes. Si on s'en tient au premier facteur, la variance restituée s'écrit :

$$\lambda_1 = \sum_{l=1}^{L} \frac{n_{.l}}{n} \times G_{l1}^2$$

Avec quelques éléments importants :

1. Les G_{l1} correspondent aux coordonnées des modalités colonnes (les filières) sur le 1er facteur.
2. Leur moyenne pondérée est nulle.
3. La variance λ_1 calculée ici coïncide avec celle calculée pour les modalités lignes, nous parlons bien du même facteur.

Nous reprenons le calcul avec la librairie « fanalysis », nous affichons les coordonnées des modalités colonnes (les filières) pour les 3 facteurs de l'AFC sur les données « CSP Filières ».

```
#coordonnées des filières
print(pandas.DataFrame(afc.col_coord_,index=D.columns))
```

```
                0         1         2
Droit     0.027987  0.060669 -0.016545
Sciences  0.160462  0.002734  0.037583
Medecine -0.303125 -0.029662 -0.005200
IUT       0.640174 -0.060749 -0.030870
```

Et nous passons à la représentation graphique dans le plan.

```
#affichage dans le premier plan factoriel
fig, ax = plt.subplots(figsize=(10,10))
ax.axis([-0.7,+0.7,-0.7,+0.7])
ax.plot([-0.7,+0.7],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-0.7,+0.7],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Carte des modalités colonnes")
```

```
for i in range(D.shape[1]):
    ax.text(afc.col_coord_[i,0],afc.col_coord_[i,1],D.columns[i])

plt.show()
```

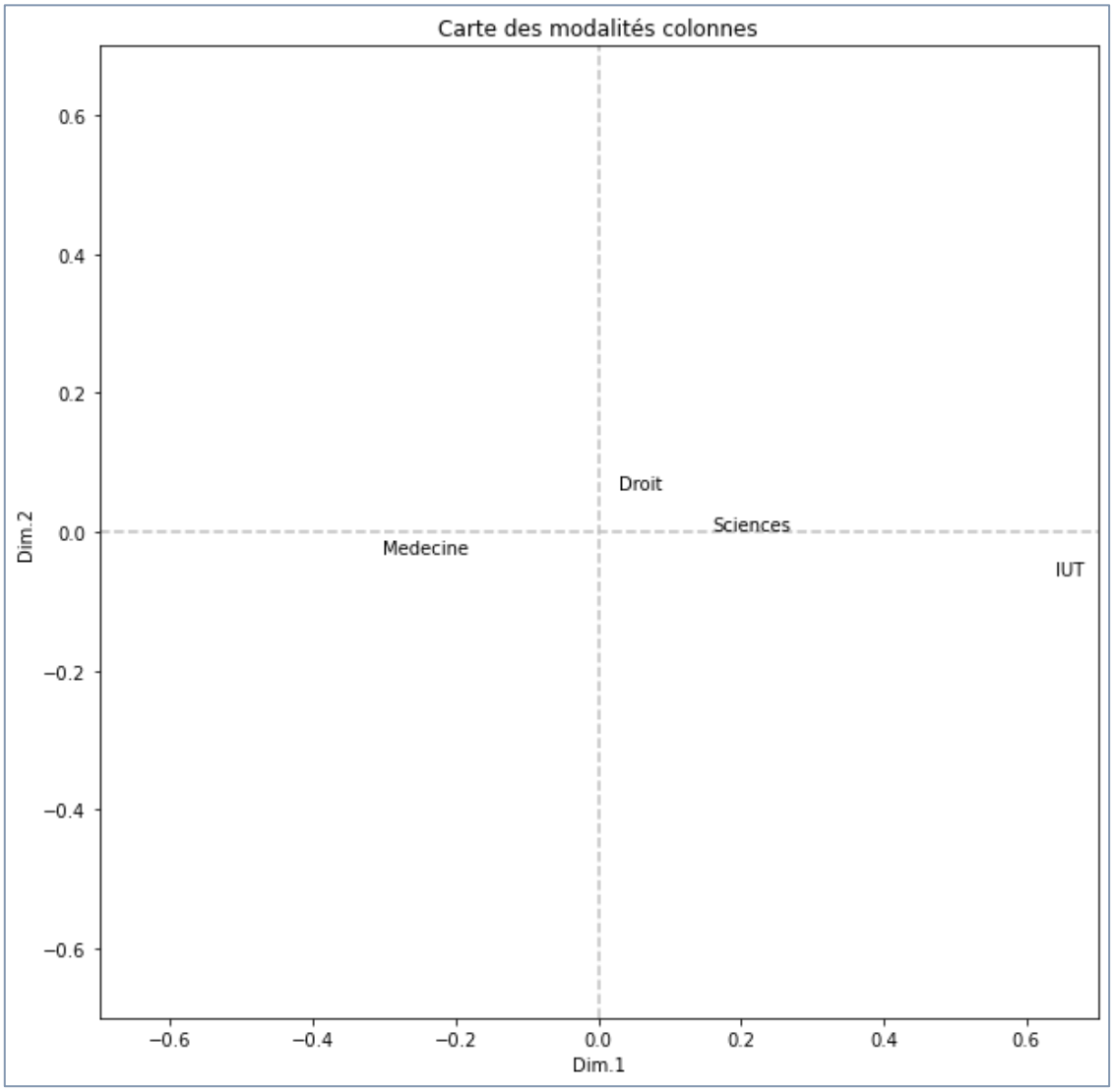


Figure 108 – Carte des modalités colonnes – 1er plan factoriel – Données "CSP Filières"

Les différenciations se jouent encore une fois essentiellement sur le 1er facteur. L'IUT présente visiblement un profil à part, à l'opposé de celui de « Médecine » notamment.

Remarque : Nous le verrons dans la section suivante, il est licite de fondre des cartes des modalités lignes et colonnes dans un seul graphique et, dans un sens que nous préciserons, d'interpréter les proximités.

4.1.4 Analyse de l'association lignes-colonnes

4.1.4.1 Statistique de l'écart à l'indépendance

Au-delà de la comparaison des profils, l'intérêt majeur de l'AFC est d'identifier les associations entre les modalités lignes et colonnes. Est-ce que les enfants de « Cadres.Sup » sont plus attiré par la « Médecine » que les autres ? Est-ce que les IUT exercent une attraction pour les enfants d'ouvriers ? Lesquels se détourneraient de la « Médecine » ? Etc.

Avant de s'intéresser aux relations entre les modalités, il faudrait déjà vérifier qu'il existe bien une liaison exploitable entre les variables en ligne et colonne du tableau de contingence. La statistique du χ^2 de l'écart à l'indépendance, dite de Pearson, est l'outil privilégié pour ce faire.

Le χ^2 de Pearson repose sur un mécanisme de test d'hypothèses. Il compare les effectifs observés (n_{kl}) avec les effectifs théoriques (e_{kl}) du tableau sous l'hypothèse (nulle) d'indépendance des variables (Rakotomalala R., « Dépendances des variables qualitatives », version 2.1, avril 2020). Dans ce cas, le contenu du tableau est entièrement défini par ses marges. En effet, sous H_0, la probabilité conjointe est égale au produit des probabilités marginales :

$$P(Y = y_k \,\&\, X = x_l) = P(Y = y_k) \times P(X = x_l)$$

Le même principe est appliqué aux effectifs :

$$e_{kl} = \frac{n_{k.} \times n_{.l}}{n}$$

Au tableau des effectifs observés (Figure 98), nous opposons donc le tableau des effectifs sous l'hypothèse nulle (Figure 109).

CSP\Filière	Droit	Sciences	Médecine	IUT	Total
Exp.agri	82.1	76.8	112.6	30.5	302
Patron	156.4	146.2	214.4	58.0	575
Cadre.sup	496.3	464.0	680.5	184.2	1825
Employé	127.0	118.7	174.1	47.1	467
Ouvrier	167.2	156.4	229.3	62.1	615
Total	1029	962	1411	382	3784

Figure 109 – Tableau des effectifs sous hypothèse d'indépendance – Données "CSP Filières"

La statistique de test s'écrit comme suit :

$$\chi^2 = \chi^2_{total} = \sum_{k=1}^{K}\sum_{l=1}^{L}\frac{(n_{kl} - e_{kl})^2}{e_{kl}}$$

Sous Ho, elle suit une loi du χ^2 à $[(K-1)\times(L-1)]$ degrés de liberté.

Avec Python, nous formons le tableau des effectifs théoriques :

```
#effectifs totaux
n = numpy.sum(D.values)

#tableau sous indépendance
E = numpy.dot(numpy.reshape(tot_lig,(5,1)),numpy.reshape(tot_col,(1,4)))/n
print(E)
```

```
[[ 82.12420719  76.7769556  112.6115222   30.48731501]
 [156.36231501 146.18128964 214.40935518  58.04704017]
 [496.28039112 463.9667019  680.51664905 184.23625793]
 [126.99339323 118.72463002 174.13768499  47.14429175]
 [167.23969345 156.35042283 229.32478858  62.08509514]]
```

Nous pouvons calculer la statistique de test et la probabilité critique :

```
#statistique du KHI-2
KHI2 = numpy.sum(((D.values-E)**2)/E)
print(KHI2)
```

```
320.2658717522244
```

```
#degré de liberté
ddl = (E.shape[0]-1)*(E.shape[1]-1)
print(ddl)
```

```
12
```

```
#librairie scipy pour calcul des CDF
import scipy

#p-value du test
print(1-scipy.stats.chi2.cdf(KHI2,ddl))
```

```
0.0
```

Le test conduit au rejet de l'hypothèse nulle (p-value ≈ 0). Manifestement, les variables CSP des parents et choix de filières sont fortement liées.

Remarque : On émet souvent des restrictions quant à la validité de ce test : il faut que 80% des cellules présentent en effectif théorique supérieur ou égal à 5, mais au-delà il faut surtout se rendre compte que les faibles valeurs de (e_{kl}) peut gonfler exagérément la valeur de la statistique de test ; le test conduit quasi-systématiquement au rejet de l'hypothèse

d'indépendance quand l'effectif total (n) est élevé, parce que les degrés de liberté ne tiennent pas compte de (n) justement.

4.1.4.2 Mesures dérivées du KHI-2

On sait que les valeurs de la statistique χ^2 sont bornées par :

$$0 \leq \chi^2 \leq n \times \min(K-1, L-1)$$

En pratique, on procède à des normalisations pour disposer d'une mesure qui ne dépend pas des effectifs :

- Le PHI-2 est le rapport entre le χ^2 et l'effectif total :

$$\phi^2 = \frac{\chi^2}{n} = \frac{320.3}{3784} = 0.0846$$

- Le t de Tschuprow qui est une normalisation du ϕ^2 par les degrés de liberté. Sous certaines conditions, il varie entre 0 et 1 :

$$t = \sqrt{\frac{\phi^2}{\sqrt{(K-1)(L-1)}}} = \sqrt{\frac{0.0846}{\sqrt{(5-1)(4-1)}}} = 0.1563$$

- Le v de Cramer, une autre normalisation qui, elle, varie toujours entre 0 et 1 :

$$v = \sqrt{\frac{\phi^2}{\min(K-1, L-1)}} = \sqrt{\frac{0.0846}{\min(5-1, 4-1)}} = 0.1680$$

L'indicateur qui nous intéresse au premier chef est le ϕ^2 dans cette section. Il est égal à l'inertie totale. Ainsi, l'information disponible dans le tableau, qui était présentée sous la forme d'une dispersion des modalités autour du profil moyen, peut être également perçue sous l'angle de l'association entre Y et X.

4.1.4.3 Résidus standardisés et contributions au KHI-2

L'information globale associant les variables lignes et colonnes peut être dispatchée en informations locales associant les modalités lignes et colonnes. Elles permettent d'approfondir la nature de la liaison. Parmi les différents indicateurs disponibles, nous nous intéressons au

résidu standardisé dans ce section. Il est souvent utilisé pour identifier le caractère significatif de l'écart à l'indépendance d'une case. Il s'écrit :

$$r_{kl} = \frac{n_{kl} - e_{kl}}{\sqrt{e_{kl}}}$$

Il suit très approximativement une loi normale centrée et réduite. La liaison entre deux modalités de Y et X s'écarte significativement de l'indépendance lorsque $|r_{kl}| > 1.96$ pour un test à 5%.

Mais le résidu standardisé nous intéresse surtout parce qu'il est directement impliqué dans le calcul de la solution l'analyse factorielle des correspondances (section 4.2), et parce qu'il permet de décomposer en éléments additifs le χ^2 global, on parle de contributions au χ^2.

En effet,

$$\sum_{k=1}^{K}\sum_{l=1}^{L} r_{kl}^2 = \chi^2$$

Et la contribution au χ^2 correspond à la fraction d'information portée par chaque case du tableau de contingence,

$$c_{kl} = \frac{r_{kl}^2}{\chi^2}$$

Pour les données « CSP Filières », voici le tableau des résidus standardisés (Figure 110) :

CSP\Filière	Droit	Sciences	Médecine	IUT
Exp.agri	-0.23	2.54	-4.49	4.98
Patron	0.93	-0.76	-0.44	0.52
Cadre.sup	-1.18	-2.97	7.49	-7.75
Employé	1.60	1.31	-2.97	1.00
Ouvrier	-0.10	2.93	-6.76	8.49

Figure 110 – Tableau des résidus standardisés – Données "CSP Filières"

Premier commentaire important, le résidu standardisé est signé. Nous observons une attraction entre les modalités lorsque $(r_{kl} > 0)$, une répulsion dans le cas inverse $(r_{kl} < 0)$.

Dans le cas des données « CSP Filières », nous observons notamment des attractions fortes – que nous avions confusément pressentis lors de l'étude des profils lignes et colonnes – entre (IUT & Ouvrier), (Médecine & Cadres.Sup) ; des répulsions tout aussi radicales entre (IUT & Cadres.Sup), (Ouvrier & Médecine).

Les contributions au χ^2 ne sont pas signés. Elles permettent de hiérarchiser les couples de modalités selon leur importance dans le tableau de contingence. Elles s'additionnent en ligne et en colonne pour identifier les modalités porteuses d'information. Nous les avons exprimées en pourcentages (Figure 111).

CSP\Filière	Droit	Sciences	Médecine	IUT	Total
Exp.agri	0.02%	2.01%	6.29%	7.75%	16.06%
Patron	0.27%	0.18%	0.06%	0.08%	0.59%
Cadre.sup	0.43%	2.75%	17.53%	18.77%	39.49%
Employé	0.80%	0.54%	2.75%	0.31%	4.39%
Ouvrier	0.00%	2.68%	14.26%	22.52%	39.46%
Total	1.52%	8.16%	40.88%	49.44%	

Figure 111 – Tableau des contributions – Données « CSP – Filières »

Les contributions sont forcément en adéquation avec les résidus standardisés. Mais nous observons de surcroit que les modalités les plus contributives en ligne sont « Cadres.Sup » et « Ouvrier », pour 78.95% ; en colonne, ce sont « Médecine » et « IUT » pour 90.32%.

4.1.4.4 Indice d'attraction et répulsion – Test du rapport de vraisemblance

L'indice d'attraction-répulsion est un autre indicateur qui permet d'approfondir la nature des relations entre les modalités lignes et colonnes du tableau. Il est défini par le ratio :

$$\frac{P(Y = y_k \,\&\, X = x_l)}{P(Y = y_k) \times P(X = x_l)}$$

En pratique, nous le calculons par le rapport entre les effectifs observés et théoriques :

$$i_{kl} = \frac{o_{kl}}{e_{kl}}$$

Si l'indice est supérieur à 1, nous avons une attraction ; s'il est inférieur, nous avons une répulsion entre les modalités.

Il est sous-jacent à la statistique du rapport de vraisemblance permettant d'éprouver la réalité de l'association entre les variables en ligne et colonne du tableau de contingence :

$$G = 2\sum_{k=1}^{K}\sum_{l=1}^{L} n_{kl} \times \ln(i_{kl})$$

Laquelle suit une loi du χ^2 à [(K-1)(L-1)] degrés de liberté sous l'hypothèse d'indépendance.

Voici le tableau des indices pour les données « CSP Filières » (Figure 112) :

CSP\Filière	Droit	Sciences	Médecine	IUT
Exp.agri	0.974	1.289	0.577	1.902
Patron	1.074	0.937	0.970	1.068
Cadre.sup	0.947	0.862	1.287	0.429
Employé	1.142	1.120	0.775	1.145
Ouvrier	0.993	1.234	0.554	2.078

Figure 112 – Tableau des indices d'attractions et répulsions – Données "CSP Filières"

Les informations sont de même nature et cohérentes avec le tableau des résidus standardisés. Le calcul de la statistique de test ne présente aucune difficulté sous Python. Les résultats sont très proches de la statistique du χ^2 de Pearson.

```
#tableau des indices d'attraction et répulsion
IAR = D.values/E

#stat. de test du rapport de vraisemblance
G = 2*numpy.sum(D.values*numpy.log(IAR))
print(G)

321.96365868622877

#degrés de liberté (4 x 3)
print(ddl)

12

#p-value du test du rapport de vraisemblance
print(1-scipy.stats.chi2.cdf(G,ddl))

0.0
```

Remarque 1 : En réalité, la statistique de Pearson est une approximation pratique du rapport de vraisemblance, mise en avant du temps où les opérations à l'aide des tables de logarithmes ou

des règles à calcul étaient malaisés. Finalement, la première s'est imposée auprès des praticiens, même aujourd'hui, alors que les ordinateurs sont légion.

Remarque 2 : L'intérêt de l'indice d'attraction-répulsion par rapport au résidu standardisé peut paraître très relatif. Les deux indicateurs sont le fruit de la confrontation entre les probabilités conjointes et les produits des probabilités marginales. Ils produisent des résultats aux lectures comparables. Son utilité sera plus évidente lorsque nous étudierons la reconstitution du tableau des indices à partir des coordonnées factorielles des modalités lignes et colonnes (section 4.1.4.6). L'indice est en relation directe avec les positions relatives et par rapport à l'origine des modalités dans le repère factoriel.

4.1.4.5 Représentation simultanée des profils – Relation de transition

3 points de vue pour la même analyse. L'analyse factorielle des correspondances permet de positionner les modalités lignes dans un repère factoriel. Il en est de même pour les modalités colonne. Nous avons vu aussi que les facteurs (F_h, G_h) avaient le même pouvoir de représentation. Nous irons plus loin dans cette section en affirmant qu'il s'agit en réalité des mêmes facteurs et qu'il est possible de projeter les modalités lignes et colonnes dans le même repère. Les proximités donnent alors des indications sur l'étude des attractions et répulsions, qui constitue le 3ème prisme de l'analyse du tableau de contingence.

Relation quasi-barycentrique entre les coordonnées factorielles. Les coordonnées factorielles des modalités lignes et colonnes sont liées par les relations de transition dites quasi-barycentriques. En effet, sur un facteur (F_h) quelconque, il est possible d'obtenir les coordonnées des modalités lignes (F_{kh}) à partir des modalités colonnes (G_{lh}) avec :

$$F_{kh} = \frac{1}{\sqrt{\lambda_h}} \sum_{l=1}^{L} \frac{n_{kl}}{n_{k.}} \times G_{lh}$$

Et inversement :

$$G_{lh} = \frac{1}{\sqrt{\lambda_h}} \sum_{k=1}^{K} \frac{n_{kl}}{n_{.l}} \times F_{kh}$$

La coordonnée d'un point modalité ligne (resp. colonne) est obtenue par la moyenne pondérée – par son profil – des coordonnées de l'ensemble des modalités colonne (resp. ligne), déflaté par la racine de la valeur propre c.-à-d. de la variance restituée par le facteur.

Exemple. Prenons un exemple pour le 1er facteur (F1) où ($\lambda_1 = 0.08239$), le profil – ligne – de choix de filières des enfants d'ouvrier (5ème modalité ligne) est (Figure 100) :

$$ouvrier = (0.270, 0.314, 0.207, 0.210)$$

Les coordonnées des modalités colonnes (G_1), les filières, sur ce 1er facteur est (Figure 108) :

$$G_1 = (0.02799, 0.16046, -0.30313, 0.64017)$$

La position de la modalité « Ouvrier » sur ce premier facteur est donc égal à :

$$F_{51} = \frac{1}{\sqrt{0.08239}}(0.270 \times 0.02799 + 0.314 \times 0.16046 + \cdots) = 0.45148$$

Nous retrouvons bien cette coordonnée des « Ouvrier » sur le 1er facteur de la carte des modalités lignes (Figure 105).

Proximités entre les modalités lignes et colonnes. Ainsi, le rapprochement des coordonnées des modalités lignes et colonnes est licite grâce à la relation quasi-barycentrique. Attention néanmoins, proximité ne veut pas dire automatiquement association. L'appréciation ne peut se faire que globalement c.-à-d. une modalité ligne doit être située par rapport à l'ensemble des modalités colonnes, et inversement.

Nous représentons les modalités lignes et colonnes sur le 1er facteur pour approfondir ces idées de positionnement global et de rapprochement (Figure 113).

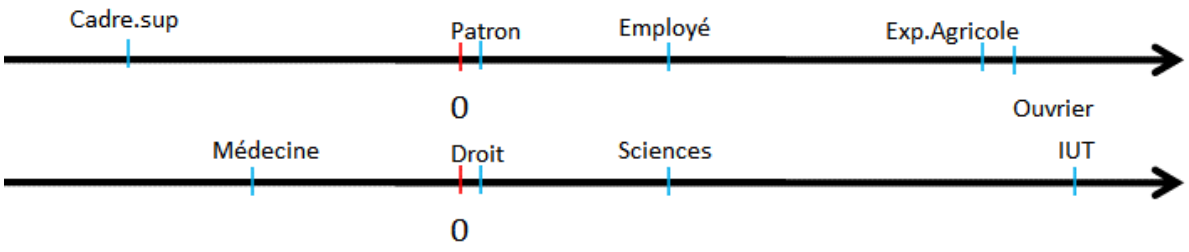


Figure 113 – Positions des modalités lignes et colonnes sur le 1er facteur – Données "CSP Filières"

Voyons deux exemples :

1. (IUT, Ouvrier) sont proches sur le 1er facteur. Est-ce à dire que les enfants d'ouvriers sont attirés par les IUT ? Réponse OUI. Parce que le point IUT est éloigné de l'ensemble modalités lignes sauf de « Ouvrier » (et « Exp.Agricole » aussi d'ailleurs). Conclure à une attraction est valable.
2. (Droit, Patron) sont proches sur le 1er facteur. Peut-on dire que les enfants de « Patron » sont attirés préférentiellement par le « Droit » ? Réponse NON. Parce que « Patron » est positionné quasiment au milieu de l'ensemble des filières. En réalité, ils (les enfants de « Patron ») ne sont attirés par aucune filière en particulier.



Et surtout, ces conclusions sont confirmées par le tableau des indices d'attraction-répulsion (Figure 112) – ou à défaut, le tableau des contributions au χ^2 – que l'on devrait toujours avoir sous la main quand on se lance dans l'interprétation des plans factoriels en AFC.

Représentation dans le plan. Forts de ces précautions, nous représentons les modalités lignes et colonnes des données « CSP Filières » dans le premier plan factoriel (Figure 114). J'ai cerclé à la main les attractions fortes (supérieures à 1.2 pour cet exemple).

```
#représentation simultanée
fig, ax = plt.subplots(figsize=(10,10))
ax.axis([-0.7,+0.7,-0.7,+0.7])
ax.plot([-0.7,+0.7],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-0.7,+0.7],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Carte des modalités lignes et colonnes")

#modalités ligne
for i in range(D.shape[0]):
    ax.text(afc.row_coord_[i,0],afc.row_coord_[i,1],D.index[i],color='blue')

#modalités colonne
for i in range(D.shape[1]):
    ax.text(afc.col_coord_[i,0],afc.col_coord_[i,1],D.columns[i],color='green')

plt.show()
```

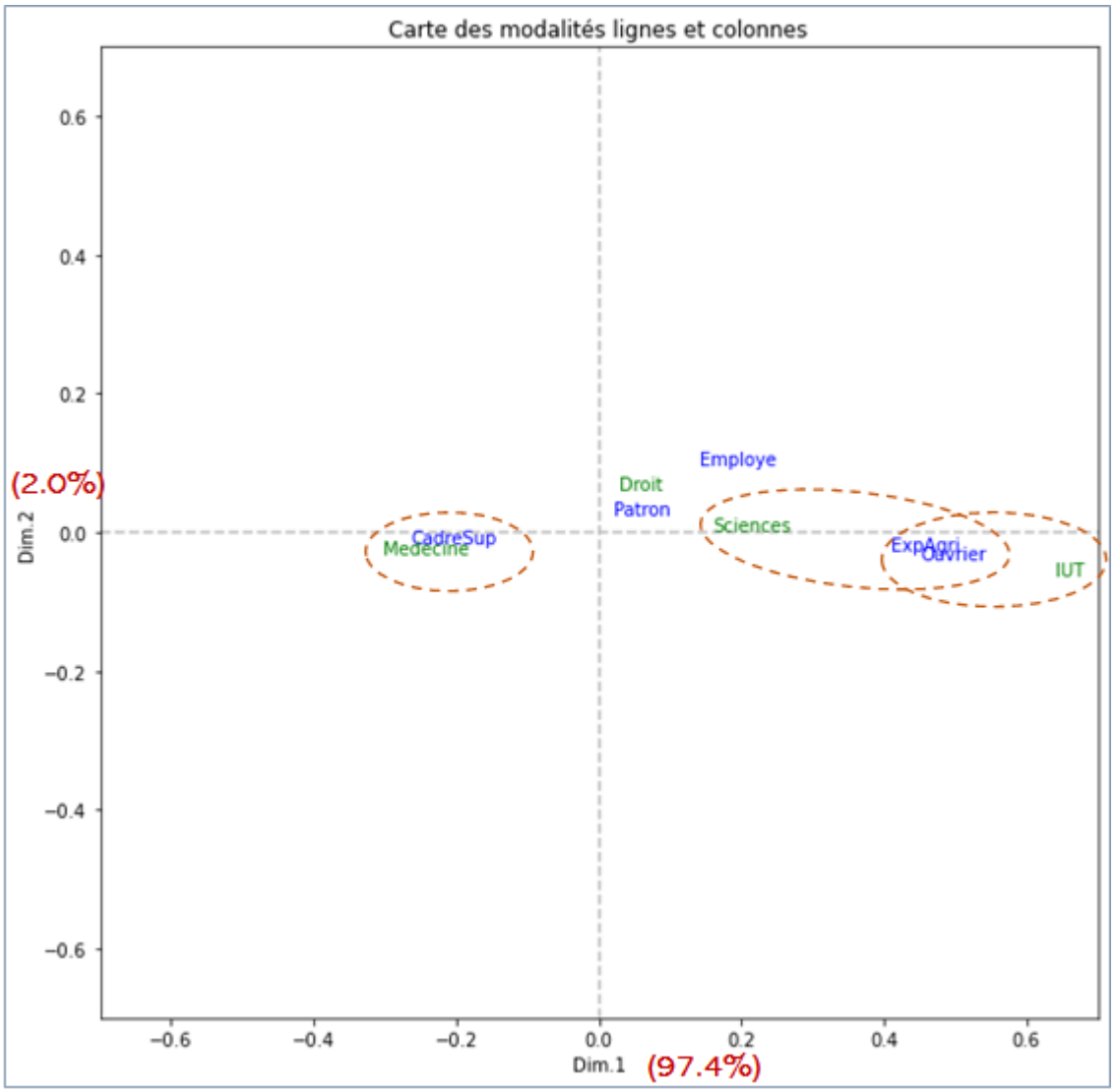


Figure 114 – Représentation simultanée dans le plan – Données "CSP Filières"

4.1.4.6 Restitution des indices d'attraction et répulsion

Comme en ACP où il est possible de reconstituer approximativement les données originelles à partir des coordonnées factorielles des individus (section 1.4), nous pouvons approcher le tableau des indices d'attraction-répulsion (i_{kl}) à partir des coordonnées factorielles des modalités lignes et colonnes. La formule s'écrit comme suit si l'on prend en compte les H premiers facteurs :

$$\hat{\imath}_{kl}^{(H)} = 1 + \sum_{h=1}^{H} \frac{F_{kh} \times G_{lh}}{\sqrt{\lambda_h}}$$

L'initiative de représenter les points lignes et colonnes dans le même repère s'en retrouve renforcée. En effet, nous lisons avec cette formule :

a. 2 modalités s'attirent (resp. se repoussent) si leurs coordonnées sont de même signe (resp. de signe contraire) sur les axes factoriels.
b. Le trait est d'autant plus marqué que les valeurs des coordonnées sont élevées (en valeur absolue) c.-à-d. que les points sont situés aux extrémités des facteurs.
c. Lesquelles coordonnées doivent être relativisées par le pouvoir de restitution du facteur (λ_h).

A l'instar de l'ACP, la reconstitution du tableau des indices est parfaite si nous utilisons tous les facteurs disponible [$H_{max} = \min(K-1, L-1)$].

Exemple. Pour les données « CSP Filières », si l'on s'en tient au 1er facteur qui capte déjà 97.4% de l'information disponible, nous calculons les indices d'attraction-répulsion estimés, et nous effectuons le parallèle avec le tableau initial. Nous constatons que l'approximation est déjà de très bonne tenue parce que le pouvoir de représentation du facteur est très bon (Figure 115).

Indice d'attraction - répulsion

CSP\Filière	Droit	Sciences	Médecine	IUT
Exp.agri	0.974	1.289	0.577	1.902
Patron	1.074	0.937	0.970	1.068
Cadre.sup	0.947	0.862	1.287	0.429
Employé	1.142	1.120	0.775	1.145
Ouvrier	0.993	1.234	0.554	2.078

Indices reconstitués à partir du 1er axe

CSP\Filière	Droit	Sciences	Médecine	IUT	Coord (F1)
Exp.agri	1.040	1.229	0.567	1.915	0.41012
Patron	1.002	1.011	0.979	1.045	0.02015
Cadre.sup	0.974	0.853	1.277	0.414	-0.26272
Employé	1.014	1.079	0.850	1.317	0.14209
Ouvrier	1.044	1.252	0.523	2.007	0.45148
Coord (G1)	0.02799	0.16046	-0.30313	0.64017	
Lambda.1	0.082394				

Figure 115 – Indices d'attraction-répulsion observés et reconstitués sur le 1er facteur

Le calcul est aisé sous Python. Nous affichons pour contrôle les indices calculés précédemment pour le test du rapport de vraisemblance (section 4.1.4.4). Nous affichons ensuite les indices estimés à l'aide de la formule ci-dessus.

```
#pour rappel - les indices observés
print(pandas.DataFrame(IAR,index=D.index,columns=D.columns))
```

```
                 Droit  Sciences  Medecine       IUT
CSP_vs_Filiere
ExpAgri       0.974134  1.289449  0.577206  1.902431
Patron        1.074428  0.937192  0.970107  1.068099
CadreSup      0.947045  0.862131  1.287257  0.428797
Employe       1.141792  1.120239  0.775249  1.145420
Ouvrier       0.992587  1.234407  0.553800  2.077793
```

```
#approximation des indices d'attraction-répulsion
estIAR = 1+numpy.dot(numpy.reshape(afc.row_coord_[:,0],(D.shape[0],1)),numpy.resha
pe(afc.col_coord_[:,0],(1,D.shape[1])))/numpy.sqrt(afc.eig_[0][0])
print(pandas.DataFrame(estIAR,index=D.index,columns=D.columns))
```

```
                  Droit  Sciences  Medecine       IUT
CSP_vs_Filiere
ExpAgri        1.039987  1.229261  0.566907  1.914655
Patron         1.001965  1.011265  0.978720  1.044941
CadreSup       0.974385  0.853137  1.277436  0.414078
Employe        1.013854  1.079431  0.849949  1.316895
Ouvrier        1.044020  1.252385  0.523224  2.006911
```

4.1.4.7 Décomposition orthogonale du KHI-2

L'AFC permet de décomposer la quantité totale d'information (χ^2_{total}) en éléments additifs. En effet, la reconstitution des données peut être étendue aux effectifs originels (n_{kl}). Si l'on prend en compte les H premiers facteurs, l'effectif estimé s'écrit :

$$\hat{n}_{kl}^{(H)} = \frac{n_{k.} \times n_{.l}}{n} \times \hat{\imath}_{kl}^{(H)}$$

$$\hat{n}_{kl}^{(H)} = e_{kl} \times \hat{\imath}_{kl}^{(H)}$$

Où (e_{kl}) est l'effectif sous l'hypothèse d'indépendance.

Nous pouvons dès lors calculer (1) l'information restituée par les H premiers facteurs...

$$\chi^2_{(H)} = \sum_k \sum_l \frac{\left(\hat{n}_{kl}^{(H)} - e_{kl}\right)^2}{e_{kl}}$$

... et (2) l'information résiduelle sur les derniers facteurs restants :

$$\chi^2_{residuel} = \sum_k \sum_l \frac{\left(n_{kl} - \hat{n}_{kl}^{(H)}\right)^2}{e_{kl}}$$

Ces deux résultats sont liés par la relation suivante :

$$\chi^2_{total} = \chi^2_{(H)} + \chi^2_{residuel}$$

On parle de décomposition orthogonale du χ^2.

Nous montrons le détail des calculs pour les données « CSP Filières » avec (H=1) premier facteur vs. les deux suivants dans notre support de cours consacré à l'AFC (COURS AFC, page 28). Pour varier les plaisirs, nous réalisons – sous Python – les manipulations pour (H = 2) premiers facteurs vs. la dernière dans ce support.

Pour rappel, nous affichons tout d'abord les effectifs (e_{kl}) sous Ho, les coordonnées factorielles des modalités lignes (F_{kh}) et colonnes (G_{lh}) dans le plan, les valeurs propres (λ_h), et enfin la quantité totale d'information (χ^2_{total}).

```
#effectifs sous H0
print(E)
```

```
[[ 82.12420719  76.7769556  112.6115222   30.48731501]
 [156.36231501 146.18128964 214.40935518  58.04704017]
 [496.28039112 463.9667019  680.51664905 184.23625793]
 [126.99339323 118.72463002 174.13768499  47.14429175]
 [167.23969345 156.35042283 229.32478858  62.08509514]]
```

```
#coordonnées des modalités lignes dans le plan
print(pandas.DataFrame(afc.row_coord_[:,:2],index=D.index))
```

```
                     0         1
CSP_vs_Filiere
ExpAgri          0.410115 -0.026253
Patron           0.020151  0.026585
CadreSup        -0.262717 -0.015596
Employe          0.142090  0.097326
Ouvrier          0.451481 -0.039588
```

```
#coordonnées des modalités colonnes dans le plan
print(pandas.DataFrame(afc.col_coord_[:,:2],index=D.columns))
```

```
                0         1
Droit     0.027987  0.060669
Sciences  0.160462  0.002734
Medecine -0.303125 -0.029662
IUT       0.640174 -0.060749
```

```
#les valeurs propres des 2 premiers facteurs
print(afc.eig_[0][:2])
```

```
[0.0823936  0.00170345]
```

```
#quantité totale d'information -- KHI2 total
print(KHI2)
```

```
320.2658717522244
```

Plusieurs étapes sont nécessaires pour obtenir les effectifs approchés du tableau de contingence.

```
#produit des coordonnées croisés dans le plan
#axe 1
s1 = numpy.dot(numpy.reshape(afc.row_coord_[:,0],(5,1)),numpy.reshape(afc.col_coor
d_[:,0],(1,4)))/numpy.sqrt(afc.eig_[0][0])
#axe 2
s2 = numpy.dot(numpy.reshape(afc.row_coord_[:,1],(5,1)),numpy.reshape(afc.col_coor
d_[:,1],(1,4)))/numpy.sqrt(afc.eig_[0][1])

#indice d'attraction répulsion dans le plan
IAR_plan = 1+(s1+s2)
print(IAR_plan)
```

```
[[1.00139614 1.22752211 0.58577439 1.95329696]
 [1.04104396 1.01302582 0.95961396 1.00581056]
 [0.95145948 0.85210367 1.28864471 0.43703345]
 [1.15691823 1.08587835 0.7800031  1.17364341]
 [0.98582719 1.24976275 0.5516749  2.06518005]]
```

```
#effectifs estimés dans le plan
est_n_plan = E*IAR_plan
print(est_n_plan)
```

```
[[ 82.23886415  94.24541071  65.96494546  59.55077968]
 [162.7800432  148.08542057 205.75021025  58.38432598]
 [472.19068402 395.34772898 876.94417993  80.51740706]
 [146.9209723  128.92050534 135.82793478  55.33058757]
 [164.86943632 195.4009344  126.51272957 128.2168997 ]]
```

Reste à calculer le $\chi^2_{(H)}$ restitué sur ces (H= 2) premiers facteurs...

```
#KHI2 dans le premier plan factoriel
KHI2plan = numpy.sum((est_n_plan-E)**2/E)
print(KHI2plan)
```

```
318.2232419212952
```

... et le $\chi^2_{residuel}$ sur le dernier.

```
#KHI2 résiduel
KHI2residuel = numpy.sum((D.values-est_n_plan)**2/E)
print(KHI2residuel)
```

```
2.0426298309291298
```

Nous effectuons la somme. Elle correspond bien à la quantité totale d'information véhiculée par les données.

```
#addition des deux = KHI2 total
print(KHI2plan+KHI2residuel)
```

```
320.2658717522243
```

4.2 Organisation des calculs

Mettre les mains dans le cambouis permet toujours de mieux appréhender la teneur des méthodes. Concernant l'AFC, une solution simple consiste à appliquer un programme d'ACP aux profils lignes (ou colonnes) en pondérant les modalités par les effectifs marginaux. Dans cette section, nous préférons mettre en exergue une implémentation qui met en lumière la recherche des associations entre les modalités lignes et colonnes inhérente à l'AFC.

Soit la matrice M avec :

$$M = \frac{1}{\sqrt{n}} R$$

Où R est la matrice des résidus standardisés dans ce contexte de l'AFC (section 4.1.4.3).

La solution de l'AFC correspond à la décomposition en valeurs singulières de M :

$$M = U \, \Delta \, V^T$$

Où :

- La matrice U, de dimension (K,K), contient les K vecteurs singuliers à gauche. U est orthonormée. Ils permettent d'obtenir les coordonnées factorielles des modalités lignes.
- Δ (K, L) est une matrice diagonale dont les éléments correspondent aux valeurs singulières.
- V (L, L) contient les vecteurs singuliers à droite. V est orthonormé.

Rappelons que la valeurs singulières (δ_h) sont définies comme suit :

$$\begin{cases} M \, v_h = \delta_h \, u_h \\ M^T \, u_h = \delta_h \, v_h \end{cases}$$

Nous percevons distinctement la nature croisée de l'analyse. Concrètement, la solution répond aux caractéristiques suivantes :

- Les dispersions des modalités lignes sont maximales sur les facteurs. Il en est de même pour les points colonnes.

- Les dispersions sont identiques pour les modalités lignes et colonnes.
- Les facteurs sont deux à deux orthogonaux.

Reste à retrouver les résultats usuels de l'AFC.

a. La variance restitue par les facteurs, les valeurs propres, correspondent au carré des valeur singulières, avec ($\lambda_h \leq 1$) (Lebart et al., 2000, pages 85 et 86)

$$\lambda_h = \delta_h^2$$

b. Les coordonnées des modalités lignes sont obtenues à partir des vecteurs de U

$$F_{kh} = \frac{u_{kh} \times \delta_h}{\sqrt{\frac{n_{k.}}{n}}}$$

c. Les coordonnées des modalités colonnes sont obtenues à partir des vecteurs de V

$$G_{lh} = \frac{v_{lh} \times \delta_h}{\sqrt{\frac{n_{.l}}{n}}}$$

Voyons tout cela sous Python. Nous essayons de retrouver à l'identique les résultats de l'objet « afc » de la librairie « fanalysis » que nous avions instancié plus haut (section 4.1.2.5). Nous calculons tout d'abord la matrice des résidus standardisés à partir des effectifs observés et théoriques (sous Ho), puis nous formons la matrice M.

```
#matrice des résidus standardisé
R = (D.values - E)/numpy.sqrt(E)
print(R)
```

```
[[-0.23440198  2.53622693 -4.48663409  4.98279921]
 [ 0.93068037 -0.75937762 -0.43771605  0.5188387 ]
 [-1.179691   -2.96968614  7.49359811 -7.7531439 ]
 [ 1.59786954  1.31013775 -2.96584705  0.99847632]
 [-0.09586159  2.93102443 -6.75702234  8.49237551]]
```

```
#matrice M
M = R/numpy.sqrt(n)
print(M)
```

```
[[-0.00381053  0.04122993 -0.07293653  0.08100239]
 [ 0.01512951 -0.01234475 -0.00711569  0.00843445]
 [-0.01917753 -0.04827641  0.12181894 -0.12603823]
```

```
 [ 0.02597561  0.02129813 -0.048214    0.01623163]
 [-0.00155836  0.04764791 -0.10984487  0.13805547]]
```

Nous faisons ensuite appel à la fonction svd() de la librairie « Numpy » pour effectuer la décomposition en valeurs singulières.

Valeurs propres. Nous affichons les valeurs singulières.

```
#décomposition en valeurs singulières
sol = numpy.linalg.svd(M)

#valeurs singulières
print(sol[1])

[2.87042858e-01 4.12728563e-02 2.32337478e-02 1.31134238e-17]
```

Seules les 3 premières valeurs sont non-nulles, conformément à ce qui est attendu [le nombre maximal de facteurs est égal à min(K-1, L-1) = min(5-1,4-1) = 3].

Nous les passons au carré et nous les comparons aux valeurs propres fournies par l'objet « afc » de la librairie « fanalysis ». Nous retrouvons les bons résultats.

```
#passés au carré
numpy.set_printoptions(suppress=True)
print(sol[1]**2)

[0.0823936  0.00170345 0.00053981 0.        ]

#valeurs propres de l'objet fanalysis
print(afc.eig_[0])

[0.0823936  0.00170345 0.00053981]
```

Coordonnées des modalités lignes. Passons maintenant aux vecteurs singuliers à gauche.

```
#vecteurs singuliers à gauche - les facteurs en colonne
print(sol[0])

[[ 0.40363357 -0.17969877  0.46550401 -0.75304822 -0.14495751]
 [ 0.02736552  0.25109394 -0.78656052 -0.55264881  0.11002184]
 [-0.63561972 -0.2624212   0.18528807 -0.28686474  0.64070246]
 [ 0.17390042  0.82841073  0.32118986  0.01236585  0.42447434]
 [ 0.63409564 -0.38669233 -0.1647239   0.21225781  0.61335504]]
```

Nous appliquons la formule pour produire les coordonnées factorielles des CSP, que nous confrontons avec ceux de « afc ».

```
#coordonnées sur les 3 premiers facteurs
F = numpy.apply_along_axis(arr=sol[0][:,:3]*sol[1][:3],axis=0,func1d=lambda x:x/nu
```

```
mpy.sqrt(poidsLig))
print(pandas.DataFrame(F,index=D.index))

                      0         1         2
CSP_vs_Filiere
ExpAgri         0.410115 -0.026253  0.038284
Patron          0.020151  0.026585 -0.046881
CadreSup       -0.262717 -0.015596  0.006199
Employe         0.142090  0.097326  0.021242
Ouvrier         0.451481 -0.039588 -0.009493

#coordonnées des modalités lignes -- objet "afc"
print(pandas.DataFrame(afc.row_coord_,index=D.index))

                      0         1         2
CSP_vs_Filiere
ExpAgri         0.410115 -0.026253  0.038284
Patron          0.020151  0.026585 -0.046881
CadreSup       -0.262717 -0.015596  0.006199
Employe         0.142090  0.097326  0.021242
Ouvrier         0.451481 -0.039588 -0.009493
```

Coordonnées des modalités colonnes. La matrice des vecteurs singuliers à droite est le dernier résultat à exploiter.

```
#vecteurs singuliers à droite - les facteurs en ligne
print(sol[2])

[[ 0.05084468  0.28186205 -0.64485656  0.70861035]
 [ 0.76654132  0.03340234 -0.43885508 -0.46765896]
 [-0.37134184  0.81560391 -0.13667626 -0.4221553 ]
 [ 0.52147336  0.5042106   0.61064379  0.31772846]]
```

Nous appliquons la formule pour obtenir les coordonnées des « Filières ». Encore une fois, nous comparons les résultats avec ceux de « afc ».

```
#calcul des coordonnées sur les 3 premiers facteurs
G = numpy.apply_along_axis(arr=numpy.transpose(sol[2][:3,:])*sol[1][:3],axis=0,fun
c1d=lambda x:x/numpy.sqrt(poidsCol))
print(pandas.DataFrame(G,index=D.columns))

                0         1         2
Droit     0.027987  0.060669 -0.016545
Sciences  0.160462  0.002734  0.037583
Medecine -0.303125 -0.029662 -0.005200
IUT       0.640174 -0.060749 -0.030870

#coordonnées des modalités colonnes -- objet "afc"
print(pandas.DataFrame(afc.col_coord_,index=D.columns))

                0         1         2
Droit     0.027987  0.060669 -0.016545
Sciences  0.160462  0.002734  0.037583
Medecine -0.303125 -0.029662 -0.005200
IUT       0.640174 -0.060749 -0.030870
```

Nous le constatons avec cet exemple, il est très facile d'implémenter l'AFC avec une bonne bibliothèque de calcul matriciel. Pour ma part, je m'étais beaucoup appuyé sur l'excellente librairie TPMATH de Jean Debord pour la programmation de la méthode dans TANAGRA.

4.3 Pratique de l'AFC avec « fanalysis » sous Python

Dans cette section, nous déroulons l'analyse complète en mettant en avant les fonctionnalités du package « fanalysis », les outils d'aides à l'interprétation qui permettent de mieux comprendre les résultats numériques fournis par l'AFC, la possibilité de projeter une ligne ou colonne supplémentaire dans le repère factoriel. Pour plus de clarté, nous recommençons entièrement le processus à partir de l'importation des données.

```
#chargement - index_col = 0 pour indiquer que la colonne n°0 est un label
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="AFC_ETUDES",in
dex_col=0)

#affichage des données
print(D)
```

```
                Droit  Sciences  Medecine  IUT
CSP_vs_Filiere
ExpAgri            80        99        65   58
Patron            168       137       208   62
CadreSup          470       400       876   79
Employe           145       133       135   54
Ouvrier           166       193       127  129
```

```
#nombre de modalités ligne
K = D.shape[0]

#nombre de modalités colonnes
L = D.shape[1]

#numpy
import numpy

#effectif total
n = numpy.sum(D.values)
print(n)
```

```
3784
```

```
#nombre max. de facteur
Hmax = numpy.min([K-1,L-1])
print(Hmax)
```

```
3
```

Le nombre maximal de facteurs est égal à $H_{max} = \min(K - 1, L - 1) = 3$.

Pour lancer les calculs, nous importons la classe « CA » de « fanalysis », nous l'instancions et nous passons le tableau à traiter à la méthode fit(). Nous affichons les propriétés de l'objet.

```
#importation de la librairie
from fanalysis.ca import CA

#lancer les calculs
afc = CA(row_labels=D.index,col_labels=D.columns)
afc.fit(D.values)
```

```
CA(col_labels=Index(['Droit', 'Sciences', 'Medecine', 'IUT'], dtype='object'),
   n_components=None, row_labels=Index(['ExpAgri', 'Patron', 'CadreSup', 'Employe'
, 'Ouvrier'], dtype='object', name='CSP_vs_Filiere'), stats=True)
```

```
#propriétés de l'objet
print(dir(afc))
```

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format
__', '__ge__', '__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__
', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__
reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setstate__', '__sizeof__
', '__str__', '__subclasshook__', '__weakref__', '_compute_stats', '_compute_svd',
'_get_param_names', '_get_tags', '_more_tags', 'c_', 'col_contrib_', 'col_coord_',
'col_cos2_', 'col_labels', 'col_labels_', 'col_labels_short_', 'col_topandas', 'ei
g_', 'fit', 'fit_transform', 'get_params', 'mapping', 'mapping_col', 'mapping_row'
, 'model_', 'n_', 'n_components', 'n_components_', 'plot_col_contrib', 'plot_col_c
os2', 'plot_eigenvalues', 'plot_row_contrib', 'plot_row_cos2', 'r_', 'row_contrib_
', 'row_coord_', 'row_cos2_', 'row_labels', 'row_labels_', 'row_topandas', 'set_pa
rams', 'stats', 'transform']
```

4.3.1 Détermination du nombre de facteurs

Les valeurs propres sont fournies de 3 manières par « fanalysis » : les valeurs brutes, en pourcentages, et en pourcentages cumulés.

```
#affichage des valeurs propres
print(afc.eig_)
```

```
[[8.23936026e-02 1.70344867e-03 5.39807038e-04]
 [9.73495522e+01 2.01265584e+00 6.37791913e-01]
 [9.73495522e+01 9.93622081e+01 1.00000000e+02]]
```

Règle de Kaiser. La règle de Kaiser est une approche simple pour identifier le nombre de facteurs pertinents. Elle peut s'appliquer sur les valeurs propres brutes, dans ce cas le seuil d'acceptation serait la moyenne des valeurs observées. Pour notre exemple « CSP Filières », il est égal à :

```
#seuil -- moyenne des valeurs propres
meanValPropre = numpy.mean(afc.eig_[0])
print(meanValPropre)
```

```
0.028212286095157187
```

La règle d'acceptation peut aussi s'appliquer aux pourcentages de variance restituée. Le seuil devient $(\frac{1}{H_{max}} \times 100)$, soit :

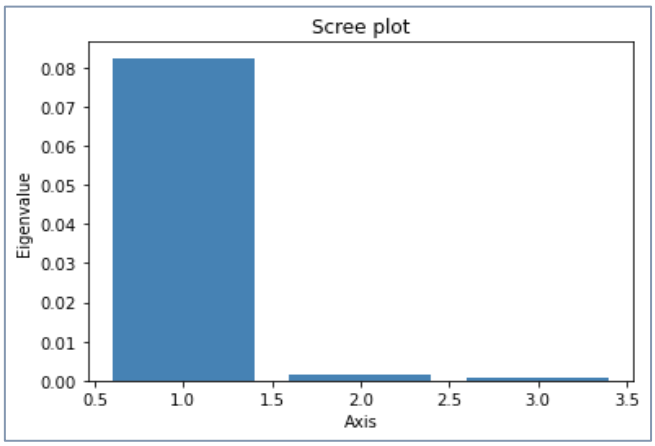
```
#ou seuil sur les pourcentages
print(1/Hmax * 100)

33.333333333
```

Ces deux approches sont totalement équivalentes. Seul le 1er facteur serait pertinent pour décrire nos données « CSP Filières ». Nous en conserverons 2 quand-même ne serait-ce que pour pouvoir réaliser les représentations graphiques.

Scree plot. L'éboulis des valeurs propres est de mise également en analyse factorielle des correspondances, avec la fameuse règle du coude.

```
#affichage graphique des v.p.
afc.plot_eigenvalues()
```



Pour notre exemple, le graphique n'est pas des plus déterminants au regard du faible nombre total de facteurs (H_{max} = 3) et l'écrasante influence du 1er.

Test statistique. Dans Saporta (2006, page 209) est décrit un test statistique pour tester la significativité des (q) derniers facteurs. Malheureusement (TUTO 15, section 3.4.2), surtout si nous travaillons sur des effectifs un tant soit peu élevés, il tend à sélectionner trop d'axes factoriels. Finalement, les méthodes empiriques restent les plus utilisés en pratique.

4.3.2 Coordonnées factorielles et aides à l'interprétation

Les informations sur les modalités lignes peuvent être regroupés dans une structure commune.

```
#informations sur les modalités lignes
print(afc.row_topandas())
```

```
                row_coord_dim1  row_coord_dim2  row_coord_dim3  \
CSP_vs_Filiere
ExpAgri               0.410115       -0.026253        0.038284
Patron                0.020151        0.026585       -0.046881
CadreSup             -0.262717       -0.015596        0.006199
Employe               0.142090        0.097326        0.021242
Ouvrier               0.451481       -0.039588       -0.009493

                row_contrib_dim1  row_contrib_dim2  row_contrib_dim3  \
CSP_vs_Filiere
ExpAgri                16.292006          3.229165         21.669399
Patron                  0.074887          6.304816         61.867745
CadreSup               40.401242          6.886489          3.433167
Employe                 3.024136         68.626434         10.316292
Ouvrier                40.207729         14.953096          2.713396

                row_cos2_dim1  row_cos2_dim2  row_cos2_dim3
CSP_vs_Filiere
ExpAgri              0.987350       0.004046       0.008604
Patron               0.122652       0.213489       0.663859
CadreSup             0.995936       0.003510       0.000554
Employe              0.670459       0.314556       0.014984
Ouvrier              0.991935       0.007627       0.000439
```

Coordonnées des points lignes. Nous savons comment ils sont calculés (section 4.2).

Contributions des modalités lignes aux facteurs – CTR. Elles qualifient l'influence relative des modalités dans la définition des axes factoriels. La contribution d'une modalité correspond à la fraction d'information qu'elle porte dans la variance restituée par le facteur (λ_h). Concrètement, pour la CTR de la modalité (k) sur le facteur (h) :

$$CTR_h(k) = \frac{\frac{n_{k.}}{n} F_{kh}^2}{\sum_{k=1}^{K} \frac{n_{k.}}{n} F_{kh}^2} = \frac{\frac{n_{k.}}{n} F_{kh}^2}{\lambda_h}$$

Les contributions des modalités sur un facteur s'additionnent. La somme intra-facteur de l'ensemble des modalités est égale à 100% (si on raisonne en pourcentages).

Pour notre exemple, nous constatons que les modalités « Cadres.Sup » et « Ouvrier » sont les plus déterminantes sur le 1er facteur, avec des rôles opposés puisqu'elles sont situées de part et

d'autre de l'origine. Pour le 2nd, « Employe » est important, mais n'oublions pas que ce facteur ne restitue que 2.01% de l'information disponible. Le résultat doit être fortement relativisé.

Pour réaliser les calculs sous Python, il nous faut tout d'abord produire le profil marginal des modalités lignes ($\frac{n_{k.}}{n}$) puis appliquer la formule. Nous retrouvons les valeurs de l'objet « afc ».

```
#profil marginal des modalités lignes
profMargLig = numpy.sum(D.values,axis=1)/n
print(profMargLig)

[0.07980973 0.1519556  0.48229387 0.12341438 0.16252643]

#contributions
contribLig = (numpy.reshape(profMargLig,(5,1))*afc.row_coord_**2)/afc.eig_[0]*100
print(contribLig)

[[16.2920062   3.22916478 21.66939863]
 [ 0.07488716  6.30481646 61.86774549]
 [40.40124242  6.88648879  3.43316701]
 [ 3.02413561 68.62643393 10.31629239]
 [40.20772861 14.95309604  2.71339648]]
```

Qualité de la représentation – Les COS2. Les COS2 indiquent la qualité de représentation individuelle et cumulée des points modalités sur les H premiers facteurs. Il représente l'information de la modalité rapportée par le ou les facteurs concernés. Il a le même numérateur que la CTR, mais normalisée cette fois-ci par l'inertie totale de la modalité (section 4.1.2.4). Il peut être éventuellement exprimé en pourcentages. La somme inter-facteur des COS2 d'une modalité sur l'ensemble des axes est égale à 100%. Pour le facteur n°h, le COS2 s'écrit :

$$COS_h^2(k) = \frac{\frac{n_{k.}}{n} F_{kh}^2}{\frac{n_{k.}}{n} d^2(k)} = \frac{F_{kh}^2}{d^2(k)}$$

La formule est en réalité simplifiée, elle revient à relativiser le carré de la coordonnée par la distance à l'origine (section 4.1.2.3).

Le COS2 est important pour l'appréciation des proximités entre les modalités homologues (soit lignes, soit colonnes). Lorsque deux modalités de la même variable sont proches et bien représentées (COS2 élevé), nous pouvons conclure à une similarité des profils. Si elles sont éloignées au contraire, on peut penser qu'ils sont dissemblables. Concernant notre exemple, « Ouvrier », « Cadres.Sup » et « Exp.Agri » sont très bien représentées sur le 1er facteur. Leur

positionnement (Figure 105) donne une indication assez précise sur les similitudes et disparités entre les profils (Figure 101, et Figure 102 pour le calcul des distances entre profils).

Le cas de « Exp.Agri » attire notre attention car il nous éclaire sur la différence de nature entre COS2 et CTR. La modalité est très bien représentée (COS2 élevé) parce qu'elle est quasiment située sur l'axe F1 (coordonnée sur F2 proche de 0) ; mais sa contribution (CTR) est moindre, par rapport à « Ouvrier » notamment, parce que son poids relatif est nettement moindre.

Pour reproduire les COS2 de l'objet « afc » sous Python, nous calculons donc tout d'abord les distances à l'origine puis nous appliquons la formule.

```
#distance à l'origine - distance du KHI2
distoLig = numpy.sum(afc.row_coord_**2,axis=1)
print(distoLig)

[0.17034955 0.00331062 0.0693019  0.03011317 0.2054925 ]

#cos2 des lignes
cos2Lig = afc.row_coord_**2/numpy.reshape(distoLig,(5,1))*100
print(cos2Lig)

[[9.87350266e+01 4.04596803e-01 8.60376599e-01]
 [1.22651866e+01 2.13488665e+01 6.63859469e+01]
 [9.95935831e+01 3.50970036e-01 5.54468168e-02]
 [6.70459405e+01 3.14556176e+01 1.49844193e+00]
 [9.91934675e+01 7.62676235e-01 4.38562850e-02]]
```

Informations sur les modalités colonnes. Le principe et les indicateurs sont les mêmes pour les modalités colonnes.

```
#statistiques pour les points colonnes
print(afc.col_topandas())

          col_coord_dim1  col_coord_dim2  col_coord_dim3  col_contrib_dim1  \
Droit           0.027987        0.060669       -0.016545          0.258518
Sciences        0.160462        0.002734        0.037583          7.944621
Medecine       -0.303125       -0.029662       -0.005200         41.583998
IUT             0.640174       -0.060749       -0.030870         50.212862

          col_contrib_dim2  col_contrib_dim3  col_cos2_dim1  col_cos2_dim2  \
Droit            58.758560         13.789476       0.165328       0.776896
Sciences          0.111572         66.520974       0.947735       0.000275
Medecine         19.259378          1.868040       0.990227       0.009482
IUT              21.870490         17.821510       0.988797       0.008904

          col_cos2_dim3
Droit          0.057776
Sciences       0.051990
Medecine       0.000291
IUT            0.002299
```

4.3.3 Représentation simultanée

Nous représentons les modalités lignes et colonnes dans le premier plan factoriel. L'instruction tient en une ligne de commande avec « fanalysis ».

```
#représentation simultanée
afc.mapping(num_x_axis=1,num_y_axis=2,figsize=(7,7))
```

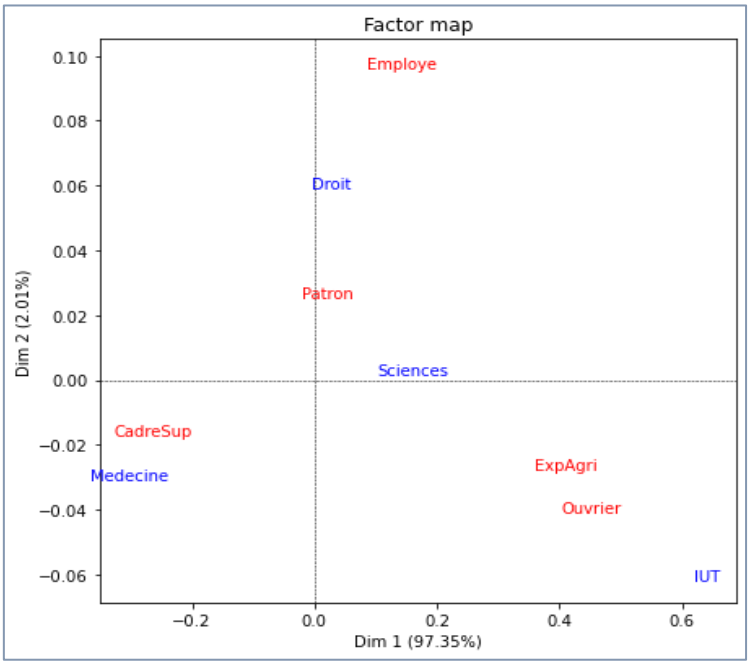


Figure 116 – Représentation simultanée "fanalysis" – AFC – Données "CSP Filières"

Attention. L'outil graphique ajuste automatiquement les échelles en abscise et ordonnées. Les indications sur les parts d'inerties restituées sur les axes sont fondamentales pour une bonne lecture des positions relatives et des proximités entres modalités lignes et colonnes, lesquelles doivent toujours s'apprécier globalement rappelons-le.

Tableau des contributions et/ou des indices d'attraction-répulsion. La librairie « fanalysis » ne les propose pas mais, je le répète encore une fois, la lecture des résultats d'une AFC lorsqu'on se lance dans l'étude des associations entre modalités lignes et colonnes est périlleuse. Elle doit être accompagnée par, au choix : le tableau des résidus standardisés, celui des contributions au χ^2, ou celui des indices d'attraction-répulsion. En effet, les proximités sont parfois trompeuses en AFC, guider notre interprétation par les données sous-jacentes aux représentations est un garde-fou forcément salutaire.

4.3.4 Traitement des lignes-colonnes supplémentaires

Coordonnées. Même s'il est moins mis en lumière dans la littérature, le mécanisme des modalités supplémentaires existe pour l'analyse factorielle des correspondances. Il permet déjà de prendre en compte les profils atypiques qu'on a dû écarter de la construction des facteurs parce qu'ils risquaient de fausser les calculs. Il permet aussi de situer les modalités particulières, de nature différente, ou dont on ne maîtrise pas très bien le statut.

Dans cette section, nous essayons de situer les choix de filières des enfants de « Bourgeois ». Nous sommes d'accord, « Bourgeois » n'est pas une CSP. Il s'agirait d'une catégorie sociale aux contours assez flous, mais qu'on identifie quand-même assez bien lorsque nous les croisons. L'introduire parmi les modalités actives n'avait pas de sens à cause de sa définition incertaine. En revanche, après coup, nous pouvons analyser son positionnement par rapport aux autres catégories (CSP) qui, elles, ont une définition reconnue.

Remarque : Nous décrivons la procédure pour une modalité ligne ici, mais la démarche est très facilement généralisable aux modalités colonnes supplémentaires.

La modalité supplémentaire « Bourgeois » (choix d'études des enfants de « Bourgeois ») se présente comme une ligne additionnelle dans le tableau de contingence (Figure 117).

CSP\Filière	Droit	Sciences	Médecine	IUT	Total
Exp.agri	80	99	65	58	**302**
Patron	168	137	208	62	**575**
Cadre.sup	470	400	876	79	**1825**
Employé	145	133	135	54	**467**
Ouvrier	166	193	127	129	**615**
Bourgeois	638	537	1084	141	**2400**

Figure 117 - Effectifs de la modalité supplémentaire "Bourgeois" - Données "CSP Filières"

Nous utilisons la relation quasi-barycentrique (section 4.1.4.5) pour positionner cette modalité ligne supplémentaire dans l'espace factoriel. Nous avons besoin pour ce faire de sa description sous forme de profil ligne (Figure 118)…

CSP\Filière	Droit	Sciences	Médecine	IUT	Total
Exp.agri	0.265	0.328	0.215	0.192	**1**
Patron	0.292	0.238	0.362	0.108	**1**
Cadre.sup	0.258	0.219	0.480	0.043	**1**
Employé	0.310	0.285	0.289	0.116	**1**
Ouvrier	0.270	0.314	0.207	0.210	**1**
Bourgeois	0.266	0.224	0.452	0.059	**1**

Figure 118 – Profils lignes, dont celui de la modalité supplémentaire – Données "CSP Filières"

... et des coordonnées factorielles des modalités colonnes (Figure 119).

	Lambda.1	Lambda.2
	0.0824	0.001703
Values	**Coord.1**	**Coord.2**
Droit	0.0280	-0.0607
Sciences	0.1605	-0.0027
Médecine	-0.3031	0.0297
IUT	0.6402	0.0608

Figure 119 – Coordonnées factorielles (F1, F2) des modalités colonnes – Données "CSP Filières"

La coordonnée de la modalité supplémentaire (y_{k^*}) sur le facteur (F_h) est obtenue avec :

$$F_{k^*h} = \frac{1}{\sqrt{\lambda_h}} \sum_{l=1}^{L} \frac{n_{k^*l}}{n_{k^*.}} \times G_{lh}$$

Par exemple, sur le premier facteur (F1), la coordonnée est égale à :

$$F_{Bourgeois,1} = \frac{1}{\sqrt{0.0824}} (0.266 \times 0.0280 + \cdots + 0.059 \times 0.6402) = -0.1949$$

Sous « fanalysis », l'opération est encapsulée dans la fonction transform(). Nous créons un vecteur de description de la modalité supplémentaire et nous la passons à la fonction.

```
#point supplémentaire – vecteur de drerscription
bourgeois = numpy.array([638,537,1084,141])

#coordonnées factorielles (F1, F2, F3)
coordBourges = afc.transform(numpy.reshape(bourgeois,(1,4)))
print(coordBourges)

[[-0.19494662 -0.0054899  -0.0065181 ]]
```

Nous insérons la modalité « Bourgeois » dans le premier plan factoriel à l'aide de ces coordonnées (Figure 120).

```
#Librairie graphique
import matplotlib.pyplot as plt

#reste à l'ajouter dans le plan factoriel
fig, ax = plt.subplots(figsize=(10,10))
ax.axis([-0.7,+0.7,-0.7,+0.7])
ax.plot([-0.7,+0.7],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-0.7,+0.7],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Modalité ligne supplémentaire")

#modalités ligne
for i in range(D.shape[0]):
    ax.text(afc.row_coord_[i,0],afc.row_coord_[i,1],D.index[i],color='red')

#modalités colonne
for i in range(D.shape[1]):
    ax.text(afc.col_coord_[i,0],afc.col_coord_[i,1],D.columns[i],color='blue')

#point supplémentaire
ax.text(coordBourges[0][0],coordBourges[0][1],"Bourgeois",color='green')

plt.show()
```

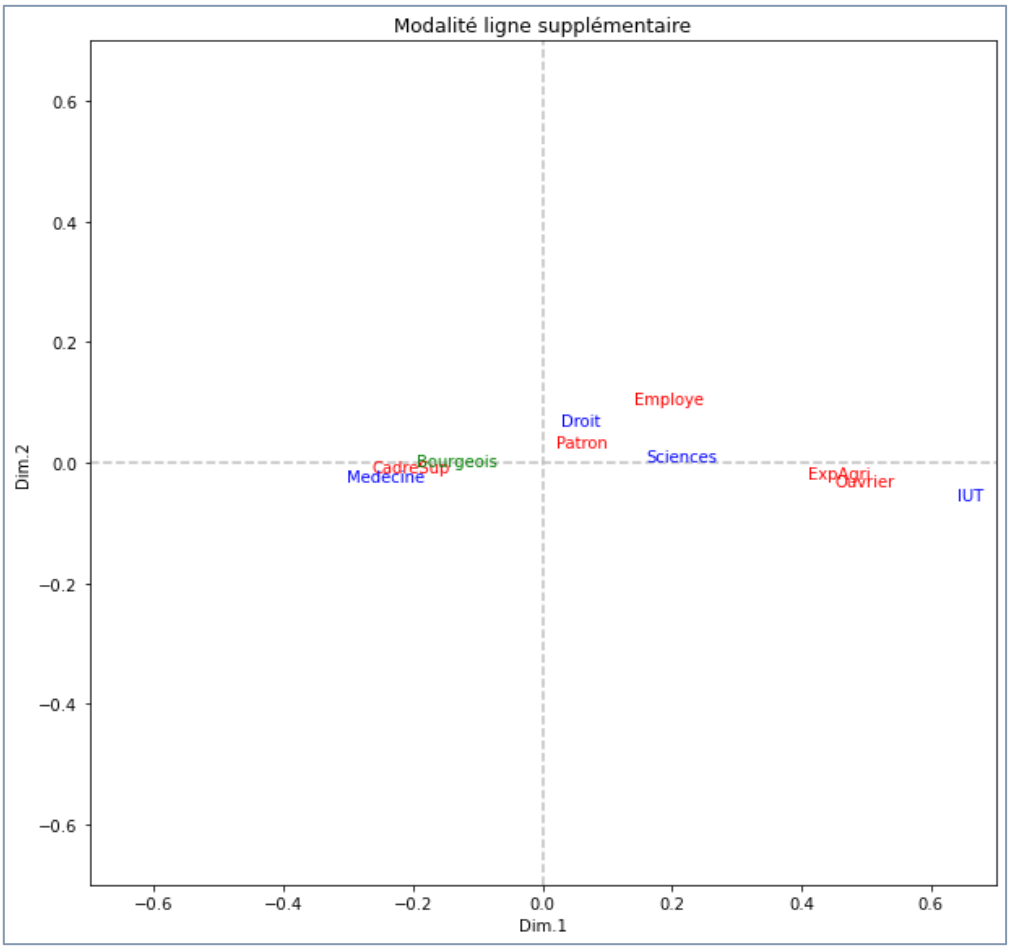


Figure 120 – Position de la modalité supplémentaire dans le plan factoriel – Données "CSP Filières"

Que le profil de choix d'études des enfants de bourgeois soient proches de celui des cadres supérieurs n'est pas fait pour me surprendre j'avoue.

Qualité de représentation – COS2. Calculer la contribution aux facteurs n'a pas de sens pour une modalité supplémentaire. En revanche, nous pouvons évaluer la qualité de sa représentation sur les facteurs (individuellement ou cumulés). Nous appliquons la formule de la section 4.3.2 : le COS2 d'une modalité sur un facteur est défini par le rapport entre le carré de sa coordonnée et sa distance (au carré) à l'origine (au profil marginal).

Sous Python, nous calculons le profil marginal (moyen), puis le profil des « Bourgeois », pour produire la distance à l'origine. Il reste à appliquer la définition du COS2. Attention, comme la modalité n'a pas participer à la construction des facteurs, la somme inter-facteur des COS2 n'est pas forcément égale à 1 (ou 100% si on travaille en pourcentages).

```
#profil marginal des modalités colonnes
profMargCol = numpy.sum(D.values,axis=0)/n
print(profMargCol)

[0.27193446 0.25422833 0.37288584 0.10095137]

#profil des bourgeois
profBourges = bourgeois/numpy.sum(bourgeois)

#distance à l'origine de la modalité supplémentaire
distoBourges = numpy.sum(1/profMargCol*(profBourges-profMargCol)**2)
print(distoBourges)

0.03807680914376361

#COS2 de la représentation
cos2Bourges = (coordBourges**2)/distoBourges
print(cos2Bourges)

[[9.98092681e-01 7.91530897e-04 1.11578809e-03]]

#soit pour récapituler
print(pandas.DataFrame(numpy.transpose([coordBourges[0],cos2Bourges[0]]),index=['D
im.1','Dim.2','Dim.3'],columns=['Coord','Cos2']))

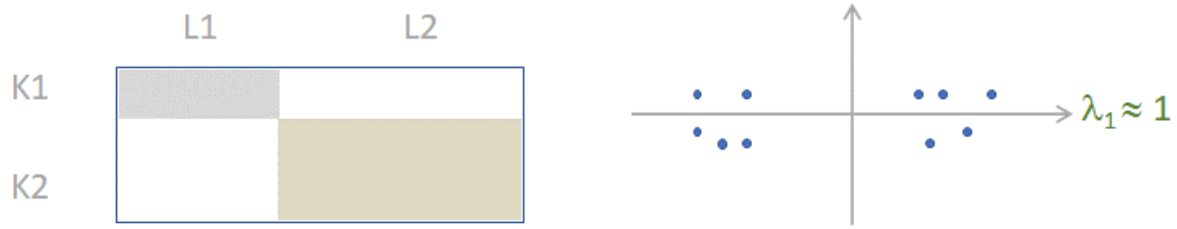
          Coord      Cos2
Dim.1 -0.194947  0.998093
Dim.2 -0.005490  0.000792
Dim.3 -0.006518  0.001116
```

La modalité « Bourgeois » est très bien représentée sur le 1er facteur. On pouvait le deviner d'emblée dans le graphique précédent (Figure 120), la modalité est quasiment située sur l'axe.

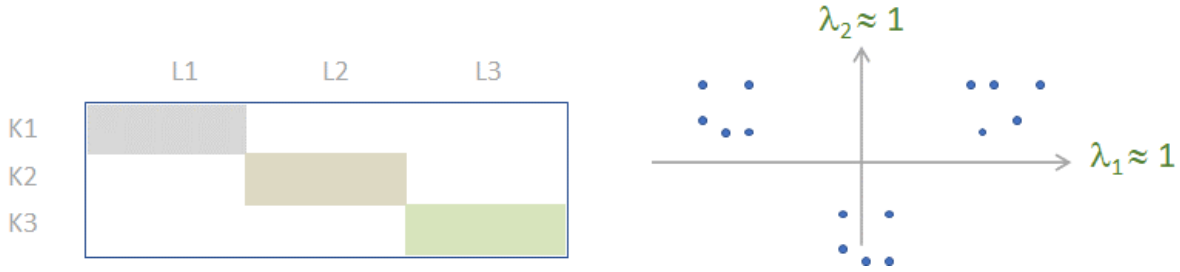
4.4 Quelques formes caractéristiques

Les nuages de points modalités peuvent présenter des formes caractéristiques qu'il faut savoir reconnaître en AFC parce qu'elles correspondent à des configurations particulières du tableau de contingence (Lebart et al., 2000, pages 92 à 94 ; Tenenhaus, 2007, pages 241 à 248).

– **Deux blocs distincts dans le tableau** après réorganisation selon la position des modalités sur le premier facteur. Les points modalité sont disposés en 2 groupes. Il est peut-être nécessaire de scinder les données en deux parties pour mener des analyses distinctes.



– **Des blocs distincts dans le tableau** (après réorganisation). L'idée d'analyses distinctes reste de mise.



– **Diagonale chargée** (après réorganisation). La forme parabolique du nuage est caractéristique de que ce qu'on appelle « Effet Guttman ». Il y a une relation non-linéaire entre les facteurs.



Ce type de configuration survient surtout lorsqu'il y a un ordonnancement sous-jacent aux modalités lignes et/ou colonnes. Nous l'avions étudié en détail dans un tutoriel (TUTO 12) où nous mettions en relation le niveau d'études (variable en ligne) de personnes et le type d'emploi (colonne) qu'ils occupent. Il y avait clairement un ordre sur les modalités de la première variable qui correspondent simplement au nombre d'années d'études. Il est moins évident pour la seconde. Mais on peut imaginer que les différents types d'emploi requièrent une qualification en lien avec les études. Les positions des points-modalités dans le premier plan factoriel sont très caractéristiques de l'effet Guttman (Figure 121).

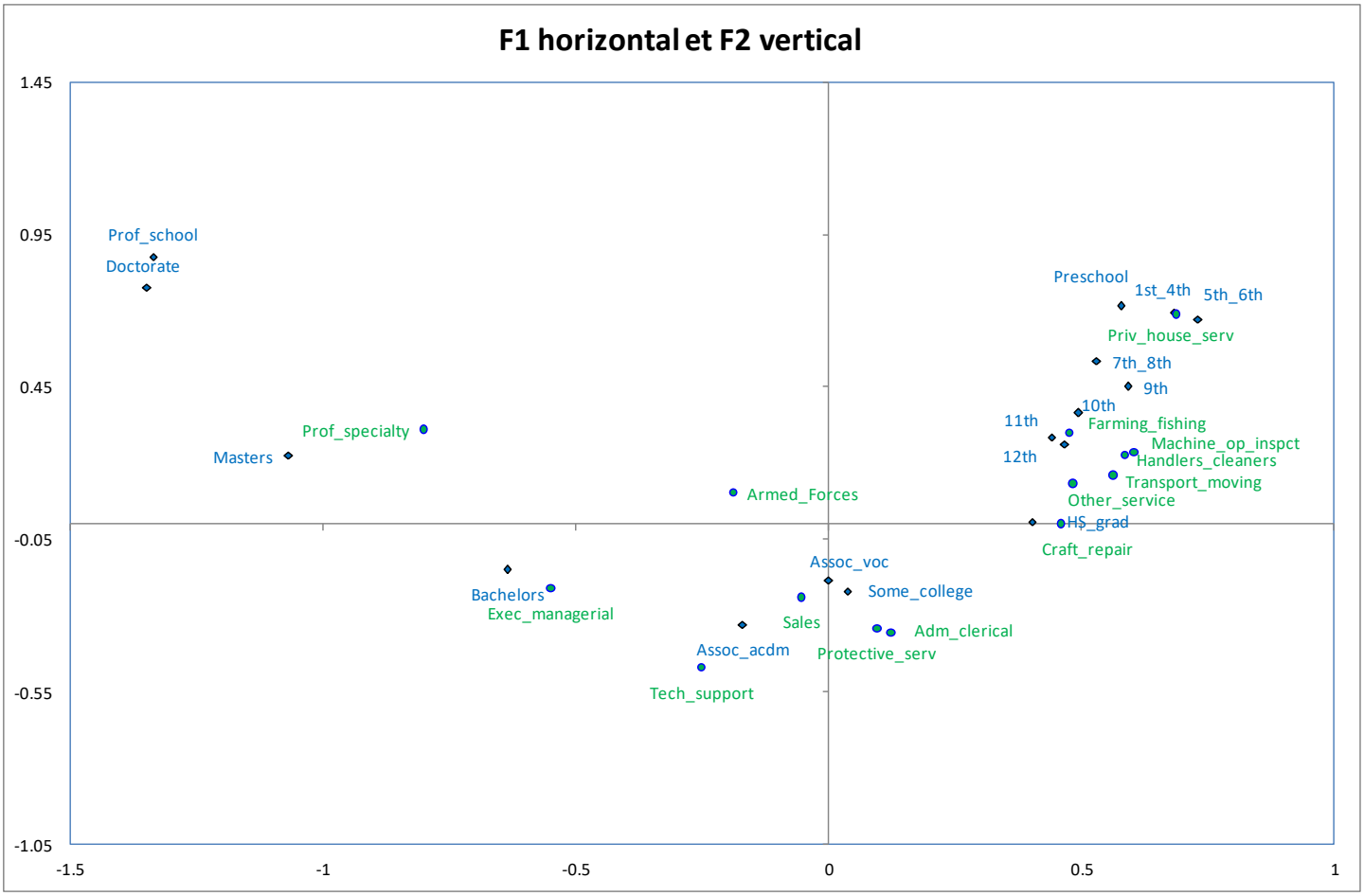


Figure 121 - Lien entre le niveau d'études et le type d'emploi occupé - TUTO 12

4.5 AFC avec d'autres outils (SAS, TANAGRA, R)

Nous explorons d'autres outils qui permettent de pratiquer l'analyse factorielle des correspondances dans cette section. D'une manière assez succincte toujours. L'objectif est de montrer dans les grandes lignes les fonctionnalités des logiciels.

4.5.1 Données « Qualificatifs Aliments »

Nous inaugurons un nouveau jeu de données décrit dans un tutoriel accessible en ligne (Bendixen, 1996). Le tableau de contingence – issu d'une enquête auprès de 100 ménages – croise 8 types d'aliments pour petit déjeuner avec 14 mots clés destinés à les qualifier. Nous avons un tableau avec **K** = **14 lignes** et **L** = **8 colonnes**. Les réponses multiples étant autorisées, l'effectif total est de **n** = **1760**. L'étude cherche à mettre en évidence d'une part les ressemblances et dissemblances entre les profils, d'autre part les relations (attractions ou répulsions) les plus marquantes entre les mots clés et les aliments.

		Foods							
		Cereals	Muesli	Porridge	BaconEggs	ToastTea	FreshFruit	StewedFruit	Yoghurt
Statement	Healthy	14	38	25	18	8	31	28	34
	Nutritious	14	28	25	25	7	32	26	31
	GoodSummer	42	22	11	13	7	37	16	35
	GoodWinter	10	10	32	26	6	11	19	8
	Expensive	6	33	5	27	3	9	18	10
	QuickEasy	54	33	8	2	15	26	8	20
	Tasty	24	21	16	34	11	33	26	26
	Economical	24	3	20	3	16	7	3	7
	ForATreat	5	3	3	31	4	4	16	17
	ForWeekdays	47	24	15	9	13	11	6	10
	ForWeekends	12	5	8	56	16	10	23	18
	Tasteless	8	6	2	2	0	0	2	1
	TooLongToPrepare	0	0	9	35	1	0	10	0
	FamilyFavourite	14	4	10	31	5	7	2	5

Figure 122 – Données "Qualificatifs aliments"

J'ai décrit en détail la mise en œuvre de l'AFC sur ces données dans un tutoriel qui avait pour objectif de comparer les fonctionnalités de plusieurs logiciels (TUTO 15). J'en étais arrivé au constat qu'ils produisaient des résultats numériques identiques. En revanche, de par le prisme adopté, ils éclairaient avec plus ou moins d'acuité différentes facettes de l'analyse.

4.5.2 AFC avec TANAGRA

Le composant « CORRESPONDENCE ANALYSIS » (onglet FACTORIAL ANALYSIS) implémente l'analyse factorielle des correspondances. L'importation des données et le mode de paramétrage de l'outil sont décrits dans notre précédent tutoriel (TUTO 15, sections 3.2 et 3.3). Dans ce support, nous demandons :

1. La production de 2 facteurs (nous comprendrons mieux pourquoi plus loin).

2. Le calcul des contributions au χ^2 des couples de modalités.
3. De trier les modalités selon les contributions dans les tableaux de résultats (coordonnées des modalités sur les facteurs notamment).

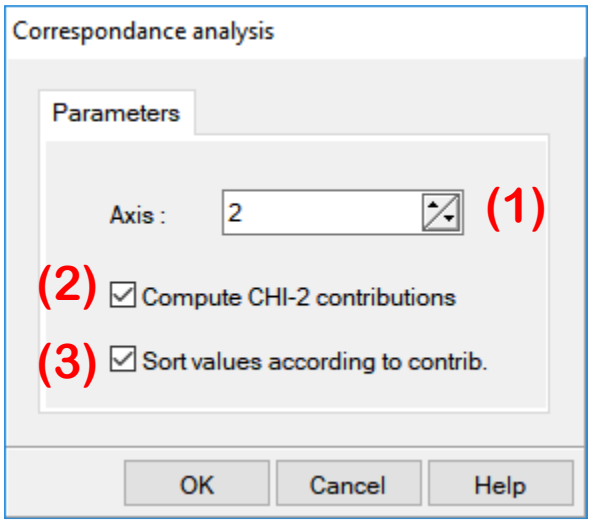


Figure 123 – Boîte de dialogue de paramétrage – AFC – Tanagra

Les sorties sont subdivisées en plusieurs zones, énumérerons-les tour à tour dans ce qui suit.

4.5.2.1 KHI-2 (global) de l'écart à l'indépendance

Le premier tableau indique la statistique du test du χ^2 d'écart à l'indépendance[3]. Ce résultat est fondamental. En effet, si la liaison globale est trop faible, l'étude des relations entre les modalités ne sert à rien. Il faut s'assurer qu'il existe une information exploitable dans le tableau.

CHI-SQUARE statistic

Trace	0.3678
Chi²	647.31
d.f	91
p-value	0.0000

Figure 124 –Test du χ^2 – AFC – Tanagra

[3] Le test du χ^2 n'est pas strictement applicable ici. En effet chaque individu a pu choisir plusieurs couples de valeurs (statement x food). De fait, les observations ne sont pas indépendantes. Il faut dès lors voir le χ^2 plutôt comme un indicateur de la quantité d'information exploitable dans le tableau.

En l'occurrence, nous avons χ^2_{global} = 647.31, avec un degré de liberté égal à 91 [(14 – 1) x (8 – 1)], la liaison est très significative (p-value < 0.0001). De plus, Tanagra fournit la valeur du coefficient ϕ^2 (**Trace**), avec $\phi^2 = \frac{\chi^2}{n} = \frac{647.31}{1760} = 0.3678$), il symbolise la quantité d'information disponible. En pratique, on peut penser que le tableau recèle des informations intéressantes à partir de ($\phi^2 > 0.2$) (Bendixen, 1996, page 7). L'AFC va décomposer cette quantité sur les différents axes factoriels.

4.5.2.2 Tableau des valeurs propres – Choix du nombre d'axes

Tableau des valeurs propres. Tanagra affiche ensuite le tableau des valeurs propres (λ_h). Elles expriment la part d'inertie expliquée par les axes. Ainsi, puisque la décomposition est orthogonale, elles s'additionnent et la somme (0.193095 + 0.077731 + … + 0.002363) est égale à ϕ^2 = 0.3678. Nous pouvons réécrire les valeurs en pourcentage d'inertie expliquée par les axes (ex. % axe 1 : 0.193095 / 0.3678 = 52.50 % ; % axe 2 : 0.07731 = 21.13 %).

Eigen values

Matrix trace = 0.3678
SQRT(Matrix trace) = 0.6065

Axis	Eigen value	% explained	Histogram	% cumulated
1	0.193095	52.50%		52.50%
2	0.077731	21.13%		73.64%
3	0.043854	11.92%		85.56%
4	0.032804	8.92%		94.48%
5	0.012257	3.33%		97.81%
6	0.005687	1.55%		99.36%
7	0.002363	0.64%		100.00%
Tot.	0.367791	-	-	-

Figure 125 – Tableau des valeurs propres – AFC – Tanagra

Choix du nombre d'axes – Scree plot. La règle du coude peut être mise à contribution pour la détection du nombre de facteurs pertinents. Dans le diagramme ci-dessus (Figure 125), on peut imaginer observer une inflexion au niveau de (Axis = 2).

Choix du nombre d'axes – Règle de Kaiser. L'autre approche consiste à utiliser la règle de Kaiser. Le nombre maximum d'axes factoriels que nous pouvons produire est H_{max} = MIN(L –1, C–1) =

MIN (13, 7) = 7. Dès lors, une règle très simple consiste à choisir les axes pour lesquels la part d'inertie restituée est supérieure à (1 / H_{max}) = 14.3 %, soit les axes 1 (52.50 %) et 2 (21.13 %).

Le même raisonnement peut porter sur les valeurs propres brutes. Nous retenons les axes factoriels portés par une valeur propre supérieure à leur moyenne c.-à-d. supérieure à (0.3678 / 7) = 0.0525 ; nous avons bien les axes 1 (λ_1=0.19309) et 2 (λ_2= 0.07773). TANAGRA met automatiquement en surbrillance (très légèrement parce que ce n'est pas une règle irréfragable) les valeurs concernées.

4.5.2.3 Représentation des modalités lignes

La représentation des lignes couvre plusieurs informations : les statistiques sur les points lignes (poids, distance à l'origine, l'inertie) [A] ; les coordonnées factorielles [B] ; les contributions (CTR) aux axes (en %) [C] ; et la qualité de représentation (COS^2) par axe et cumulée [D].

Rows analysis (A) (B) (C) (D)

Characterization				Coord.		Contributions (%)		COS^2			
Values	Weight	Sq. Dist.	Inertia	coord 1	coord 2	ctr 1	ctr 2	cos^2 1		cos^2 2	
TooLongToPrepare	0.03125	1.82554	0.05705	-1.27778	0.33401	26.42	4.49	0.89	(0.89)	0.06	(0.96)
QuickEasy	0.09432	0.46617	0.04397	0.64365	0.08476	20.24	0.87	0.89	(0.89)	0.02	(0.90)
ForWeekends	0.08409	0.43297	0.03641	-0.56005	0.17123	13.66	3.17	0.72	(0.72)	0.07	(0.79)
ForWeekdays	0.07670	0.42038	0.03225	0.50675	0.33799	10.20	11.27	0.61	(0.61)	0.27	(0.88)
ForATreat	0.04716	0.54240	0.02558	-0.61946	-0.06492	9.37	0.26	0.71	(0.71)	0.01	(0.72)
Economical	0.04716	0.80423	0.03793	0.41692	0.65687	4.25	26.18	0.22	(0.22)	0.54	(0.75)
Healthy	0.11136	0.16313	0.01817	0.08662	-0.34591	0.43	17.14	0.05	(0.05)	0.73	(0.78)
Expensive	0.06307	0.40466	0.02552	-0.20067	-0.36125	1.32	10.59	0.10	(0.10)	0.32	(0.42)
FamilyFavourite	0.04432	0.43902	0.01946	-0.38784	0.42183	3.45	10.15	0.34	(0.34)	0.41	(0.75)
Nutritious	0.10682	0.10686	0.01141	-0.00861	-0.26886	0.00	9.93	0.00	(0.00)	0.68	(0.68)
GoodSummer	0.10398	0.22035	0.02291	0.34879	-0.13275	6.55	2.36	0.55	(0.55)	0.08	(0.63)
Tasty	0.10852	0.03870	0.00420	-0.03963	-0.11273	0.09	1.77	0.04	(0.04)	0.33	(0.37)
GoodWinter	0.06932	0.33945	0.02353	-0.27741	0.12713	2.76	1.44	0.23	(0.23)	0.05	(0.27)
Tasteless	0.01193	0.78910	0.00942	0.45096	0.15778	1.26	0.38	0.26	(0.26)	0.03	(0.29)

Figure 126 – Représentation des lignes – AFC – Tanagra

TANAGRA intègre une fonctionnalité assez pratique : il peut trier en cascade les modalités lignes selon leur contributions aux facteurs, en mettant des codes couleurs pour identifier les coordonnées selon leur signe. Comme l'information est décomposée sur plusieurs facteurs, on ne peut pas se contenter d'effectuer un tri sur le premier uniquement. Tanagra s'appuie sur la stratégie suivante : pour le premier axe, il identifie les variables qui contribuent plus que la

moyenne non pondérée (1/K pour les lignes, 1/L pour les colonnes), il trie les modalités selon leurs contributions ; parmi les modalités restantes, il identifie celles qui pèsent sur le second axe, il les trie de nouveau, etc. Ainsi, nous avons une représentation en cascade (en diagonale) qui permet d'identifier très rapidement la nature des informations véhiculées par les facteurs.

TANAGRA met en surbrillance les coordonnées des modalités répondant aux conditions suivantes : (CTR > 1/K)[4] et (COS^2 > $1/H_{max}$) c.-à-d. la modalité contribue plus que la moyenne (non pondérée), et l'information qu'elle véhicule est concentrée sur le facteur (plus que la moyenne). L'idée est d'attirer l'œil de l'utilisateur sur les éléments saillants du tableau qui est, reconnaissons-le, assez rébarbatif à lire.

Ainsi, le 1er facteur est déterminé par l'opposition entre (TooLongToPrepare, ForWeekends, ForATreat) et (QuickEasy, ForWeekdays), avec l'idée de l'antagonisme entre la cuisine des grandes occasions vs. celle de tous les jours. Le 2nd par l'opposition (Healthy, Expensive, Nutritous) et (ForWeekdays, Economical, FamilyFavourite), la cuisine saine, onéreuse, serait incompatible avec la boustifaille de tous les jours, économique, pourtant favorite des familles (on mange des frites aux pâtes aujourd'hui les enfants, ça vous va ? ouaiiiis !!!).

On notera surtout que les modalités à inertie élevée vont souvent conditionner les résultats sur les premiers axes. Ce n'est pas un problème en soi. Il faut en avoir conscience simplement pour une lecture distanciée. Il est important que ces éléments (poids, distance à l'origine, inertie – la partie A du tableau) soient visibles en même que les coordonnées, CTR et COS2.

Enfin, les COS^2 indiquent la qualité de représentation individuelle et cumulée des modalités sur les H premiers facteurs. Dans notre exemple, les modalités « GoodWinter » et « TasteLess » sont les moins bien restituées par (F1, F2) (moins que les autres modalités tout du moins).

[4] Nous avons préféré cette règle à (CTR > poids de la modalité) (Saporta, 2006 ; page 207) pour éviter de mettre en surbrillance des modalités sous-représentées qui, de toute manière, contribuent faiblement au χ^2_{global}. Par exemple, avec cette seconde condition, nous aurions dû mettre « tasteless » en évidence sur le premier axe (CTR = 1.26%, poids = 1.19%). Or, cette modalité pèse finalement très peu dans l'analyse. Elle ne correspond ni à une attraction ni à une répulsion réellement marquée avec l'une des modalités colonnes.

4.5.2.4 Représentation des modalités colonnes

Les mêmes schémas que la représentation des lignes s'applique pour les modalités colonne.

Columns analysis

Characterization				Coord.		Contributions (%)		COS^2	
Values	Weight	Sq. Dist.	Inertia	coord 1	coord 2	ctr 1	ctr 2	cos^2 1	cos^2 2
BaconEggs	0.17727	0.66963	0.11871	-0.78344	0.14814	56.35	5.00	0.92 (0.92)	0.03 (0.95)
Cereals	0.15568	0.48968	0.07623	0.56272	0.35791	25.53	25.66	0.65 (0.65)	0.26 (0.91)
Muesli	0.13068	0.33926	0.04433	0.31310	-0.31869	6.63	17.08	0.29 (0.29)	0.30 (0.59)
ToastTea	0.06364	0.37744	0.02402	0.17534	0.44702	1.01	16.36	0.08 (0.08)	0.53 (0.61)
Yoghurt	0.12614	0.15878	0.02003	0.08599	-0.26416	0.48	11.32	0.05 (0.05)	0.44 (0.49)
FreshFruit	0.12386	0.19655	0.02434	0.24619	-0.24493	3.89	9.56	0.31 (0.31)	0.31 (0.61)
StewedFruit	0.11534	0.19120	0.02205	-0.31540	-0.24281	5.94	8.75	0.52 (0.52)	0.31 (0.83)
Porridge	0.10739	0.35450	0.03807	-0.05363	0.21310	0.16	6.27	0.01 (0.01)	0.13 (0.14)

Figure 127 – Représentation des modalités colonnes – AFC – Tanagra

Il faut connaître un peu les aliments pour se lancer dans des commentaires circonstanciés. Il me semblait par exemple que les « Muesli » étaient des céréales. Le fait que la modalité soit en opposition avec « Cereals » me laisse à penser qu'elle recouvre peut-être autre chose dans la culture des personnes enquêtées.

4.5.2.5 Représentation simultanée

Dans l'onglet CHART, TANAGRA propose la représentation simultanée par couple de facteurs (Figure 128). La popularité des analyses factorielles repose pour beaucoup sur les « cartes » graphiques. On a l'impression de tout comprendre en un coup d'œil. Il est toujours très stimulant de pouvoir associer visuellement des points. Certains outils proposent même la visualisation 3D. On passe un temps fou alors à tourner le graphique dans tous les sens, à voir le nuage par en dessous, sur le côté, de travers, etc., en perdant complètement le fil de l'analyse au passage.

Pour nos données, nous observons bien les proximités et oppositions entre les modalités homologues (les aliments entre eux d'une part, les qualificatifs entre eux d'autre part), en se cantonnant aux points à forts CTR et/ou COS2 toujours.

L'affaire est plus périlleuse lorsqu'il s'agit de rapprocher (ou opposer) les modalités lignes et colonnes pour identifier les attractions (ou les répulsions). Il faut toujours s'astreindre à une

vision relative des points (une modalité ligne est positionnée globalement par rapport à l'ensemble des modalités colonnes) en raison de la relation quasi-barycentrique.

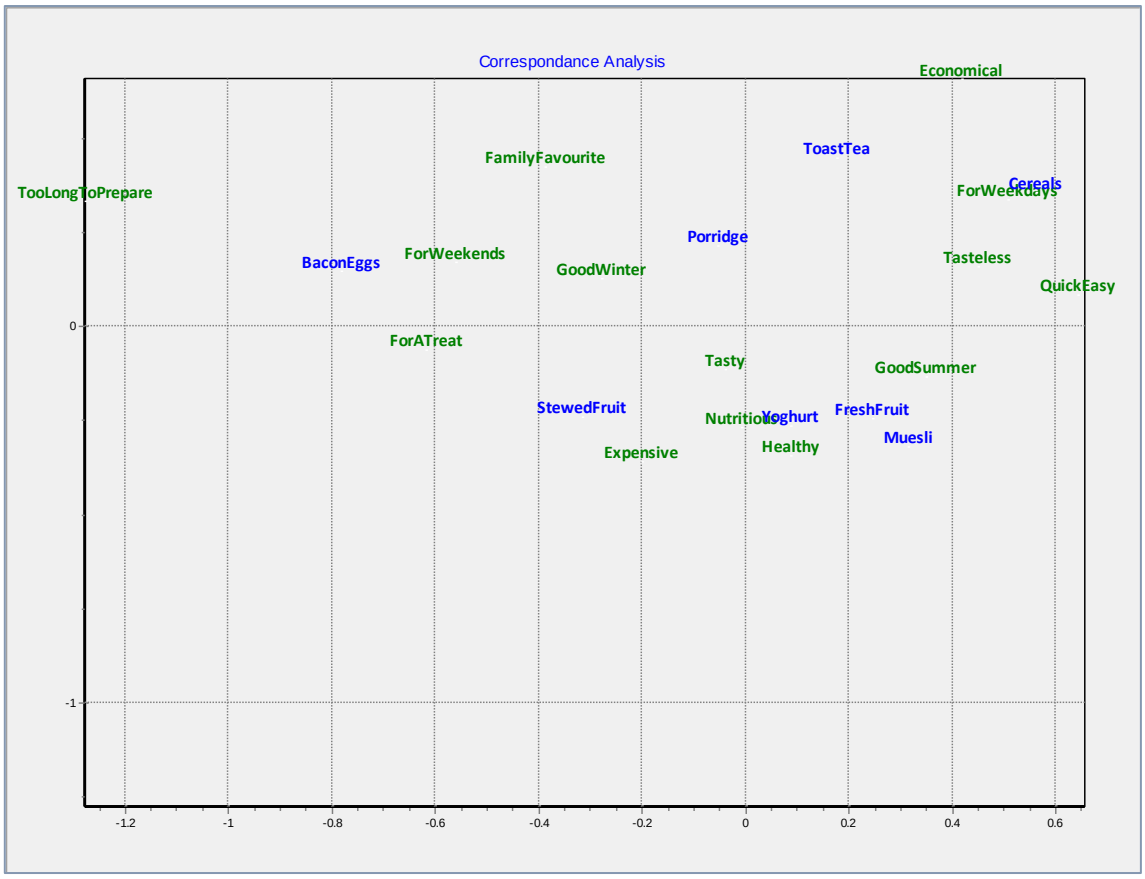


Figure 128 - Représentation simultanée - AFC - Tanagra

4.5.2.6 Contributions au KHI-2

Pour guider le praticien dans l'étude des attractions et répulsions entre les modalités lignes et colonnes, TANAGRA fournir le tableau des contributions au χ^2_{global}.

Pour chaque combinaison des modalités lignes / colonnes, TANAGRA propose : l'effectif observé, l'effectif théorique (sous indépendance), le résidu standardisé, la contribution absolue au χ^2 [(+) attraction ou (-) répulsion], et la contribution relative (en pourcentage). Il trie le tableau de manière décroissante pour que les informations les plus marquantes apparaissent au premier plan. Seules les contributions supérieures à la moyenne [contribution relative > 1 / (K x L)] sont affichées (Figure 129).

Mettre en relation ce tableau avec la représentation simultanée nous évite bien des déboires dans la lecture des résultats de l'analyse factorielle des correspondances. On se rend compte notamment que l'information portée par le tableau de contingence est en réalité « écrasée » par l'attraction entre « BaconEggs » et « TooLongToPrepare » (10.10%).

CHI-2 contributions

Id	Row	Column	Value	Expected	Std.Resid.	Contrib.		%
1	TooLongToPrepare	BaconEggs	35	9.8	8.09	(+)	65.39	10.10
2	ForWeekends	BaconEggs	56	26.2	5.81	(+)	33.77	5.22
3	ForWeekdays	Cereals	47	21.0	5.67	(+)	32.12	4.96
4	QuickEasy	Cereals	54	25.8	5.54	(+)	30.68	4.74
5	GoodWinter	Porridge	32	13.1	5.22	(+)	27.26	4.21
6	QuickEasy	BaconEggs	2	29.4	-5.06	(-)	25.56	3.95
7	Expensive	Muesli	33	14.5	4.86	(+)	23.58	3.64
8	Economical	ToastTea	16	5.3	4.66	(+)	21.75	3.36
9	FamilyFavourite	BaconEggs	31	13.8	4.62	(+)	21.33	3.29
10	ForATreat	BaconEggs	31	14.7	4.25	(+)	18.03	2.78
11	Economical	Porridge	20	8.9	3.71	(+)	13.79	2.13
12	GoodSummer	BaconEggs	13	32.4	-3.41	(-)	11.65	1.80
13	ForWeekends	Muesli	5	19.3	-3.26	(-)	10.63	1.64
14	Economical	Cereals	24	12.9	3.08	(+)	9.50	1.47
15	Economical	BaconEggs	3	14.7	-3.05	(-)	9.33	1.44
16	ForWeekdays	BaconEggs	9	23.9	-3.05	(-)	9.32	1.44
17	GoodSummer	FreshFruit	37	22.7	3.01	(+)	9.06	1.40
18	Healthy	Cereals	14	30.5	-2.99	(-)	8.94	1.38
19	TooLongToPrepare	Cereals	0	8.6	-2.93	(-)	8.56	1.32
20	Healthy	BaconEggs	18	34.7	-2.84	(-)	8.07	1.25
21	Nutritious	Cereals	14	29.3	-2.82	(-)	7.96	1.23
22	Expensive	Cereals	6	17.3	-2.71	(-)	7.36	1.14
23	TooLongToPrepare	Muesli	0	7.2	-2.68	(-)	7.19	1.11
24	TooLongToPrepare	Yoghurt	0	6.9	-2.63	(-)	6.94	1.07
25	Tasteless	Cereals	8	3.3	2.62	(+)	6.85	1.06
26	TooLongToPrepare	FreshFruit	0	6.8	-2.61	(-)	6.81	1.05
27	QuickEasy	StewedFruit	8	19.1	-2.55	(-)	6.49	1.00
28	GoodSummer	Cereals	42	28.5	2.53	(+)	6.41	0.99
29	GoodSummer	Yoghurt	35	23.1	2.48	(+)	6.15	0.95
30	Healthy	Muesli	38	25.6	2.45	(+)	5.99	0.93
31	QuickEasy	Muesli	33	21.7	2.43	(+)	5.89	0.91
32	ForWeekdays	StewedFruit	6	15.6	-2.43	(-)	5.88	0.91

Figure 129 – Contributions au KHI-2 – AFC – Tanagra

4.5.2.7 Traitement des lignes ou colonnes supplémentaires

Le principe des observations supplémentaires est applicable en AFC. Dans le tutoriel originel, l'auteur décrit un tableau croisant la fréquence de consommation avec le type d'aliment.

	Cereals	Muesli	Porridge	BaconEggs	ToastTea	FreshFruit	StewedFruit	Yoghurt
I	24	3	4	8	18	2	9	11
II	58	15	8	13	16	10	10	29
III	6	10	12	46	8	14	15	8
IV	2	4	28	9	4	47	4	2
V	10	68	48	24	54	27	62	50

I : Daily ; II : Several times per week ; III : Several times per month ; IV : Every few months ; V : Never.

A partir de la relation quasi-barycentrique, il est possible de les positionner sur les 2 premiers facteurs à l'aide des coordonnées des modalités colonnes (section 4.3.4). Mais l'idée n'est pas toujours très bien comprise, surtout que la plupart des outils proposent de le faire directement en interne, masquant le mécanisme de projection aux utilisateurs. Tanagra propose une approche simplifiée. Il fournit les coefficients de fonctions score c.-à-d. les fonctions permettant de projeter les points supplémentaires dans le repère factoriel à partir de leur profil. Ces coefficients peuvent être exportés facilement dans d'autres outils (ex. dans le tableur Excel), permettant de calculer facilement les coordonnées des points supplémentaires.

Voici les coefficients de la fonction score pour les lignes additionnelles (Figure 130).

Factor score coefficients for supplementary row

From column values (relative frequency)

Column value	Factor 1	Factor 2
Cereals	1.280579	1.283725
Muesli	0.712525	-1.143072
Porridge	-0.122041	0.764323
BaconEggs	-1.782883	0.531349
ToastTea	0.399026	1.603371
FreshFruit	0.560261	-0.878505
StewedFruit	-0.717755	-0.870916
Yoghurt	0.195685	-0.947476

Figure 130 – Coefficients des fonctions score – Lignes supplémentaires – AFC – Tanagra

Pour être applicable aux modalités supplémentaires, nous devons transformer le tableau des effectifs en profils.

	Cereals	Muesli	Porridge	BaconEggs	ToastTea	FreshFruit	StewedFruit	Yoghurt
I	0.304	0.038	0.051	0.101	0.228	0.025	0.114	0.139
II	0.365	0.094	0.050	0.082	0.101	0.063	0.063	0.182
III	0.050	0.084	0.101	0.387	0.067	0.118	0.126	0.067
IV	0.020	0.040	0.280	0.090	0.040	0.470	0.040	0.020
V	0.029	0.198	0.140	0.070	0.157	0.079	0.181	0.146

Figure 131 – Modalités supplémentaires – Profils lignes – AFC – Tanagra

Nous pouvons dès lors calculer leurs coordonnées pour les deux premiers axes.

	Scores	
	Factor.1	Factor.2
I	0.280	0.551
II	0.448	0.321
III	-0.562	0.082
IV	0.114	-0.161
V	0.042	-0.157

Figure 132 – Coordonnées factorielles des modalités lignes supplémentaires – AFC – Tanagra

Détaillons le calcul pour « III : Several times per month » sur le 1er facteur.

C(III, Factor 1) = 1.280579 x 0.050 + 0.712525 x 0.084 + (-0.122041) x 0.101 + (-1.782883) x 0.387 + 0.399026 x 0.067 + 0.560261 x 0.118 + (-0.717755) x 0.126 + 0.195685 x 0.067 = **-0.562**

Ce qui le situerait plutôt du côté de (TooLongToPrepare, ForATreat, ForWeekends). Cela semble logique dans la mesure où « BaconEggs » est l'aliment le plus prisé dans ce cas (38.7 % dixit le tableau des profils lignes, Figure 131).

Remarque : Le même dispositif existe pour les modalités colonnes supplémentaires. Il sera possible de positionner un nouveau produit alimentaire à partir de ses qualificatifs.

4.5.3 AFC avec SAS – PROC CORRESP

J'ai énormément étudié la PROC CORRESP de SAS pour développer l'AFC dans TANAGRA. La procédure s'applique aussi à l'analyse des correspondances multiples (chapitre 5). Elle regorge d'options multiples qu'il faut savoir cerner. Quoiqu'en en dise, elle constitue une référence incontournable qu'il faut absolument connaître.

Voici la commande utilisée sur les données « Qualificatifs Aliments » : (DIMENS = 2) nous fixons à 2 le nombre de facteurs à générer pour éviter d'être noyés sous les tableaux de chiffres ; (CELLCHI2) fait apparaître les contributions au χ^2, si importants pour identifier les associations lignes-colonnes, dans les sorties.

```
PROC CORRESP DATA=FOODS DIMENS=2 CELLCHI2;
      VAR Cereals -- Yoghurt;
      ID Statement;
RUN;
```

Voici les « Displayed Output » de la PROC CORRESP sur nos données.

Tableau des contributions au χ^2. SAS attaque directement par le tableau des contributions exprimés en valeur absolue c.-à-d. en effectuant la somme des contributions, nous aboutissons au χ^2_{global}. L'enjeu consiste à repérer rapidement les associations les plus significatives. Un formatage en ce sens aurait été appréciable. Pouvoir lire facilement le sens de la relation (attraction ou répulsion) aurait été salutaire également. J'ai surligné manuellement (fond vert) les 5 attractions/contributions les plus marquantes (Figure 133). Nous remarquons que les contributions s'additionnent en ligne et en colonne. Nous pouvons également identifier les modalités lignes et colonnes les plus importantes (fond bleu).

La procédure CORRESP

Contributions à la statistique du khi-2 totale

	Cereals	Muesli	Porridge	BaconEggs	ToastTea	FreshFruit	StewedFruit	Yoghurt	Somme
Healthy	8.937	5.990	0.742	8.070	1.604	1.862	1.287	3.481	31.973
Nutritious	7.965	0.479	1.147	2.081	2.059	3.261	0.859	2.239	20.089
GoodSummer	6.407	0.153	3.809	11.650	1.853	9.063	1.236	6.152	40.324
GoodWinter	4.258	2.215	27.262	0.884	0.401	1.119	1.726	3.548	41.413
Expensive	7.364	23.580	4.017	2.725	2.338	1.640	2.110	1.143	44.917
QuickEasy	30.678	5.893	5.416	25.563	1.863	1.439	6.489	0.042	77.383
Tasty	1.106	0.628	0.992	0.001	0.110	3.689	0.715	0.151	7.392
Economical	9.498	5.676	13.791	9.325	21.750	1.047	4.513	1.150	66.751
ForATreat	4.856	5.676	3.923	18.027	0.311	3.837	4.314	4.074	45.019
ForWeekdays	32.122	2.291	0.017	9.316	2.263	1.958	5.883	2.901	56.752
ForWeekends	5.291	10.634	3.920	33.765	4.600	3.787	2.060	0.024	64.079
Tasteless	6.845	3.862	0.029	0.797	1.336	2.601	0.074	1.026	16.571
TooLongToPrepare	8.563	7.188	1.621	65.391	1.786	6.813	2.107	6.938	100.405
FamilyFavourite	0.284	3.763	0.315	21.328	0.000	0.733	5.441	2.380	34.243
Somme	134.174	78.030	67.001	208.924	42.274	42.847	38.814	35.249	647.312

Figure 133 – Tableau des contributions au KHI-2 – AFC – SAS PROC CORRESP

Tableau des inerties. Il ressemble beaucoup au tableau des valeurs propres de TANAGRA (section 4.5.2.2) (plutôt l'inverse). Nous avons pour chaque facteur : la valeur singulière, son carré qui correspond à la valeur propre (inertie principale), le χ^2 (n x inertie), la proportion individuelle et cumulée.

Décomposition de l'inertie et du Khi-2

Valeur singulière	Inertie principale	Khi-2	Pourcentage	Pourcentage cumulé
0.43943	0.19309	339.846	52.50	52.50
0.27880	0.07773	136.806	21.13	73.64
0.20941	0.04385	77.183	11.92	85.56
0.18112	0.03280	57.735	8.92	94.48
0.11071	0.01226	21.572	3.33	97.81
0.07541	0.00569	10.010	1.55	99.36
0.04861	0.00236	4.159	0.64	100.00
	0.36779	647.312	100.00	

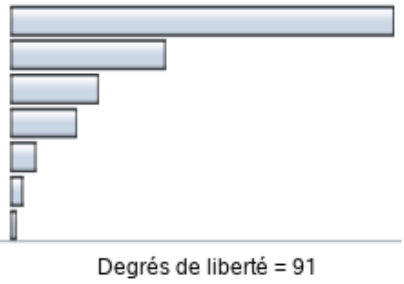


Figure 134 – Tableau des inerties – AFC – SAS PROC CORRESP

Coordonnées, CTR et COS2. Ils sont dispatchés dans plusieurs tableau. J'ai fait un petit montage pour que nous ayons une vision synthétique. Voici les informations pour les modalités lignes (Figure 135). Nous disposons des mêmes tableaux pour les modalités colonnes.

Coord.

Coordonnées des lignes		
	Dim1	Dim2
Healthy	0.0866	-0.3459
Nutritious	-0.0086	-0.2689
GoodSummer	0.3488	-0.1328
GoodWinter	-0.2774	0.1271
Expensive	-0.2007	-0.3612
QuickEasy	0.6437	0.0848
Tasty	-0.0396	-0.1127
Economical	0.4169	0.6569
ForATreat	-0.6195	-0.0649
ForWeekdays	0.5068	0.3380
ForWeekends	-0.5600	0.1712
Tasteless	0.4510	0.1578
TooLongToPrepare	-1.2778	0.3340
FamilyFavourite	-0.3878	0.4218

CTR

Contributions partielles à l'inertie des points des lignes		
	Dim1	Dim2
Healthy	0.0043	0.1714
Nutritious	0.0000	0.0993
GoodSummer	0.0655	0.0236
GoodWinter	0.0276	0.0144
Expensive	0.0132	0.1059
QuickEasy	0.2024	0.0087
Tasty	0.0009	0.0177
Economical	0.0425	0.2618
ForATreat	0.0937	0.0026
ForWeekdays	0.1020	0.1127
ForWeekends	0.1366	0.0317
Tasteless	0.0126	0.0038
TooLongToPrepare	0.2642	0.0449
FamilyFavourite	0.0345	0.1015

COS2

Carrés du cosinus pour les points des lignes		
	Dim1	Dim2
Healthy	0.0460	0.7335
Nutritious	0.0007	0.6765
GoodSummer	0.5521	0.0800
GoodWinter	0.2267	0.0476
Expensive	0.0995	0.3225
QuickEasy	0.8887	0.0154
Tasty	0.0406	0.3284
Economical	0.2161	0.5365
ForATreat	0.7075	0.0078
ForWeekdays	0.6109	0.2717
ForWeekends	0.7244	0.0677
Tasteless	0.2577	0.0315
TooLongToPrepare	0.8944	0.0611
FamilyFavourite	0.3426	0.4053

Figure 135 – Coordonnés, Contributions, COS2 – AFC – SAS PROC CORRESP

Statistiques des points lignes. Le tableau des statistiques regroupe le COS2 cumulé de chaque modalité sur le nombre de facteurs spécifié (DIMENS) (Qualité), son poids relatif ($\frac{n_k}{n}$) (Masse),

la part qui lui est imputable dans l'inertie globale [$\frac{Inertie(y_k)}{\chi^2_{global}}$] (Inertie). Un tableau équivalent est proposé pour les modalités colonnes.

Statistiques descriptives pour les points des lignes			
	Qualité	Masse	Inertie
Healthy	0.7795	0.1114	0.0494
Nutritious	0.6772	0.1068	0.0310
GoodSummer	0.6321	0.1040	0.0623
GoodWinter	0.2743	0.0693	0.0640
Expensive	0.4220	0.0631	0.0694
QuickEasy	0.9041	0.0943	0.1195
Tasty	0.3689	0.1085	0.0114
Economical	0.7527	0.0472	0.1031
ForATreat	0.7152	0.0472	0.0695
ForWeekdays	0.8826	0.0767	0.0877
ForWeekends	0.7921	0.0841	0.0990
Tasteless	0.2893	0.0119	0.0256
TooLongToPrepare	0.9555	0.0313	0.1551
FamilyFavourite	0.7480	0.0443	0.0529

Figure 136 – Statistiques pour les points lignes – SAS PROC CORRESP

Indices des coordonnées contributives. SAS propose un tableau des « indices des coordonnées qui contribuent le plus à l'inertie des points ». Il permet d'identifier les modalités présentant les plus fortes contributions sur chaque facteur. Je n'ai pas trouvé la valeur seuil dans la documentation. Pour les modalités lignes, nous avons (Figure 137) :

Indices des coordonnées qui contribuent le plus to à l'inertie des points de ligne			
	Dim1	Dim2	Meilleur
Healthy	0	2	2
Nutritious	0	2	2
GoodSummer	1	0	1
GoodWinter	0	0	1
Expensive	0	2	2
QuickEasy	1	0	1
Tasty	0	0	2
Economical	0	2	2
ForATreat	1	0	1
ForWeekdays	2	2	2
ForWeekends	1	0	1
Tasteless	0	0	1
TooLongToPrepare	1	0	1
FamilyFavourite	0	2	2

Figure 137 – Identification des modalités à fortes contributions sur les facteurs – SAS PROC CORRESP

Par rapport à TANAGRA (Figure 126), SAS a intégré – en plus – « Good Summer » (CTR = 6.55%) parmi les modalités « remarquables » pour le 1er facteur. On note que « ForWeekdays » est déterminant sur les deux facteurs.

Bien évidemment, un tableau avec la même finalité est proposé pour les modalités colonnes.

Carte des modalités – Représentation simultanée. Que seraient les méthodes factorielles sans les représentations graphiques ? SAS propose par défaut la carte des modalités lignes et colonnes fondus dans la même plan factoriel (nous n'avions demandé que DIMENS = 2 dimensions) (Figure 138).

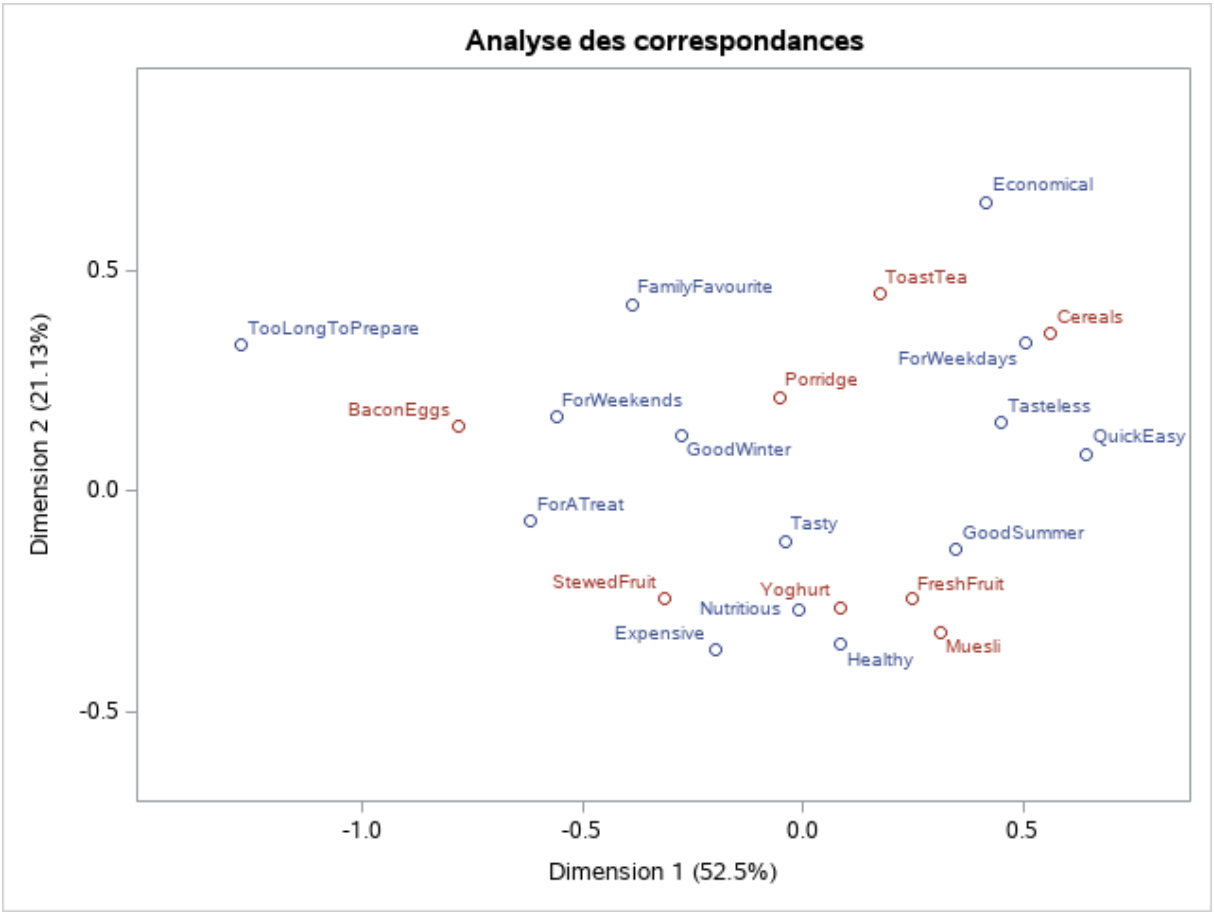


Figure 138 – Représentation simultanée – SAS PROC CORRESP

4.5.4 AFC avec R – Package « ade4 »

Plusieurs packages peuvent faire l'affaire sous R. Cette multiplicité est à la fois un atout et une faiblesse. Un atout parce que nous disposons de plusieurs points de vue pour un même traitement. Cela ne peut qu'enrichir l'analyse. Une faiblesse parce qu'obtenir des résultats

présentés différemment pour un même problème est toujours compliqué, surtout lorsqu'on a peu de recul par rapport aux techniques. Parfois, les sorties de certaines librairies ne correspondent pas aux présentations que l'on retrouve dans les ouvrages de référence. Cela peut rapidement semer la confusion dans l'esprit de l'utilisateur.

Heureusement, les packages sont de qualité pour l'analyse factorielle sous R. Nous avions déjà exploré « FactoMineR » (TUTO 14), « ca » et « ade4 » (TUTO 15). Nous reprenons l'étude avec la librairie « ade4 » en mettant l'accent sur les outils graphiques.

Nous chargeons les données – 7ème feuille du fichier « **Data_Methodes_Factorielles.xlsx** » – avec le package « openxlsx ». La première colonne représente les étiquettes de ligne.

```
#charger les données
library(openxlsx)
D <- read.xlsx("Data_Methodes_Factorielles.xlsx",sheet=7,rowNames=TRUE)
str(D)

## 'data.frame':    14 obs. of  8 variables:
##  $ Cereals    : num  14 14 42 10 6 54 24 24 5 47 ...
##  $ Muesli     : num  38 28 22 10 33 33 21 3 3 24 ...
##  $ Porridge   : num  25 25 11 32 5 8 16 20 3 15 ...
##  $ BaconEggs  : num  18 25 13 26 27 2 34 3 31 9 ...
##  $ ToastTea   : num  8 7 7 6 3 15 11 16 4 13 ...
##  $ FreshFruit : num  31 32 37 11 9 26 33 7 4 11 ...
##  $ StewedFruit: num  28 26 16 19 18 8 26 3 16 6 ...
##  $ Yoghurt    : num  34 31 35 8 10 20 26 7 17 10 ...
```

Nous importons la librairie « ade4 », après l'avoir installé préalablement si nécessaire. Nous faisons appel ensuite à la fonction dudi.coa() pour l'AFC. Nous lui passons les données (D), nous demandons la production de (nf = 2) facteurs, il n'est pas nécessaire de demander interactivement le nombre de facteurs (scannf = FALSE).

```
#librairie ade4
library(ade4)

#afc
afc <- dudi.coa(D,nf=2,scannf=FALSE)
summary(afc)

## Class: coa dudi
## Call: dudi.coa(df = D, scannf = FALSE, nf = 2)
##
## Total inertia: 0.3678
##
## Eigenvalues:
##     Ax1     Ax2     Ax3     Ax4     Ax5
## 0.19309 0.07773 0.04385 0.03280 0.01226
```

```
##
## Projected inertia (%):
##     Ax1     Ax2     Ax3     Ax4     Ax5
##  52.501  21.135  11.924   8.919   3.333
##
## Cumulative projected inertia (%):
##     Ax1   Ax1:2   Ax1:3   Ax1:4   Ax1:5
##   52.50   73.64   85.56   94.48   97.81
##
## (Only 5 dimensions (out of 7) are shown)
```

La fonction summary() affiche l'inertie totale et la qualité de restitution des axes factoriels en valeur propre, en pourcentage d'information restituée, et en pourcentage cumulé. Nous affichons l'éboulis des valeurs propres. Le « coude » est assez marqué pour (Axis = 2).

```
#screeplot pour le diagramme des val.propres
screeplot(afc,main="Eboulis des val. propres",type="lines")
```

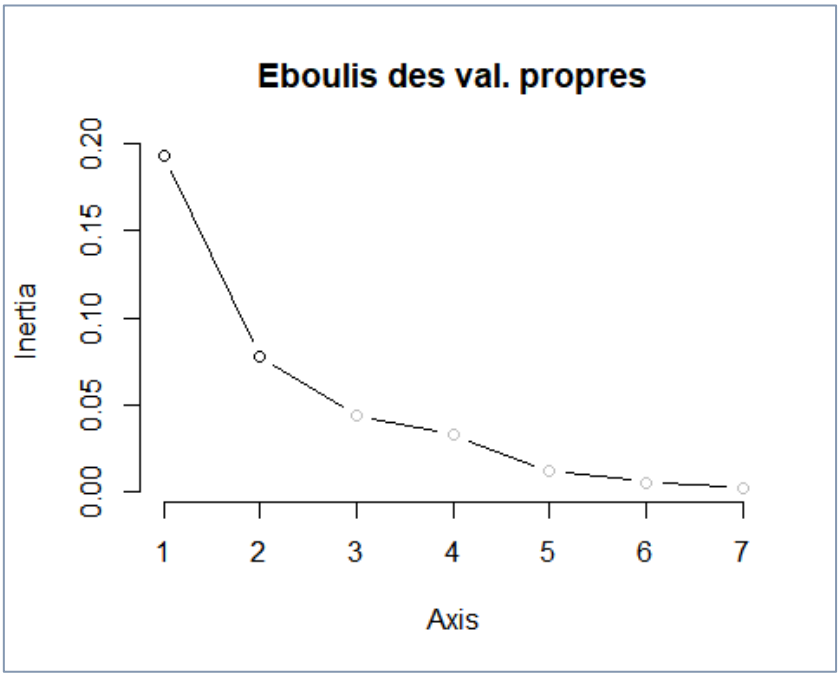


Figure 139 – Eboulis des valeurs propres – AFC – R, package "ade4"

Nous affichons les positions des modalités lignes et colonnes sur un facteur. Ce type de graphique remplace d'une certaine manière les coordonnées brutes empilées dans les tableaux, avec l'avantage certain d'une présentation triée, limité au facteur étudié, facilitant les comparaisons des modalités intra et inter-variables. Voici la représentation pour le 1er facteur (F1). Nous ne disposons pas en revanche, dans le même temps, des contributions et des COS2 pour situer l'influence ou la qualité de représentation des modalités.

```
#position des modalités sur l'axe numéro "xax"
score(afc,xax=1,clab.r=0.7,clab.c=0.75)
```

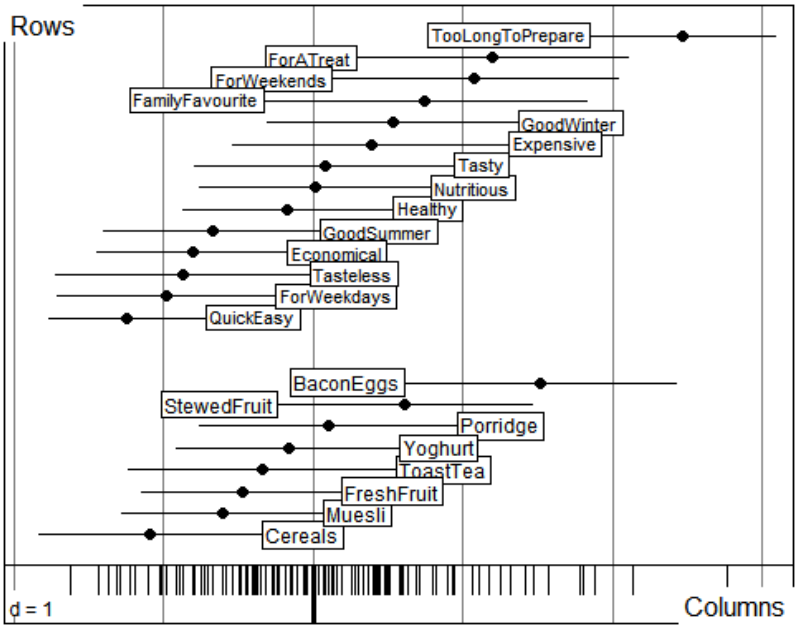


Figure 140 – Variante de la représentation simultanée – AFC – R, package "ade4"

Enfin, nous disposons de la représentation simultanée dans le premier plan factoriel. Nous spécifions l'option (method = 1) pour que les dispersions des points modalités lignes et colonnes soient identiques sur les facteurs.

```
#représentation simultanée
scatter(afc,method=1)
```

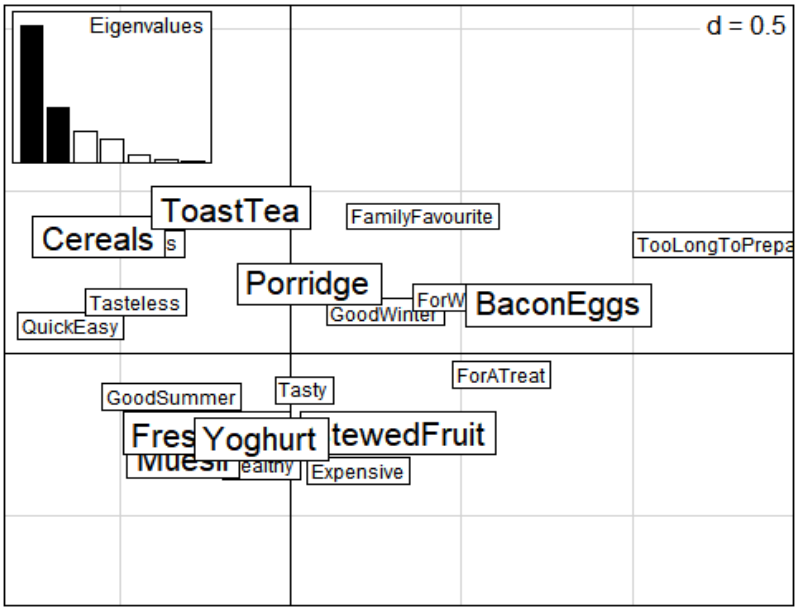


Figure 141 – Représentation simultanée – AFC – R, package "ade4"

5 Analyse des correspondances multiples (ACM)

L'analyse factorielle des correspondances multiples (AFCM) ou, plus simplement, l'analyse des correspondances multiples (ACM), est le pendant de l'analyse en composantes principales lorsque les variables actives sont catégorielles. Les deux approches se rejoignent d'ailleurs lorsque les variables sont exclusivement binaires. Elle répond donc à la même problématique, elle cherche à projeter les individus dans un espace de dimension réduit, en respectant au mieux les proximités entre eux, mais en se plaçant dans un cadre théorique différent : elle substitue la distance du χ^2 à la distance euclidienne pour mesurer les proximités entre les individus. Bien sûr, au-delà du traitement des observations, comme pour toute méthode factorielle, la description synthétique des données met également en évidence les association entre les descripteurs et, dans le cas de l'analyse des correspondances multiples, leurs modalités.

L'ACM peut être aussi vue comme une technique de transformation de variables. Elle permet de passer d'un espace discret, décrit par les variables originales, en un espace continu, décrit par les axes factoriels. Sans perte d'informations si l'on choisit de conserver tous les axes. Avec une perte d'information contrôlée lorsque l'on ne conserve que les premiers axes. En effet, les derniers facteurs traduisent essentiellement les fluctuations aléatoires dues à l'échantillonnage, on lisse ainsi les données en nous concentrant sur les informations essentielles. Elle rend ainsi possible d'utilisation subséquente d'algorithmes de machine learning qui ne savent appréhender que les descripteurs quantitatifs. On citera, entres autres, l'exemple bien connu de la méthode DISQUAL (Saporta, 2006) en analyse prédictive. Elle consiste à projeter les observations dans un espace continu avant d'appliquer une analyse discriminante linéaire pour expliquer et prédire

les valeurs d'une variable cible nominale (Rakotomalala R., « Pratique de l'Analyse Discriminante Linéaire », version 1.0, mai 2020).

Pour le scientifique enfin, l'ACM est séduisante parce que, peut-être plus que les autres méthodes factorielles, elle se prête à une multitude de points de vue. Il est possible de la mettre en œuvre avec un programme d'analyse factorielle des correspondances simples ou d'analyse en composantes principales. Le tout est de préparer les données à bon escient, préparations qui constituent autant de prismes sur le même tableau initial.

5.1 Principe de l'analyse des correspondances multiples

5.1.1 Tableau de données

Données brutes. L'analyse des correspondances multiples traite les tableaux individus-variables, lesquelles sont toutes qualitatives. Nous reprenons une version réduite des données « Canidés » de Tenenhaus (2007, page 254) comprenant (n = 8) observations et (p = 3) variables.

Variables « actives » qualitatives
c.-à-d. sont utilisées pour la
construction des facteurs

$j : 1, \ldots, p$

$i : 1, \ldots, n$
Individus actifs

ID	Chien	Taille	Velocite	Affection
1	Beauceron	Taille++	Veloc++	Affec+
2	Basset	Taille-	Veloc-	Affec-
3	Berger All	Taille++	Veloc++	Affec+
4	Boxer	Taille+	Veloc+	Affec+
5	Bull-Dog	Taille-	Veloc-	Affec+
6	Bull-Mastif	Taille++	Veloc-	Affec-
7	Caniche	Taille-	Veloc+	Affec+
8	Labrador	Taille+	Veloc+	Affec+

Figure 142 – Données "Canidés"

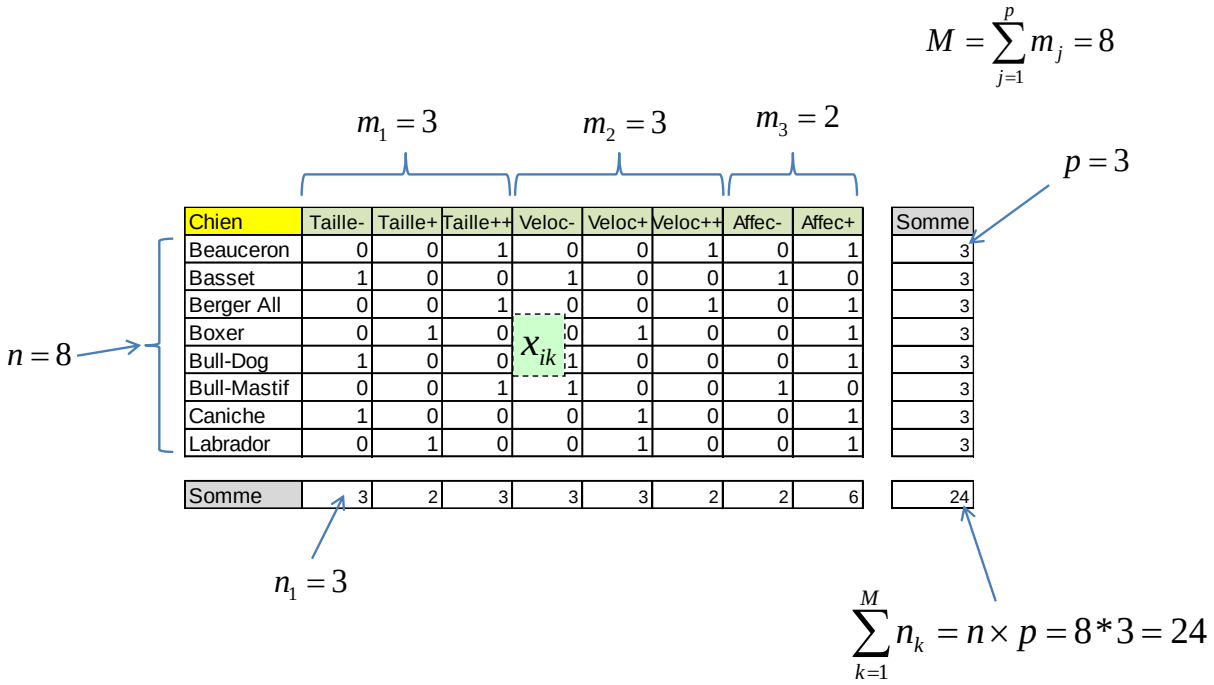
On se pose les questions usuelles de l'analyse factorielle :

- Quels sont les chiens qui se ressemblent ? Qui sont dissemblables ? (proximité entre les individus)
- Sur quelles caractères sont fondées ces ressemblances / dissemblances ?

- Quelles sont les associations entre les modalités ? Ex. Un animal de grande taille est-il plus affectueux ou moins affectueux ?
- Quelles sont les relations entre les variables ? Ex. Y-at-il une relation entre la taille et l'affection ou bien sont-ce des caractères orthogonaux ?

Il y a différentes manières d'appréhender l'analyse de ce tableau de données.

Codage disjonctif complet. Dans ce qui suit, nous travaillerons sur une version modifiée des données où les variables sont transformées en indicatrices o/1 via un codage disjonctif complet. Nous regroupons dans le graphique suivant les éléments de notation (Figure 143).



Chien	Taille-	Taille+	Taille++	Veloc-	Veloc+	Veloc++	Affec-	Affec+	Somme
Beauceron	0	0	1	0	0	1	0	1	3
Basset	1	0	0	1	0	0	1	0	3
Berger All	0	0	1	0	0	1	0	1	3
Boxer	0	1	0	0	1	0	0	1	3
Bull-Dog	1	0	0	1	0	0	0	1	3
Bull-Mastif	0	0	1	1	0	0	1	0	3
Caniche	1	0	0	0	1	0	0	1	3
Labrador	0	1	0	0	1	0	0	1	3
Somme	3	2	3	3	3	2	2	6	24

Figure 143 - Codage disjonctif complet - Notations

Le tableau binaire comporte toujours (n = 8) lignes, mais avec (M = 8) colonnes. M étant la somme du nombre des modalités des variables. Les sommes en ligne et colonne sont importantes parce qu'elles vont définir les profils moyens. L'effectif total est modifié parce que nous avons introduit de la redondance dans les données, il est égal au produit $(n \times p)$.

Sous Python, nous importons les données et nous procédons au codage. L'opération est très simple avec la librairie « Pandas ».

```
#chargement des données - index_col = 0 pour indiquer que la colonne n°0 est un label
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="ACM_CANINES",index_col=0)

#affichage des caractéristiques
print(D.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8 entries, Beauceron to Labrador
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Taille     8 non-null      object
 1   Velocite   8 non-null      object
 2   Affection  8 non-null      object
 3   Cote       8 non-null      float64
 4   Fonction   8 non-null      object
dtypes: float64(1), object(4)
memory usage: 384.0+ bytes
```

```
#récupération des variables actives
DActives = D[['Taille','Velocite','Affection']]
print(DActives)
```

```
              Taille Velocite Affection
Chien
Beauceron    Taille++  Veloc++    Affec+
Basset         Taill-    Velo-     Affe-
Berger All   Taille++  Veloc++    Affec+
Boxer         Taille+   Veloc+    Affec+
Bull-Dog       Taill-    Velo-    Affec+
Bull-Mastif  Taille++    Velo-     Affe-
Caniche        Taill-   Veloc+    Affec+
Labrador      Taille+   Veloc+    Affec+
```

```
#récupération des infos - nombre de variables
p = DActives.shape[1]

#nombre d'observations
n = DActives.shape[0]

#codage en 0/1 - les noms de modalités sont explicites
#pas nécessaire de préfixer les indicatrices par les noms de variables
X = pandas.get_dummies(DActives,prefix='',prefix_sep='')
print(X)
```

```
             Taill-  Taille+  Taille++  Velo-  Veloc+  Veloc++  Affe-  Affec+
Chien
Beauceron         0        0         1      0       0        1      0       1
Basset            1        0         0      1       0        0      1       0
Berger All        0        0         1      0       0        1      0       1
Boxer             0        1         0      0       1        0      0       1
Bull-Dog          1        0         0      1       0        0      0       1
Bull-Mastif       0        0         1      1       0        0      1       0
Caniche           1        0         0      0       1        0      0       1
Labrador          0        1         0      0       1        0      0       1
```

5.1.2 Analyse des proximités entre les individus

Le premier prisme d'analyse consiste à étudier les proximités entre les individus exprimés par leurs profils lignes (Figure 144). L'individu « moyen » est caractérisé par le profil moyen. Sa description est formée par les effectifs des modalités rapportés à l'effectif total.

$\frac{x_{ik}}{p}$

Chien	Taille-	Taille+	Taille++	Veloc-	Veloc+	Veloc++	Affec-	Affec+
Beauceron	0.000	0.000	0.333	0.000	0.000	0.333	0.000	0.333
Basset	0.333	0.000	0.000	0.333	0.000	0.000	0.333	0.000
Berger All	0.000	0.000	0.333	0.000	0.000	0.333	0.000	0.333
Boxer	0.000	0.333	0.000	0.000	0.333	0.000	0.000	0.333
Bull-Dog	0.333	0.000	0.000	0.333	0.000	0.000	0.000	0.333
Bull-Mastif	0.000	0.000	0.333	0.333	0.000	0.000	0.333	0.000
Caniche	0.333	0.000	0.000	0.000	0.333	0.000	0.000	0.333
Labrador	0.000	0.333	0.000	0.000	0.333	0.000	0.000	0.333
Profil moyen	0.125	0.083	0.125	0.125	0.125	0.083	0.083	0.250

Barycentre (O)

$\frac{n_k}{n \times p}$

Figure 144 – Tableau des profils lignes et profil moyen

5.1.2.1 Distance entre individus

Puisque nous traitons des variables catégorielles, nous utilisons la distance du χ^2 qui a pour particularité d'exacerber les différences entre les modalités rares.

$$d^2(i,i') = \sum_{k=1}^{M} \frac{1}{\frac{n_k}{n \times p}} \left(\frac{x_{ik}}{p} - \frac{x_{i'k}}{p} \right)^2$$

Entre « Beauceron » et « Basset » par exemple,

$$d^2(beauceron, basset) = \frac{1}{0.125}(0.000 - 0.333)^2 + \cdots + \frac{1}{0.250}(0.333 - 0.333)^2 = 5.778$$

Entre « Basset » et « Caniche »,

$$d^2(basset, caniche) = \frac{1}{0.125}(0.333 - 0.333)^2 + \cdots + \frac{1}{0.250}(0.000 - 0.333)^2 = 3.556$$

Visiblement, le Basset a plus de caractères en commun avec le Caniche qu'avec le Beauceron.

Sous Python, nous produisons le profil moyen qui fait également office de pondération dans l'expression de la distance du χ^2. Nous pouvons ensuite procéder aux calculs des distances entre les individus précités.

```
#librairie numpy
import numpy

#profil individu moyen
ind_moy = numpy.sum(X.values,axis=0)/(n*p)
print(ind_moy)

[0.125      0.08333333 0.125      0.125      0.125      0.08333333
 0.08333333 0.25      ]

#distance du KHI-2 entre beauceron (n°0) et basset (n°1)
print(numpy.sum(1/ind_moy*(X.values[0,:]/p-X.values[1,:]/p)**2))

5.777777777777777

#idem entre basset(n°1) et caniche(n°6)
print(numpy.sum(1/ind_moy*(X.values[1,:]/p-X.values[6,:]/p)**2))

3.5555555555555554
```

5.1.2.2 Distance à l'origine

La distance à l'origine correspond à la distance des individus au profil moyen représenté par le vecteur (0.125, 0.083, …, 0.250).

$$d^2(basset) = \frac{1}{125}(0.333 - 0.125)^2 + \cdots + \frac{1}{0.250}(0.000 - 0.250)^2 = 2.111$$

$$d^2(caniche) = \frac{1}{125}(0.333 - 0.125)^2 + \cdots + \frac{1}{0.250}(0.333 - 0.250)^2 = 1.222$$

Nous concluons que le Caniche est plus proche du « chien moyen » que le Basset.

```
#distance à l'origine du basset (n°1)
print(numpy.sum(1/ind_moy*(X.values[1,:]/p-ind_moy)**2))

2.1111111111111107

#distance à l'origine du caniche (n°6)
print(numpy.sum(1/ind_moy*(X.values[6,:]/p-ind_moy)**2))

1.2222222222222219
```

5.1.2.3 Inertie totale

L'inertie totale exprime la quantité d'information portée par les données. A l'instar de l'ACP (section 1.1.2.3), il est possible de la définir de deux manières : à travers les distances entre paires d'individus...

$$I = \frac{1}{2n^2}\sum_{i=1}^{n}\sum_{i' \neq i} d^2(i,i')$$

... ou via les distances à l'origine.

$$I = \frac{1}{n}\sum_{i=1}^{n} d^2(i)$$

Dans ce qui suit, nous calculons successivement pour les individus des données « Canidés » leurs poids ($\frac{1}{n}$), leurs distances à l'origine, leurs inerties [$\frac{1}{n}d^2(i)$], que nous sommons enfin.

```
#distance à l'origine des individus (obs.)
disto_ind = numpy.apply_along_axis(arr=X.values,axis=1,func1d=lambda x:numpy.sum(1
/ind_moy*(x/p-ind_moy)**2))

#poids des obs.
poids_ind = numpy.ones(X.shape[0])/n

#inertie
inertie_ind = poids_ind*disto_ind

#afffichage
print(pandas.DataFrame(numpy.transpose([poids_ind,disto_ind,inertie_ind]),index=D.index))
```

```
                0         1         2
Chien
Beauceron    0.125  1.666667  0.208333
Basset       0.125  2.111111  0.263889
Berger All   0.125  1.666667  0.208333
Boxer        0.125  1.666667  0.208333
Bull-Dog     0.125  1.222222  0.152778
Bull-Mastif  0.125  2.111111  0.263889
Caniche      0.125  1.222222  0.152778
Labrador     0.125  1.666667  0.208333
```

```
#inerrie totale
inertie_tot_ind = numpy.sum(inertie_ind)
print(inertie_tot_ind)
```

```
1.6666666666666665
```

L'information totale portée par les données, comptabilisée via les distances entre individus, est égale à ($I = 1.6667$). Nous verrons plus bas qu'il y a d'autres manières de la valoriser, et que nous aboutissons, quel que soit le point de vue, sur la même valeur numérique.

5.1.2.4 Principe de l'analyse des correspondances multiples – 1

L'objectif de l'ACM est de décomposer cette information sur une succession d'axes factoriels orthogonaux. Le nouveau système de représentation est conçu pour préserver au mieux les distances entre individus. Le nombre maximal de facteurs est égal à :

$$H_{max} = M - p$$

Lorsque nous les utilisons tous, les distances euclidiennes entre individus dans le repère factoriel sont identiques aux distances du χ^2 dans le repère original.

Pour le premier facteur, l'idée est de discerner le mieux possible les individus entre eux lorsqu'ils y sont projetés c.-à-d. concrètement, à maximiser les carrés des écarts à l'origine.

$$\lambda_1 = \sum_{i=1}^{n} \frac{1}{n} F_{i1}^2$$

Où :

- F_{ih} est la coordonnée de l'individu n°i sur le facteur n°h (sur le facteur [h=1] ci-dessus).
- Les facteurs sont centrés, la somme des coordonnés sur l'ensemble des individus est nulle.
- λ_h est la dispersion (inertie) associée au facteur n°h, on parle aussi de « valeur propre ».

Le second facteur est élaboré à partir de la fraction inexpliquée – la partie résiduelle – par le 1er facteur, … etc. Toute l'information est absorbée lorsque nous considérons les (H_{max}) facteurs. La part d'inertie restituée par le 1er facteur est ($\frac{\lambda_1}{I}$). Nous avons bien une décomposition orthogonale dans le sens où les variances portées par les facteurs s'additionnent :

$$\sum_{h=1}^{H_{max}} \lambda_h = I$$

Pour les données « Canidés », nous utilisons la classe de calcul MCA du package « fanalysis » sans en expliciter les propriétés à ce stade. Nous y reviendrons plus loin (section 5.3). Après avoir construit le repère factoriel, nous vérifions que la variance du premier facteur est bien égale à la première valeur propre.

```
#analyse avec fanalysis
from fanalysis.mca import MCA

#instanciation
acm = MCA(row_labels=DActives.index,var_labels=DActives.columns)
acm.fit(DActives.values)

#valeurs propres
print(acm.eig_[0])
```

```
[0.70803126 0.59148936 0.26199182 0.06974652 0.03540771]
```

Nous nous intéressons plus particulièrement à la première valeur propre.

Voici les coordonnées des individus sur le 1er facteur. Nous calculons explicitement la variance via la somme des carrés des valeurs puisque l'on sait qu'il est centré.

```
#coordonnées sur le 1er facteur
print(pandas.DataFrame(acm.row_coord_[:,0],index=DActives.index))
```

```
                        0
Chien
Beauceron    4.924471e-15
Basset       1.150779e+00
Berger All   4.763557e-15
Boxer       -1.150779e+00
Bull-Dog     4.284137e-01
Bull-Mastif  1.150779e+00
Caniche     -4.284137e-01
Labrador    -1.150779e+00
```

```
#inertie (variance) du 1er facteur
lambda_1_ind = numpy.mean(acm.row_coord_[:,0]**2)
print(lambda_1_ind)
```

```
0.7080312581410022
```

La part d'inertie correspond à la valeur propre divisée par l'inertie totale obtenue plus haut (section 5.1.2.3).

```
#part d'inertie
print(lambda_1_ind/inertie_tot_ind)
```

```
0.42481875488460136
```

5.1.3 Analyse des associations entre les modalités

L'analyse des associations entre les modalités revient à travailler sur les profils colonnes. Le tableau de données est normalisé par les sommes en colonne. Le profil moyen est égal au poids des observations (Figure 145).

$\frac{x_{ik}}{n_k}$ Barycentre : $\frac{p}{n \times p} = \frac{1}{n}$

Chien	Taille-	Taille+	Taille++	Veloc-	Veloc+	Veloc++	Affec-	Affec+	Profil moyen
Beauceron	0.000	0.000	0.333	0.000	0.000	0.500	0.000	0.167	0.125
Basset	0.333	0.000	0.000	0.333	0.000	0.000	0.500	0.000	0.125
Berger All	0.000	0.000	0.333	0.000	0.000	0.500	0.000	0.167	0.125
Boxer	0.000	0.500	0.000	0.000	0.333	0.000	0.000	0.167	0.125
Bull-Dog	0.333	0.000	0.000	0.333	0.000	0.000	0.000	0.167	0.125
Bull-Mastif	0.000	0.000	0.333	0.333	0.000	0.000	0.500	0.000	0.125
Caniche	0.333	0.000	0.000	0.000	0.333	0.000	0.000	0.167	0.125
Labrador	0.000	0.500	0.000	0.000	0.333	0.000	0.000	0.167	0.125

Figure 145 – Tableau des profils colonne – ACM – Données "Canidés"

5.1.3.1 Distance entre modalités

La distance entre modalités est définie par la distance du χ^2 toujours :

$$d^2(k,k') = \sum_{i=1}^{n} \frac{1}{\frac{1}{n}} \left(\frac{x_{ik}}{n_k} - \frac{x_{ik'}}{n_{k'}} \right)^2$$

Entre « taille- » et « vélocité- » par exemple,

$$d^2(taille-, velocite\ -) = \frac{1}{0.125}(0.000 - 0.000)^2 + \frac{1}{0.125}(0.000 - 0.000)^2 = 1.778$$

Entre « taille- » et « vélocité+ »,

$$d^2(taille-, velocite\ +) = \frac{1}{0.125}(0.000 - 0.000)^2 + \frac{1}{0.125}(0.000 - 0.333)^2 = 3.556$$

Conclusion : les individus qui partagent les caractéristiques (taille-, vélocité-) sont plus nombreux que (taille-, vélocité+).

Sous Python, nous calculons les sommes en colonne (n_k), il nous est alors facile de calculer les deux distances ci-dessus.

```
#somme en colonne
somme_col = numpy.sum(X.values,axis=0)
print(somme_col)

[3 2 3 3 3 2 2 6]

#distance entre taille- (2) et velocite- (5)
print(numpy.sum(n*((X.values[:,2]/somme_col[2]-X.values[:,5]/somme_col[5])**2)))

1.3333333333333335

#distance entre taille- (2) et velocite+ (3)
print(numpy.sum(n*((X.values[:,2]/somme_col[2]-X.values[:,3]/somme_col[3])**2)))

3.5555555555555554
```

5.1.3.2 Distance à l'origine

La distance à l'origine est définie par la distance du χ^2 au profil moyen :

$$d^2(k) = \sum_{i=1}^{n} \frac{1}{\frac{1}{n}} \left(\frac{x_{ik}}{n_k} - \frac{1}{n}\right)^2$$

Pour « taille- »,

$$d^2(taille-) = \frac{1}{0.125}(0.000 - 0.125)^2 + \cdots + \frac{1}{0.125}(0.000 - 0.125)^2 = 1.667$$

Pour « taille+ »,

$$d^2(taille-) = \frac{1}{0.125}(0.000 - 0.125)^2 + \cdots + \frac{1}{0.125}(0.500 - 0.125)^2 = 3.000$$

Conclusion : « Taille+ » est une caractéristique que l'on retrouve plus rarement que « Taille- » pour les animaux de notre base de données.

Sous Python, le calcul du profil moyen tel qu'il est défini juste ci-dessus ne pose aucune difficulté. Les distances à l'origine sont obtenues en appliquant directement la formule.

```
#profil moyen des variables-modalités
moda_moy = numpy.ones(X.shape[0])/n
```

```
#distance à l'origine taille-
print(numpy.sum(n*((X.values[:,0]/somme_col[0]-moda_moy)**2)))

1.6666666666666665

#distance à l'origine taille+
print(numpy.sum(n*((X.values[:,1]/somme_col[1]-moda_moy)**2)))

3.0
```

5.1.3.3 Inertie totale

L'inertie d'une modalité est exprimée par le produit entre son poids relatif ($\omega_k = \frac{n_k}{n \times p}$) et sa distance à l'origine :

$$I(k) = \omega_k \times d^2(k)$$

$$I(k) = \frac{n_k}{n \times p} \times d^2(k)$$

Et l'inertie totale, l'information portée par les données via le prisme des modalités, correspond à la somme de leur inertie :

$$I = \sum_{k=1}^{M} I(k)$$

```
#poids des variables_modalités (points modalités)
poids_moda = somme_col/(n*p)

#distance à l'origine des points modalités
disto_moda = numpy.apply_along_axis(arr=X.values/somme_col,axis=0,func1d=lambda x:
numpy.sum(n*(x-moda_moy)**2))

#inertie
inertie_moda = poids_moda * disto_moda

#affichage
print(pandas.DataFrame(numpy.transpose([poids_moda,disto_moda,inertie_moda]),index
=X.columns,columns=['Poids','Disto','Inertie']))

            Poids     Disto   Inertie
Taill-    0.125000  1.666667  0.208333
Taille+   0.083333  3.000000  0.250000
Taille++  0.125000  1.666667  0.208333
Velo-     0.125000  1.666667  0.208333
Veloc+    0.125000  1.666667  0.208333
Veloc++   0.083333  3.000000  0.250000
Affe-     0.083333  3.000000  0.250000
Affec+    0.250000  0.333333  0.083333
```

```
#inertie totale des points-modalités
inertie_tot_moda = numpy.sum(inertie_moda)
print(inertie_tot_moda)

1.6666666666666665
```

La valeur de l'inertie totale obtenue via le point de vue des modalités est strictement identique à celle exprimée à partir de l'analyse des proximités entre les individus (section 5.1.2.3). Nous avons deux manières d'appréhender les données, mais il s'agit bien des mêmes données.

Nous constatons de surcroît que les modalités (Taille+, Veloc++, Affec-) sont celles qui contribuent le plus à l'information globale.

5.1.3.4 Principe de l'analyse des correspondances multiples – 2

Toujours dans l'optique de produire un système de représentation qui préserve au mieux les distances entre les modalités, l'ACM produit une succession de facteurs qui permettent de les discerner au mieux c.-à-d. qui maximisent la variance de leurs coordonnées factorielles.

Pour le 1er facteur :

$$\lambda_1 = \sum_{k=1}^{M} \omega_k \times G_{k1}^2$$

Où :

- ω_k est le poids relatif de la modalité.
- G_{kh} la coordonnée de la modalité « k » sur le facteur « h ».
- λ_h est la variance associée au facteur « h », avec exactement la même valeur que son homologue obtenue via l'analyse des proximités entre les individus.

Le 2nd facteur cherche à « modéliser » l'information non captée sur le 1er facteur $(I - \lambda_1)$, etc. Le principe de la décomposition orthogonale reste valable. Les valeurs propres s'additionnent.

5.1.3.5 Quelques résultats remarquables

A ce stade de notre présentation, considérons quelques résultats importants.

- Le poids d'une modalité dépend de sa fréquence.

$$\omega_k = \frac{n_k}{n \times p}$$

- La distance à l'origine peut s'écrire plus simplement. On se rend compte alors qu'une modalité est d'autant plus distante du profil moyen qu'elle est rare.

$$d^2(k) = \frac{n}{n_k} - 1$$

- De fait, l'écriture de l'inertie d'une modalité est également simplifiée. Nous constatons qu'une modalité contribue d'autant plus à l'inertie quand elle est rare.

$$I(k) = \omega_k \times d^2(k) = \frac{1}{p}(1 - \frac{n_k}{n})$$

- Et la contribution d'une variable à l'inertie globale est fonction du nombre de ses modalités.

$$I(j) = \sum_{k=1}^{m_j} d^2(k) = \frac{1}{p}(m_j - 1)$$

- L'inertie totale ne dépend que des caractéristiques des variables : « p » leur nombre ; « M » le nombre total des modalités ; le nombre moyen de modalités par variable $(\frac{M}{p})$.

$$I = \sum_{j=1}^{p} I(j) = \frac{M}{p} - 1$$

! Ainsi, il faut se méfier des effets mécaniques qui peuvent fausser les calculs en ACM : (1) des variables à grand nombre de modalités ; (2) des modalités rares.

5.1.4 Analyse des variables

Puisque les termes de la variance restituée par les facteurs s'additionnent sur les modalités, nous pouvons les regrouper selon les variables. Pour le 1er facteur, mais l'idée est généralisable à tout facteur « h » :

$$\lambda_1 = \sum_{k=1}^{M} \omega_k G_{k1}^2$$

$$= \sum_{j=1}^{p} \sum_{k=b_j}^{b_j+m_j-1} \omega_k G_{k1}^2$$

Où $b_j = 1 + \sum_{l=1}^{j-1} m_l$

Puisque :

- G_{k1} est égale à la coordonnée des individus portant la modalité « k » sur le facteur F1.
- Elle est aussi égale, à un facteur près, à la moyenne conditionnelle de la modalité « k » sur l'axe factoriel.
- La moyenne des moyennes conditionnelles est nulle puisque les facteurs sont centrés.
- La variance totale est identique quelle que soit la variable considérée, c'est celle du facteur.

Nous poursuivons la réécrire de l'équation :

$$\lambda_1 = \sum_{j=1}^{p} \frac{1}{p} \sum_{k=b_j}^{b_j+m_j-1} \frac{n_k}{n} G_{k1}^2$$

$$= \frac{1}{p} \sum_{j=1}^{p} \eta^2(F_1, X_j)$$

Ainsi, nous pouvons voir l'ACM comme une technique visant à maximiser la moyenne des rapports de corrélation des variables sur les axes factoriels. La valeur (λ_h) obtenue correspond à la variance restituée par le facteur (Tenenhaus, 2007, page 260).

On note alors que ($\lambda_h \leq 1$) puisque les carrés des rapports de corrélation le sont [$\eta^2(F_h, X_j) \leq 1$].

Avec l'objet « acm » de la librairie « fanalysis », nous récupérons les coordonnées factorielles des points modalités. Nous vérifions que la somme pondérée des carrés des coordonnées correspond bien à la variance du facteur c.-à-d. sa valeur propre.

```
#coordonées des modalités sur le 1er facteur
print(pandas.DataFrame(acm.col_coord_[:,0],index=X.columns))
```

```
                     0
Taill-    4.558738e-01
Taille+  -1.367621e+00
Taille++  4.558738e-01
Velo-     1.081461e+00
Veloc+   -1.081461e+00
Veloc++   7.058942e-15
Affe-     1.367621e+00
Affec+   -4.558738e-01
```

```
#inertie du 1er facteur
lambda_1_moda = numpy.sum(poids_moda*acm.col_coord_[:,0]**2)
print(lambda_1_moda)
```

```
0.7080312581410024
```

Nous calculons la moyenne des coordonnées pour chaque variable. Elle est effectivement nulle.

```
#moyenne des modalités de Taille
print(numpy.mean(poids_moda[:3]*acm.col_coord_[:3,0]))
```

```
1.3877787807814457e-17
```

```
#moyenne des modalités de Velocite
print(numpy.mean(poids_moda[3:6]*acm.col_coord_[3:6,0]))
```

```
-1.6711016658649122e-17
```

```
#moyenne des modalités de Affec
print(numpy.mean(poids_moda[6:]*acm.col_coord_[6:,0]))
```

```
-1.3877787807814457e-17
```

Nous appliquons la formule ci-dessus pour produire les carrés des rapports de corrélation par variable.

```
#rapp.correl. des Taille
rc_Taille = numpy.sum(somme_col[:3]/n*acm.col_coord_[:3,0]**2)
print(rc_Taille)
```

```
0.6234627599261655
```

```
#rapp.correl. des Velocite
rc_Velocite = numpy.sum(somme_col[3:6]/n*acm.col_coord_[3:6,0]**2)
print(rc_Velocite)
```

```
0.8771682545706797
```

```
#rapp.correl. des Affec
rc_Affection = numpy.sum(somme_col[6:]/n*acm.col_coord_[6:,0]**2)
print(rc_Affection)
```

```
0.6234627599261622
```

Dont nous en tirons la moyenne non-pondérée.

```
#moyenne des rapports de corrélation
print(1/p*numpy.sum(numpy.array([rc_Taille,rc_Velocite,rc_Affection])))
```

```
0.7080312581410025
```

Elle correspond à la valeur propre du facteur.

5.2 Organisation des calculs

Les données pouvant être organisées de diverses manières en analyse des correspondances multiples, nous pouvons l'implémenter soit via l'analyse factorielle des correspondances, soit via l'analyse en composantes principales.

5.2.1 ACM via une AFC sur la matrice des indicatrices

La représentation la plus naturelle est de passer par le tableau des indicatrices (section 5.1.1). Il est constitué de valeurs positives ou nulles, les marges lignes et colonnes ont un sens, les profils ont un sens également. Les conditions sont réunies pour pouvoir appliquer l'algorithme de l'analyse factorielle des correspondances. Nous analyserons par ce biais : les relations entre les modalités, les proximités entre les individus, et les associations individus-modalités. Soit exactement le propos de l'analyse des correspondances multiples.

Sous Python, nous passons par la classe CA du module « fanalysis ». Nous lui passons le tableau des indicatrices.

```
#** ACM via l'AFC sur le tableau des indicatrices **
from fanalysis.ca import CA
afc_1 = CA(row_labels=X.index,col_labels=X.columns)
afc_1.fit(X.values)

#valeurs propres
print(pandas.DataFrame(numpy.transpose(afc_1.eig_),index=range(1,8),columns=['Val.P','%','Cumsum(%)']))
```

```
          Val.P             %   Cumsum(%)
1  7.080313e-01  4.248188e+01   42.481875
2  5.914894e-01  3.548936e+01   77.971237
3  2.619918e-01  1.571951e+01   93.690746
4  6.974652e-02  4.184791e+00   97.875538
5  3.540771e-02  2.124462e+00  100.000000
6  4.326339e-33  2.595803e-31  100.000000
7  3.843959e-34  2.306375e-32  100.000000
```

L'AFC ne sait pas que nous lui avons passé une variante très particulière d'un tableau de contingence, comprenant de nombreuses redondances, raison pour laquelle elle produit deux facteurs en trop (n°6 et n°7) qui ont une variance nulle. Mis à part cela, les valeurs propres sont bien ceux de l'ACM obtenues à l'aide de l'objet MCA de « fanalysis ».

Les coordonnées des lignes (des observations) et des colonnes (des modalités) sont complètement raccords également.

```
#coordonnées fact. des obs. (1er plan)
print(pandas.DataFrame(afc_1.row_coord_[:,:2],index=X.index,columns=['Fact.1','Fact.2']))

                  Fact.1    Fact.2
Chien
Beauceron    4.924471e-15 -1.279924
Basset       1.150779e+00  0.799057
Berger All   4.763557e-15 -1.279924
Boxer       -1.150779e+00  0.385124
Bull-Dog     4.284137e-01  0.509675
Bull-Mastif  1.150779e+00 -0.028809
Caniche     -4.284137e-01  0.509675
Labrador    -1.150779e+00  0.385124

#coordonnées fact. des modalités. (1er plan)
print(pandas.DataFrame(afc_1.col_coord_[:,:2],index=X.columns,columns=['Fact.1','Fact.2']))

                Fact.1    Fact.2
Taill-    4.558738e-01  0.788128
Taille+  -1.367621e+00  0.500758
Taille++  4.558738e-01 -1.121966
Velo-     1.081461e+00  0.554740
Veloc+   -1.081461e+00  0.554740
Veloc++   7.058942e-15 -1.664220
Affe-     1.367621e+00  0.500758
Affec+   -4.558738e-01 -0.166919
```

Cette approche est certainement la plus simple pour obtenir les résultats de l'ACM si nous ne disposons pas d'une implémentation dédiée dans l'outil logiciel que nous utilisons.

5.2.2 ACM via une AFC sur le tableau de Burt

Une seconde voie consiste à travailler à partir du tableau de Burt qui est un tableau de contingence constitué du croisement de l'ensemble des variables. Il est symétrique. Sur les blocs diagonaux, nous avons le croisement des variables avec elles-mêmes (Figure 146). Ici aussi, la matrice est constituée de valeurs positives ou nulles, les marges et les profils sont interprétables. Les conditions sont réunies pour pouvoir appliquer l'AFC.

	Taille-	Taille+	Taille++	Veloc-	Veloc+	Veloc++	Affec-	Affec+
Taille-	3			2	1		1	2
Taille+		2			2			2
Taille++			3	1		2	1	2
Veloc-	2		1	3			2	1
Veloc+	1	2			3			3
Veloc++			2			2		2
Affec-	1		1	2			2	
Affec+	2	2	2	1	3	2		6

Figure 146 – Tableau de Burt – Données "Canidés"

A priori, nous analysons principalement les relations entre les modalités qui sont à la fois en ligne et colonne du tableau avec l'AFC. Mais il est possible de revenir sur les individus avec les relations de transition. Attention, les duplications étant multiples dans le tableau de Burt, les individus sont comptabilisés plusieurs fois, il faudra corriger les résultats de l'AFC.

Sous Python, nous formons le tableau de Burt à partir de la matrice des indicatrices, puis nous faisons appel à la classe de calcul CA dédiée à l'AFC.

```
#tableau de BURT
burt = numpy.dot(numpy.transpose(X.values),X.values)
print(burt)

[[3 0 0 2 1 0 1 2]
 [0 2 0 0 2 0 0 2]
 [0 0 3 1 0 2 1 2]
 [2 0 1 3 0 0 2 1]
 [1 2 0 0 3 0 0 3]
 [0 0 2 0 0 2 0 2]
 [1 0 1 2 0 0 2 0]
 [2 2 2 1 3 2 0 6]]

#AFC sur le tableau de Burt
afc_2 = CA(row_labels=X.columns,col_labels=X.columns)
afc_2.fit(burt)

#valeurs propres corrigées
print(numpy.sqrt(afc_2.eig_[0]))

[7.08031258e-01 5.91489364e-01 2.61991819e-01 6.97465196e-02
 3.54077063e-02 4.08887366e-17 2.21873573e-17]
```

Si (μ_h) sont les valeurs propres de l'AFC sur le tableau de Burt, celles de l'ACM s'écrivent :

$$\lambda_h = \sqrt{\mu_h}$$

Si (B_{kh}) sont les coordonnées des modalités issues de l'AFC sur le tableau de Burt, celles de l'ACM sont obtenues avec :

$$G_{kh} = \frac{B_{kh}}{\sqrt{\lambda_h}}$$

```
#coordonnées factorielles des points-modalités dans le 1er plan factoriel
temp = afc_2.col_coord_[:,:2]/numpy.sqrt(numpy.sqrt(afc_2.eig_[0][:2]))
print(pandas.DataFrame(temp,index=X.columns,columns=['Fact.1','Fact.2']))
```

```
               Fact.1    Fact.2
Taill-   -4.558738e-01  0.788128
Taille+   1.367621e+00  0.500758
Taille++ -4.558738e-01 -1.121966
Velo-    -1.081461e+00  0.554740
Veloc+    1.081461e+00  0.554740
Veloc++   2.012812e-15 -1.664220
Affe-    -1.367621e+00  0.500758
Affec+    4.558738e-01 -0.166919
```

5.2.3 ACM via une ACP sur le tableau des profils

La troisième piste enfin consiste à appliquer une analyse en composantes principales sur le tableau des profils. Nous faisons le choix des profils lignes ici, mais l'opération pour les profils colonnes est également possible.

Voyons les distances utilisée en ACM et ACP pour comprendre les connexions entre les deux approches. La distance du χ^2 entre 2 individus utilisée en ACM s'écrit :

$$d^2_{ACM}(i,i') = \sum_{k=1}^{M} \frac{1}{\frac{n_k}{n \times p}} \left(\frac{x_{ik}}{p} - \frac{x_{i'k}}{p} \right)^2$$

Pour l'ACP normée, nous avons :

$$d^2_{ACP}(i,i') = \sum_{k=1}^{M} \frac{1}{\sigma_k^2} (x_{ik} - x_{i'k})^2$$

Où (σ_k^2) est la variance de la k[ème] colonne des indicatrices. Son expression peut être simplifiée :

$$\sigma_k^2 = \frac{n_k(n - n_k)}{n^2}$$

Pour qu'il y ait équivalence avec l'ACM, il faudrait pondérer la k[ème] indicatrice avec (u_k) où :

$$u_k = \frac{n - n_k}{n \times p}$$

Nous pourrons dès lors obtenir les résultats de l'ACM via un programme d'ACP en appliquant cette pondération sur les indicatrices :

$$d^2_{ACP \rightarrow ACM} = \sum_{k=1}^{M} \frac{u_k}{\sigma_k^2} (x_{ik} - x_{i'k})^2$$

Sous Python, nous créons les profils lignes à partir du tableau des indicatrices.

```
#tableau des profils lignes
profil = numpy.apply_along_axis(arr=X.values,axis=1,func1d=lambda x:x/numpy.sum(x))
print(pandas.DataFrame(profil,index=X.index,columns=X.columns))
```

```
               Taill-   Taille+  Taille++     Velo-    Veloc+   Veloc++  \
Chien
Beauceron    0.000000  0.000000  0.333333  0.000000  0.000000  0.333333
Basset       0.333333  0.000000  0.000000  0.333333  0.000000  0.000000
Berger All   0.000000  0.000000  0.333333  0.000000  0.000000  0.333333
Boxer        0.000000  0.333333  0.000000  0.000000  0.333333  0.000000
Bull-Dog     0.333333  0.000000  0.000000  0.333333  0.000000  0.000000
Bull-Mastif  0.000000  0.000000  0.333333  0.333333  0.000000  0.000000
Caniche      0.333333  0.000000  0.000000  0.000000  0.333333  0.000000
Labrador     0.000000  0.333333  0.000000  0.000000  0.333333  0.000000

                Affe-    Affec+
Chien
Beauceron    0.000000  0.333333
Basset       0.333333  0.000000
Berger All   0.000000  0.333333
Boxer        0.000000  0.333333
Bull-Dog     0.000000  0.333333
Bull-Mastif  0.333333  0.000000
Caniche      0.000000  0.333333
Labrador     0.000000  0.333333
```

Nous réduisons les variables par la racine carrée de leur variance c.-à-d. l'écart-type.

```
#réduire les profils
profil = profil/numpy.std(profil,axis=0,ddof=0)
print(profil)
```

```
[[0.         0.         2.06559112 0.         0.         2.30940108
  0.         2.30940108]
 [2.06559112 0.         0.         2.06559112 0.         0.
  2.30940108 0.        ]
 [0.         0.         2.06559112 0.         0.         2.30940108
  0.         2.30940108]
 [0.         2.30940108 0.         0.         2.06559112 0.
  0.         2.30940108]
 [2.06559112 0.         0.         2.06559112 0.         0.
```

```
 0.         2.30940108]
[0.         0.         2.06559112 2.06559112 0.         0.
 2.30940108 0.        ]
[2.06559112 0.         0.         0.         2.06559112 0.
 0.         2.30940108]
[0.         2.30940108 0.         0.         2.06559112 0.
 0.         2.30940108]]
```

Nous calculons la pondération des modalités (u_k). Nous l'appliquons aux données en multipliant les colonnes par sa racine carrée. Remarque : cette manipulation n'est pas nécessaire si l'ACP implémentée sait prendre en compte les pondérations des variables (voir Cours ACM, où nous utilisons le package « FactoMineR » sous R, page 24).

```
#pondération des modalités
pond_moda = (n-somme_col)/(n*p)
print(pond_moda)

[0.20833333 0.25       0.20833333 0.20833333 0.20833333 0.25
 0.25       0.08333333]

#appliquer la pondération aux profils
profil = profil*numpy.sqrt(pond_moda)
print(profil)

[[0.         0.         0.94280904 0.         0.         1.15470054
  0.         0.66666667]
 [0.94280904 0.         0.         0.94280904 0.         0.
  1.15470054 0.        ]
 [0.         0.         0.94280904 0.         0.         1.15470054
  0.         0.66666667]
 [0.         1.15470054 0.         0.         0.94280904 0.
  0.         0.66666667]
 [0.94280904 0.         0.         0.94280904 0.         0.
  0.         0.66666667]
 [0.         0.         0.94280904 0.94280904 0.         0.
  1.15470054 0.        ]
 [0.94280904 0.         0.         0.         0.94280904 0.
  0.         0.66666667]
 [0.         1.15470054 0.         0.         0.94280904 0.
  0.         0.66666667]]
```

Il nous reste à faire appel à l'ACP avec la classe de calcul PCA de « fanalysis ». Nous utilisons une ACP non normée (std_unit = False) puisque les données ont été explicitement préparées en amont.

```
#lancer une ACP
from fanalysis.pca import PCA
acp = PCA(std_unit=False,row_labels=X.index,col_labels=X.columns)
acp.fit(profil)

#valeurs propres
print(acp.eig_[0])
```

```
[7.08031258e-01 5.91489364e-01 2.61991819e-01 6.97465196e-02
 3.54077063e-02 4.69842982e-33 4.02307068e-34 4.04188030e-35]
```

Nous avons les bonnes valeurs propres. L'ACP ne sait pas qu'il y a des redondances artificielles dans les données, d'où les facteurs aux variances nulles en excédent.

Les coordonnées factorielles des individus sont directement adéquates.

```
#coordonnées fact. des observations (1er plan)
print(pandas.DataFrame(acp.row_coord_[:,:2],index=X.index,columns=['Fact.1','Fact.2']))
```

```
                  Fact.1    Fact.2
Chien
Beauceron   -1.021387e-15 -1.279924
Basset       1.150779e+00  0.799057
Berger All  -1.100184e-15 -1.279924
Boxer       -1.150779e+00  0.385124
Bull-Dog     4.284137e-01  0.509675
Bull-Mastif  1.150779e+00 -0.028809
Caniche     -4.284137e-01  0.509675
Labrador    -1.150779e+00  0.385124
```

Nous pouvons passer de nouveau par les relations de transition si l'on souhaite produire les coordonnées des points-modalités.

5.3 Pratique de l'ACM avec « fanalysis » sous Python

A l'instar des chapitres précédents, nous déroulons une analyse complète avec la librairie « fanalysis » sous Python. Nous mettons l'accent sur les outils d'aide à l'interprétation qui permettent de mieux situer le rôle des individus et variables-modalités dans la lecture des axes factoriels. A la différence de l'ACP, il sera possible de projeter dans le même repère les individus et les points-modalités, et raisonner en termes d'attractions et répulsions. Nous nous pencherons enfin sur le traitement des variables illustratives et des individus supplémentaires.

Pour plus de clarté, nous recommençons l'analyse à partir de l'importation des données. Nous isolons les variables actives et nous récupérons les principales informations.

```
#chargement - index_col = 0 pour indiquer que la colonne n°0 est un label
import pandas
D = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="ACM_CANINES",index_col=0)

#affichage des caractéristiques
print(D.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8 entries, Beauceron to Labrador
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Taille     8 non-null      object
 1   Velocite   8 non-null      object
 2   Affection  8 non-null      object
 3   Cote       8 non-null      float64
 4   Fonction   8 non-null      object
dtypes: float64(1), object(4)
memory usage: 384.0+ bytes
```

```
#récupération des variables actives
DActives = D[['Taille','Velocite','Affection']]
print(DActives)
```

```
              Taille Velocite Affection
Chien
Beauceron    Taille++  Veloc++    Affec+
Basset         Taill-    Velo-     Affe-
Berger All   Taille++  Veloc++    Affec+
Boxer         Taille+   Veloc+    Affec+
Bull-Dog       Taill-    Velo-    Affec+
Bull-Mastif  Taille++    Velo-     Affe-
Caniche        Taill-   Veloc+    Affec+
Labrador      Taille+   Veloc+    Affec+
```

```
#récupération des infos - nombre de variables
p = DActives.shape[1]

#nombre d'observations
n = DActives.shape[0]

#codage en 0/1
X = pandas.get_dummies(DActives,prefix='',prefix_sep='')
print(X)
```

```
             Taill-  Taille+  Taille++  Velo-  Veloc+  Veloc++  Affe-  Affec+
Chien
Beauceron         0        0         1      0       0        1      0       1
Basset            1        0         0      1       0        0      1       0
Berger All        0        0         1      0       0        1      0       1
Boxer             0        1         0      0       1        0      0       1
Bull-Dog          1        0         0      1       0        0      0       1
Bull-Mastif       0        0         1      1       0        0      1       0
Caniche           1        0         0      0       1        0      0       1
Labrador          0        1         0      0       1        0      0       1
```

```
#nombre total de modalités
M = X.shape[1]
```

Nous lançons l'ACM avec la classe de calcul MCA. Nous affichons les propriétés de l'objet.

```
#analyse avec fanalysis
from fanalysis.mca import MCA

#instanciation
```

```
acm = MCA(row_labels=DActives.index,var_labels=DActives.columns)
acm.fit(DActives.values)

#propriétés
print(dir(acm))
```

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format
__', '__ge__', '__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__
', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__
reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setstate__', '__sizeof__
', '__str__', '__subclasshook__', '__weakref__', '_binarization', '_compute_stats'
, '_compute_svd', '_get_param_names', '_get_tags', '_more_tags', 'c_', 'col_contri
b_', 'col_coord_', 'col_cos2_', 'col_labels', 'col_labels_', 'col_labels_short_',
'col_labels_short_temp_', 'col_labels_temp_', 'col_topandas', 'eig_', 'fit', 'fit_
transform', 'get_params', 'mapping', 'mapping_col', 'mapping_row', 'model_', 'n_',
'n_categories_', 'n_components', 'n_components_', 'n_vars_', 'plot_col_contrib', '
plot_col_cos2', 'plot_eigenvalues', 'plot_row_contrib', 'plot_row_cos2', 'prefixes
_', 'r_', 'row_contrib_', 'row_coord_', 'row_cos2_', 'row_labels', 'row_labels_',
'row_topandas', 'set_params', 'stats', 'transform', 'var_labels']
```

5.3.1 Nombre de facteurs 1

L'identification des facteurs pertinents reste toujours un problème difficile en analyse factorielle. Tenons-nous en aux approches classiques basées sur l'étude des valeurs propres dans un premier temps. Nous les affichons ainsi que les pourcentages d'inertie associées aux facteurs, individuelles et cumulées. Rappelons que le nombre maximum de facteurs est (Hmax = M − p = 8 − 3 = 5).

```
#nombre max de facteurs
Hmax = M-p

#numpy
import numpy

#valeurs propres
print(pandas.DataFrame(numpy.transpose(acm.eig_),columns=['Val.P','%','Cumul %'],index=range(1,Hmax+1)))
```

```
      Val.P          %     Cumul %
1  0.708031  42.481875   42.481875
2  0.591489  35.489362   77.971237
3  0.261992  15.719509   93.690746
4  0.069747   4.184791   97.875538
5  0.035408   2.124462  100.000000
```

Règle de Kaiser. Avec la règle de Kaiser, nous sélectionnons les facteurs portés par au moins la moitié de l'inertie totale (I), soit :

$$\frac{I}{H_{max}} = \frac{\frac{M}{p} - 1}{M - p} = \frac{\frac{M-p}{p}}{M-p} = \frac{1}{p} = \frac{1}{3} = 0.3333$$

Les (H = 2) premiers facteurs seraient donc les seules pertinentes avec des variances égales respectivement à 0.708031 et 0.591489. Ils restituent 77.97% de l'information disponible.

Remarque : Notre exemple ne s'y prête pas trop parce que le nombre de variables est très faible. En réalité, les pourcentages d'inerties – pris tels quels – sont rarement concentrés sur les premiers facteurs en ACM. L'information semble éparpillée. Nous verrons pourquoi plus loin (section 5.3.2). La règle de Kaiser est dès lors trop permissive, enjoignant à conserver un nombre excessif de facteurs.

Eboulis des valeurs propres. Voyons maintenant ce que nous dit l'éboulis des valeurs propres.

```
#librairie graphique
import matplotlib.pyplot as plt

#éboulis des v.p.
fix,ax = plt.subplots(figsize=(5,5))
ax.plot(range(1,Hmax+1),acm.eig_[0],".-")
ax.set_xlabel("Nb. facteurs")
ax.set_ylabel("Val. propres")
plt.title("Eboulis des valeurs propres")

#seuil - Règle de Kaiser
ax.plot([1,Hmax],[1/p,1/p],"r--",linewidth=1)
plt.show()
```

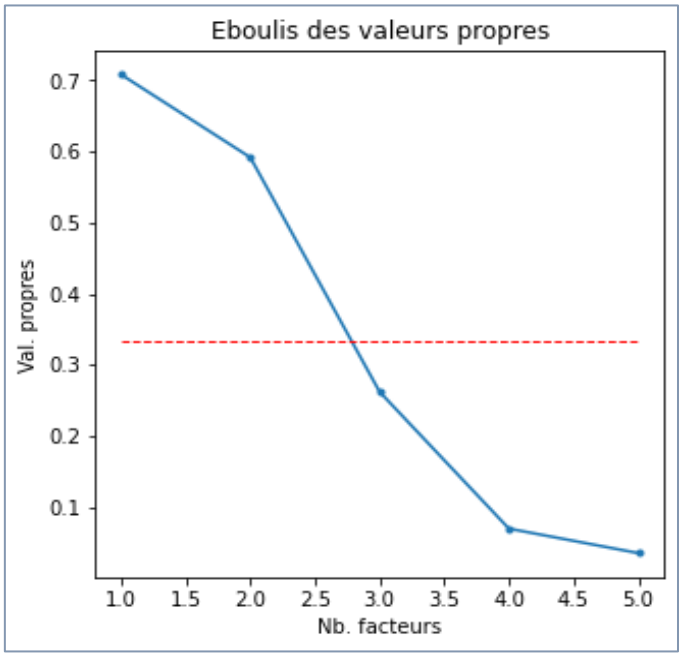


Figure 147 – Eboulis des valeurs propres – ACM – Données "Canidés"

Avec la règle du coude, il faudrait sélection (H = 3) ou (H = 4) facteurs. Ces suggestions semblent très discutables. Et bien souvent en pratique, le diagramme des valeurs propres descend en

pente douce en ACM, pour les mêmes raisons évoquées précédemment : on observe une dispersion apparente des proportions d'inertie restituées sur premiers facteurs.

En effet, de par la nature des données présentées à l'algorithme de calcul – les colonnes sont démultipliées dans le codage disjonctif, certaines sont redondantes, ou encore des croisements sont superflus dans le tableau de Burt – il est difficile de concentrer de l'inertie sur les premiers facteurs. Si l'on considère le premier facteur, nous avons vu que $(\lambda_1 \leq 1)$ par construction, nous y concentrons au mieux $(\frac{1}{I} \times 100)$ % de l'inertie, une proportion d'autant plus faible que le nombre total M de modalités augmente (puisque $I = \frac{M}{p} - 1$, rappelons-le). Pour disposer d'une indications plus réaliste sur la qualité des facteurs, nous devons utiliser un indicateur corrigé qui élimine le biais due à la redondance artificielle introduite dans les données. C'est le propos de la « correction de Benzécri ».

5.3.2 Nombre de facteurs 2 – Correction de Benzécri

Principe de la correction. La correction de Benzécri s'appuie sur l'idée qu'une partie de l'information est redondante dans les données présentées à l'algorithme de l'ACM. L'expression ajustée – défalquée des redondances – pour mieux rendre compte de l'inertie restituée par le facteur n°h s'écrit :

$$\lambda'_h = \left[\left(\frac{p}{p-1}\right) \times \left(\lambda_h - \frac{1}{p}\right)\right]^2$$

Cet ajustement n'a de sens que pour les facteurs portant une valeur propre initiale supérieure à la moyenne,

$$\lambda_h > \frac{1}{p}$$

Mise en œuvre de la correction de Benzécri. En pratique, la démarche suivante est appliquée pour produire les valeurs corrigés des variances restituées et des proportions qui en découlent.

1. Calculer le seuil $(\frac{1}{p})$

2. Pour les facteurs dont la valeur propre (λ_h) est supérieure au seuil, appliquer la correction à l'aide de l'expression ci-dessus.
3. Faire la somme (S') des (λ'_h)
4. Calculer les pourcentages d'inerties individuelles et cumulées et rapportant (λ'_h) à (S')

Voyons cela sur les données « Canidés ». Le tableau des valeurs propres intégrant la correction se présenterait comme suit (Figure 148) :

$$\lambda'_h = \left[\left(\frac{p}{p-1}\right)\times\left(\lambda_h - \frac{1}{p}\right)\right]^2 \qquad \frac{\lambda'_h}{S'}$$

	p	3			
	Seuil	0.3333		Benzécri	
	Axis	Lambda	Lambda'	% expliquée	%cumulée
2 facteurs ont une v.p. supérieures à la moyenne	1	0.7080	0.3159	67.8%	67.8%
	2	0.5915	0.1499	32.2%	100.0%
	3	0.2620			
	4	0.0697			
	5	0.0354			
	Somme	1.6667	0.4658461		

$$S' = \sum_h \lambda'_h$$

Figure 148 - Correction de Benzécri - ACM - Données "Canidés"

Nous reproduisons les calculs sous Python. Deux facteurs présentent des valeurs propres supérieures à la moyenne. Nous appliquons la correction et nous en déduisons les pourcentages individuels et cumulés corrigés.

```
#récupérer les valeurs propres supérieur à (1/p)
lambada = acm.eig_[0][acm.eig_[0]>1/p]
print(lambada)

[0.70803126 0.59148936]

#appliquer la correction
lambada_prim = ((p/(p-1))*(lambada-1/p))**2
print(lambada_prim)

[0.3158967  0.14995021]

#faire la somme
S_prim = numpy.sum(lambada_prim)
print(S_prim)
```

```
0.4658469098168333

#et produire les pourcentages
percent_prim = lambada_prim/S_prim*100

#affichage
print(pandas.DataFrame(numpy.transpose(numpy.array([lambada_prim,percent_prim,nump
y.cumsum(percent_prim)])),columns=['Val.P','%','Cumul %'],index=range(1,3)))


      Val.P          %     Cumul %
1  0.315897  67.811269   67.811269
2  0.149950  32.188731  100.000000
```

Remarque : Notre exemple ne s'y prête pas vraiment encore une fois, le nombre de variables est trop faible. Mais, avec ce dispositif de correction, l'éboulis des valeurs propres donne des indications plus intéressantes, avec souvent un coude plus marqué, permettant d'identifier plus facilement le bloc des facteurs pertinents pour l'analyse des résultats.

5.3.3 Pourcentage d'inertie restituée – Correction de Greenacre

La correction de Greenacre (« Theory and Applications of Correspondence Analysis », London : Academic Press, 1984), s'appuie sur la correction de Benzécri mais reconsidère la proportion d'inertie portée par les facteurs. Une partie de l'information est triviale dans le tableau de Burt, il s'agit du croisement endogène de chaque variable. Dans le graphique ci-dessous, nous affichons les inerties associées à chaque bloc de croisement (Figure 149).

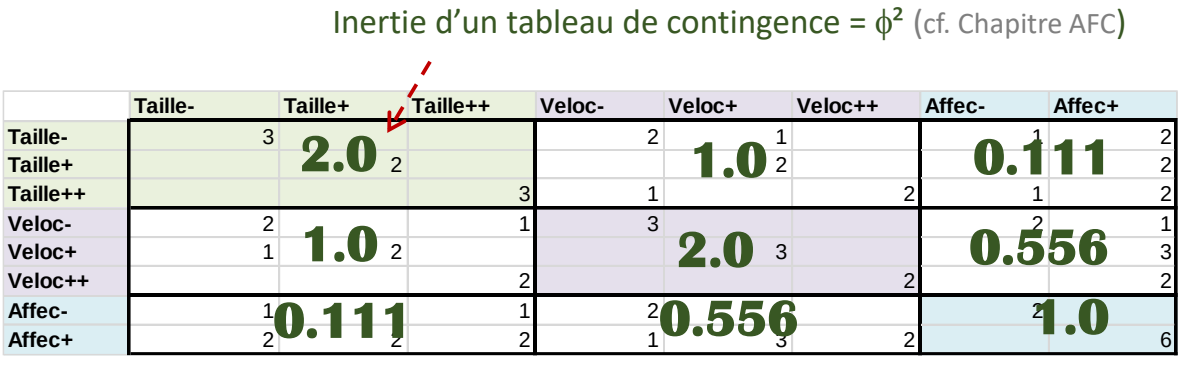


Figure 149 – Inerties associées aux blocs du tableau de Burt – Données "Canidés"

Rappelons que l'inertie du tableau de Burt équivaut à la somme des valeurs propres de l'AFC qui lui est appliquée. Elle est égale aussi à la somme des carrés des valeurs propres de l'ACM sur le tableau des indicatrices (section 5.2.2).

$$I_{Burt} = \sum_{h=1}^{H_{max}} \mu_h$$

$$I_{Burt} = \sum_{h=1}^{H_{max}} \lambda_h^2 = 0.708^2 + 0.591^2 + \cdots + 0.035^2 = 0.926$$

Nous disposons d'un autre prisme. La même quantité correspond à la moyenne des inerties des sous-tableaux qui composent le tableau de Burt (Figure 149).

$$I_{Burt} = \frac{2.0 + 1.0 + 0.111 + 1.0 + 2.0 + \cdots + 1.0}{9} = 0.926$$

Si nous souhaitons nous concentrer sur l'information « utile », nous retirons les blocs diagonaux représentant les croisements endogène des variables. L'information à traiter devient :

$$S'' = \frac{1.0 + 0.111 + 1.0 + \cdots + 0.556}{6} = 0.556 = \frac{p}{p-1}\left(I_{Burt} - \frac{M-p}{p^2}\right)$$

Greenacre, plutôt que de forcer artificiellement la somme cumulée des proportions à 100% sur les facteurs respectant la condition ($\lambda_h > \frac{1}{p}$), propose de diviser la valeur propre corrigée (λ'_h) par (S'') pour disposer d'une vision moins optimiste de la qualité des facteurs. Le pourcentage corrigé de variance restituée par le facteur s'écrit :

$$\tau''_h = \frac{\lambda'_h}{S''} \times 100$$

Sous Python, nous calculons la somme (S'') tout d'abord, puis nous formons les pourcentages d'inertie individuelles et corrigées.

```
#somme corrigée de Greenacre (S'')
S2nd = p/(p-1)*(numpy.sum(acm.eig_[0]**2)-(M-p)/(p**2))
print(S2nd)

0.5555555555555577

#pourcentage corrigé Greenacre
percent_2nd = lambada_prim/S2nd

#affichage
print(pandas.DataFrame(numpy.transpose(numpy.array([lambada_prim,percent_2nd,numpy
.cumsum(percent_2nd)])),columns=['Val.P','%','Cumul %'],index=range(1,3)))
```

```
      Val.P         %   Cumul %
1  0.315897  0.568614  0.568614
2  0.149950  0.269910  0.838524
```

Les valeurs propres ne sont pas modifiées. En revanche, nous constatons maintenant que les 2 premiers facteurs restituent en réalité 83.85% de l'information non-redondante, et non pas 100% comme nous l'indiquions dans la section précédente.

5.3.4 Analyse des modalités

5.3.4.1 Coordonnées factorielles

L'objet fournit les coordonnées des points-modalités...

```
#affichage des coordonnées
print(pandas.DataFrame(acm.col_coord_[:,:2],index=X.columns,columns=['Coord.F1','Coord.F2']))
```

```
              Coord.F1  Coord.F2
Taill-    4.558738e-01  0.788128
Taille+  -1.367621e+00  0.500758
Taille++  4.558738e-01 -1.121966
Velo-     1.081461e+00  0.554740
Veloc+   -1.081461e+00  0.554740
Veloc++   7.058942e-15 -1.664220
Affe-     1.367621e+00  0.500758
Affec+   -4.558738e-01 -0.166919
```

..., nous nous empressons de réaliser une représentation graphique pour situer les proximités (Figure 150).

```
#représentation dans le plan
fix,ax = plt.subplots(figsize=(7,7))
ax.axis([-2,+2,-2,+2])
ax.plot([-2,+2],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-2,+2],color='silver',linestyle='--')
ax.set_xlabel("Dim.1 (67.8%)")
ax.set_ylabel("Dim.2 (32.2%)")
plt.title("Représentation des points-modalités")

for i in range(X.shape[1]):
  ax.text(acm.col_coord_[i,0],acm.col_coord_[i,1],X.columns[i],color='dodgerblue')

plt.show()
```

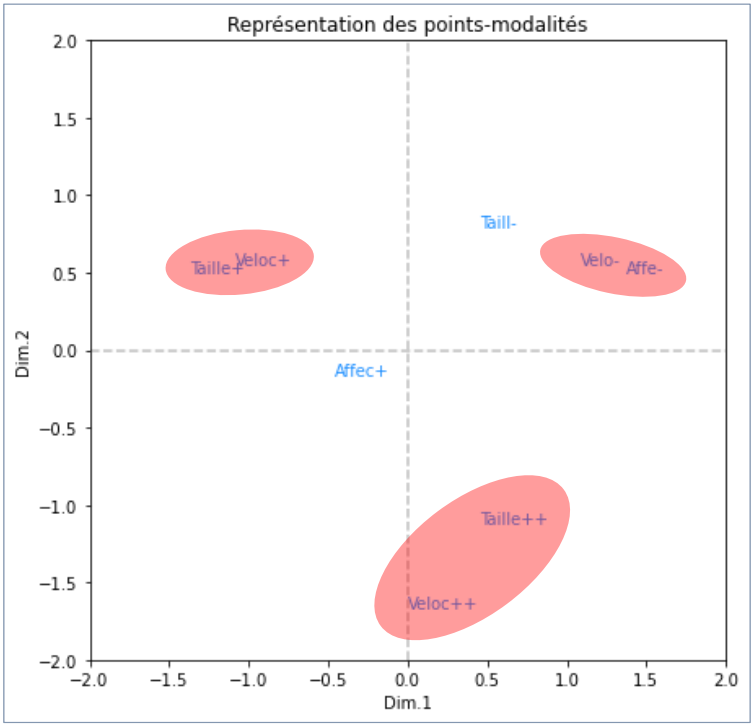


Figure 150 – Représentation des points modalités – ACM – Données "Canidés"

Il faut toujours se méfier des proximités dans les représentations factorielles (soulignées en rouge dans le graphique). Pour confirmer les perceptions visuelles, nous réalisons les tableaux croisés entre les variables concernées dans TANAGRA et nous calculons les contributions au χ^2.

		Stat	Value
Taille	Velocite	d.f.	4
		Tschuprow's t	0.707107
		Cramer's v	0.707107
		Phi²	1.000000
		Chi² (p-value)	8.00 (0.0916)
		Lambda	0.600000
		Tau (p-value)	0.4921 (0.1419)
		U(R/C) (p-value)	0.5589 (0.0462)

	Veloc++	Veloc-	Veloc+	Sum
Taille++	2 (+ 26 %)	1 (- 0 %)	0 (- 14 %)	3
Taille-	0 (- 9 %)	2 (+ 9 %)	1 (- 0 %)	3
Taille+	0 (- 6 %)	0 (- 9 %)	2 (+ 26 %)	2
Sum	2	3	3	8 *100%*

Figure 151 – Croisement (Taille, Vélocité) – Données "Canidés"

La liaison entre « Taille » et « Vélocité » semble effective (v de Cramer = 0.707) (Figure 151). Elle repose effectivement sur la double attraction entre (Taille++, Vélocité++) et (Taille+, Vélocité+) que l'on retrouve dans la représentation factorielle.

Remarque : Ça devrait être un réflexe en analyse factorielle. Il faut toujours revenir aux données pour vérifier (valider) les suggestions fournies par les représentations graphiques.

Pour ce qui est de la paire (Vélocité-, Affection-), la liaison entre les variables semble solide itou (v de Cramer = 0.745), elle est bien basée sur l'attraction entre ces modalités (Figure 152).

		Stat	Value
Velocite	Affection	d.f.	2
		Tschuprow's t	0.626767
		Cramer's v	0.745356
		Phi²	0.555556
		Chi² (p-value)	4.44 (0.1084)
		Lambda	0.400000
		Tau (p-value)	0.3016 (0.1211)
		U(R/C) (p-value)	0.2991 (0.0751)

	Affec+	Affec-	Sum
Veloc++	2 (+ 4 %)	0 (- 11 %)	2
Veloc-	1 (- 16 %)	2 (+ 47 %)	3
Veloc+	3 (+ 6 %)	0 (- 17 %)	3
Sum	6	2	8 *100%*

Figure 152 - Croisement (Vélocité, Affection) - ACM - Données "Canidés"

5.3.4.2 Contributions

Les contributions des modalités quantifient leur influence dans la construction des facteurs. Comme dans les autres analyses (ACP, AFC), elles s'additionnent et la somme intra-facteur est égale à 100% si l'on considère l'ensemble des items.

Pour l'ACM, la contribution de la modalité « k » au facteur « h » est définie par le ratio entre le carré de la coordonnée et l'importance du facteur traduite par sa variance restituée :

$$CTR_{kh} = \frac{\omega_k \times G_{kh}^2}{\lambda_h}$$

Nous affichons les contributions pour les deux premiers facteurs.

```
#affichage des contributions
print(pandas.DataFrame(acm.col_contrib_[:,:2],index=X.columns,columns=['Contrib.F1
','Contrib.F2']))
```

```
           Contrib.F1  Contrib.F2
Taill-    3.668993e+00   13.126725
Taille+   2.201396e+01    3.532865
Taille++  3.668993e+00   26.602515
Velo-     2.064806e+01    6.503426
Veloc+    2.064806e+01    6.503426
Veloc++   5.864697e-28   39.020556
Affe-     2.201396e+01    3.532865
Affec+    7.337985e+00    1.177622
```

Sans surprises, les modalités situées aux extrémités du repère sont les plus influentes. Le premier facteur est déterminé par l'opposition (Taille+, Vélocité+) vs. (Vélocité-, Affection-).

L'objet « acm » propose une visualisation graphique (et surtout triée) des contributions, très pratique lorsque le nombre de modalités est très élevé. Pour le 1er facteur :

```
#ou bien graphiquement
acm.plot_col_contrib(num_axis=1)
```

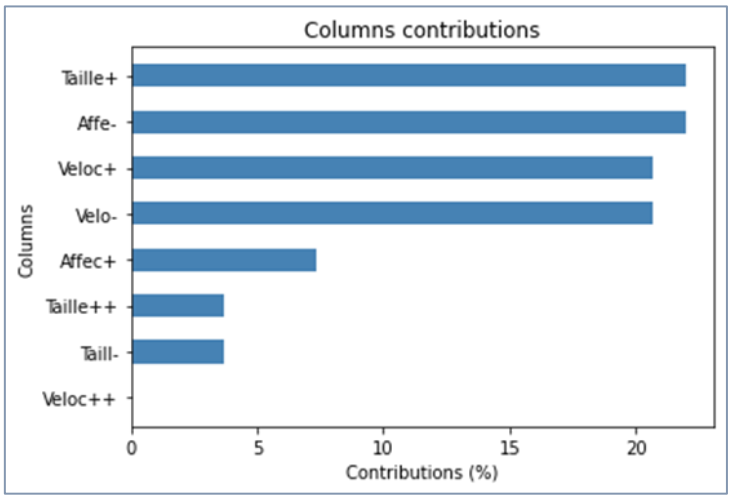


Figure 153 - Contributions des points-modalités sur le 1er facteur - ACM - Données "Canidés"

5.3.4.3 COS2

Les COS2 (cosinus carrés) représentent la qualité de représentation des modalités sur les facteurs, pris individuellement ou cumulés. Ils nous donnent une indication sur l'information des modalités captée par les facteurs. D'une certaine manière, les COS2 constituent un bon repère pour identifier le nombre adéquat de facteurs à prendre en compte dans l'analyse.

Comme les contributions, le COS2 de la modalité « k » sur le facteur « h » est basé sur le carré des coordonnées, mais normalisé par sa distance à l'origine cette fois-ci :

$$COS^2_{kh} = \frac{G^2_{kh}}{d^2(k)}$$

Voici les COS2 des deux premiers facteurs pour les données « Canidés ».

```
#affichage des COS2
print(pandas.DataFrame(acm.col_cos2_[:,:2],index=X.columns,columns=['Cos2.F1','Cos2.F2']))
```

```
              Cos2.F1   Cos2.F2
Taill-    1.246926e-01  0.372687
Taille+   6.234628e-01  0.083586
Taille++  1.246926e-01  0.755285
Velo-     7.017346e-01  0.184642
Veloc+    7.017346e-01  0.184642
Veloc++   1.660956e-29  0.923210
Affe-     6.234628e-01  0.083586
Affec+    6.234628e-01  0.083586
```

Si l'on s'intéresse aux COS2 cumulés maintenant :

```
#affichage des COS2 cumulés
print(pandas.DataFrame(numpy.cumsum(acm.col_cos2_[:,:2],axis=1),index=X.columns,columns=['Cos2.F1','Cum(Cos2).F2']))
```

```
              Cos2.F1  Cum(Cos2).F2
Taill-    1.246926e-01      0.497380
Taille+   6.234628e-01      0.707049
Taille++  1.246926e-01      0.879978
Velo-     7.017346e-01      0.886377
Veloc+    7.017346e-01      0.886377
Veloc++   1.660956e-29      0.923210
Affe-     6.234628e-01      0.707049
Affec+    6.234628e-01      0.707049
```

Si l'on regarde les COS2 cumulés sur le 2nd facteur, mise à part (« Taille- » avec Cumul(COS2) = 0.49), les modalités sont assez bien représentées, signe que les deux premiers facteurs suffisent amplement pour capter l'information portée par les données.

Comme dans les autres approches (ACP, AFC), la qualité de représentation est importante pour apprécier la réalité des proximités dans l'espace factoriel.

5.3.4.4 Valeur test

Présent dans certains logiciels, la valeur test permet de caractériser l'intensité de l'écartement des modalités par rapport à l'origine. Nous en avions déjà parlé lorsqu'il s'agissait de situer les modalités des variables illustratives qualitatives en ACP (section 1.6.2). Son expression est directement dérivée de la coordonnée, modulée par l'effectif rattachée à la modalité :

$$VT_{kh} = G_{kh}\sqrt{\frac{(n-1)n_k}{n-n_k}}$$

Elle est distribuée – très approximativement – selon la loi normale. Elle permet par ce truchement d'initier un test de de l'écartement par rapport à l'origine. Prudence néanmoins, elle

a tendance à être optimiste (conclure systématiquement à un écart significatif) à mesure que les effectifs augmentent. Il vaut mieux l'utiliser essentiellement à titre indicatif.

Voici les valeurs-test des modalités pour le premier facteur.

```
#effectifs par modalité
print(acm.c_)

[[3. 2. 3. 3. 3. 2. 2. 6.]]

#calul des valeurs test par modalité - 1er facteur
vtest = acm.col_coord_[:,0]*numpy.sqrt(((n-1)*acm.c_[0])/(n-acm.c_[0]))
print(pandas.DataFrame(vtest,index=X.columns))

                     0
Taill-    9.342633e-01
Taille+  -2.089076e+00
Taille++  9.342633e-01
Velo-     2.216335e+00
Veloc+   -2.216335e+00
Veloc++   1.078271e-14
Affe-     2.089076e+00
Affec+   -2.089076e+00
```

En vérité, par rapport aux indicateurs usuels (coordonnées, contributions, cosinus carrés), la valeur test n'apporte pas grand-chose dans le contexte spécifique de l'analyse des correspondances multiples. Plus intéressant est le rapport de corrélation que nous verrons dans la prochaine section. Il permet de caractériser les facteurs à l'aide des variables, avec leurs modalités prises dans leur ensemble, et non plus des modalités prises individuellement.

5.3.4.5 Rapport de corrélation

Le carré du rapport de corrélation caractérise, pour chaque variable, la dispersion relative de ses modalités. Il se définit par le ratio entre la variance inter-modalités et la variance totale. Sachant que les facteurs sont centrés, et que leur variance totale équivaut à la valeur propre qui leur est associée, il s'écrit pour la variable n°j sur le facteur n°h :

$$\eta_{jh}^2 = \sum_{k \epsilon X_j} \frac{n_k}{n} G_{kh}^2 = p \times \lambda_h \times CTR_{jh}$$

Où (CTR_{jh}) est l'addition des contributions des modalités de la variable n°j au facteur n°h.

Pour les deux premiers facteurs des données « Canidés », nous avons :

```
#récupération des contribution sous une forme exploitable - 2 facteurs
contrib12 = pandas.DataFrame(acm.col_contrib_[:,:2],index=X.columns)
print(contrib12)
```

```
                     0          1
Taill-    3.668993e+00  13.126725
Taille+   2.201396e+01   3.532865
Taille++  3.668993e+00  26.602515
Velo-     2.064806e+01   6.503426
Veloc+    2.064806e+01   6.503426
Veloc++   5.864697e-28  39.020556
Affe-     2.201396e+01   3.532865
Affec+    7.337985e+00   1.177622
```

```
#**calcul des carrés des rapports de corrélation par variable sur les 2 facteurs**

#structure pour les eta2
eta2 = numpy.zeros((3,2))

#pour chaque facteur
for h in range(2):
    #pour chaque variable
    for j in range(3):
        eta2[j,h] = (p*acm.eig_[0][h]*numpy.sum(contrib12.loc[numpy.unique(D.iloc[:,j])][h]))/100

#affichage
print(pandas.DataFrame(eta2,index=D.columns[:3],columns=['F1','F2']))
```

```
                 F1        F2
Taille     0.623463  0.767672
Velocite   0.877168  0.923210
Affection  0.623463  0.083586
```

Ces nouvelles informations permettent d'initier un nouveau graphique de représentation des variables dans le repère factoriel, très intéressant lorsque leur nombre est élevé. Nous notons ainsi que (Figure 154) :

- Les modalités de « Vélocité » sont bien séparées à la fois sur les 1er (Vélocité+ vs. Vélocité-) et 2nd (Vélocité++ vs. les autres) facteurs.
- Les mêmes caractéristiques mais avec moins d'intensité sont constatées pour « Taille ».
- « Affection » en revanche est essentiellement liée au 1er facteur.

```
#réprésentation des variables dans le plan
fix,ax = plt.subplots(figsize=(5,5))
ax.axis([0,+1,0,+1])
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Carrés rapports de corrélation")

for i in range(3):
    ax.text(eta2[i,0],eta2[i,1],D.columns[i],color='darkcyan')

plt.show()
```

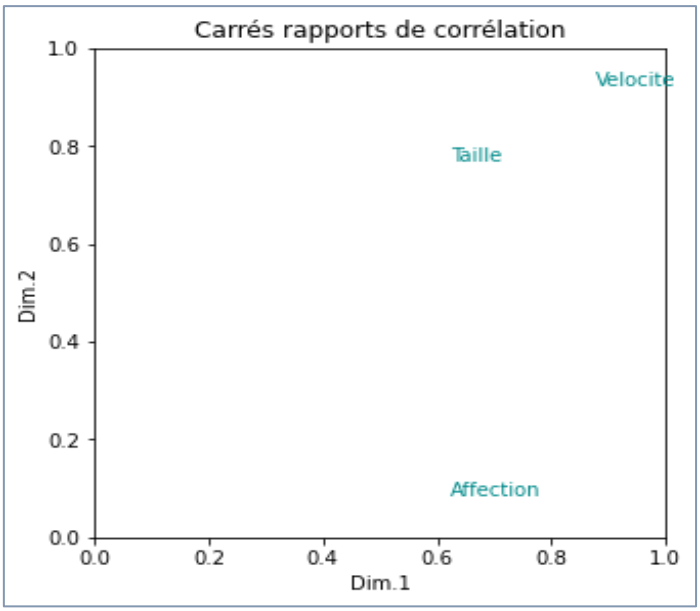


Figure 154 – Représentation des carrés des rapports de corrélation des variables – ACM – Données "Canidés"

5.3.4.6 Reconstitution des distances

A la distance du χ^2 dans l'espace originel est substitué la distance euclidienne dans le repère factoriel. La précision de l'approximation dépend à la fois du nombre de facteurs considérés, de leur qualité de restitution, et de la qualité de représentation des modalités (COS2). Si nous prenons en compte tous les (H_{max}) facteurs, la reconstitution est exacte.

Le même principe s'applique à la distance à l'origine des modalités. Rappelons d'ailleurs que le ratio entre les carrés des distances estimées et originelles constitue le COS2.

5.3.5 Analyse des individus

5.3.5.1 Coordonnées factorielles

Les coordonnées factorielles des individus (F_{ih} ; individu n°i sur le facteur n°h) sont faciles à obtenir avec « fanalysis ».

```
#coordonnées des individus dans le plan
print(pandas.DataFrame(acm.row_coord_[:,:2],index=X.index,columns=['Coord.F1','Coo
rd.F2']))
```

```
                Coord.F1  Coord.F2
Chien
Beauceron   4.924471e-15 -1.279924
Basset      1.150779e+00  0.799057
```

```
Berger All   4.763557e-15 -1.279924
Boxer       -1.150779e+00  0.385124
Bull-Dog     4.284137e-01  0.509675
Bull-Mastif  1.150779e+00 -0.028809
Caniche     -4.284137e-01  0.509675
Labrador    -1.150779e+00  0.385124
```

De nouveau, un petit graphique pour situer les positions relatives de nos canidés (Figure 155).

```
#représentation des individus dans le plan
fix,ax = plt.subplots(figsize=(7,7))
ax.axis([-2,+2,-2,+2])
ax.plot([-2,+2],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-2,+2],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Représentation des individus")

for i in range(X.shape[0]):
    ax.text(acm.row_coord_[i,0],acm.row_coord_[i,1],X.index[i],color='firebrick')

plt.show()
```

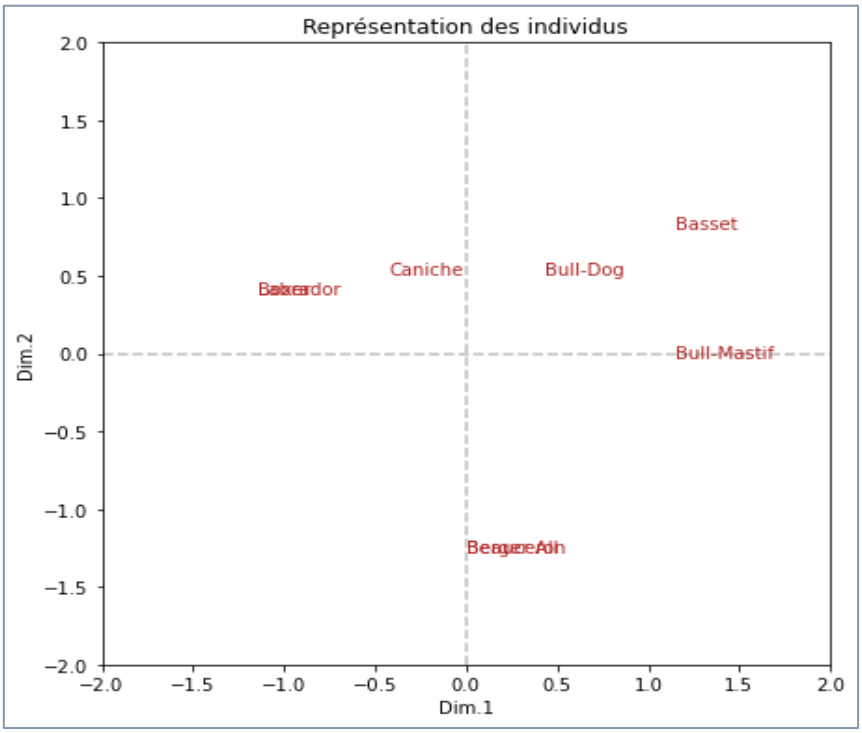


Figure 155 – Représentation des individus – ACM – Données "Canidés"

Nous observons deux paires d'observations superposées : (Boxer, Labrador) et (Beauceron, Berger All). Elles partagent respectivement les mêmes descriptions dans les données initiales. Il est normal qu'on ne puisse pas non plus les distinguer dans la représentation factorielle.

5.3.5.2 Contributions

Ça devient une habitude maintenant (cf. ACP), la contribution d'une observation n°i au facteur n°h correspond au carré de la coordonnées ramenée à la variance restituée (par le facteur) :

$$CTR_{ih} = \frac{1}{n} \times \frac{F_{ih}^2}{\lambda_h}$$

Nous pouvons afficher les valeurs :

```
#afficher les contributions
print(pandas.DataFrame(acm.row_contrib_[:,:2],index=X.index,columns=['Contrib.F1',
'Contrib.F2']))
```

```
              Contrib.F1  Contrib.F2
Chien
Beauceron   4.281311e-28   34.620365
Basset      2.337985e+01   13.493329
Berger All  4.006086e-28   34.620365
Boxer       2.337985e+01    3.134479
Bull-Dog    3.240293e+00    5.489722
Bull-Mastif 2.337985e+01    0.017539
Caniche     3.240293e+00    5.489722
Labrador    2.337985e+01    3.134479
```

Ou, comme pour les modalités, afficher graphiquement les contributions à un facteur particulier.

Pour le 1er par exemple :

```
#ou bien graphiquement
acm.plot_row_contrib(num_axis=1)
```

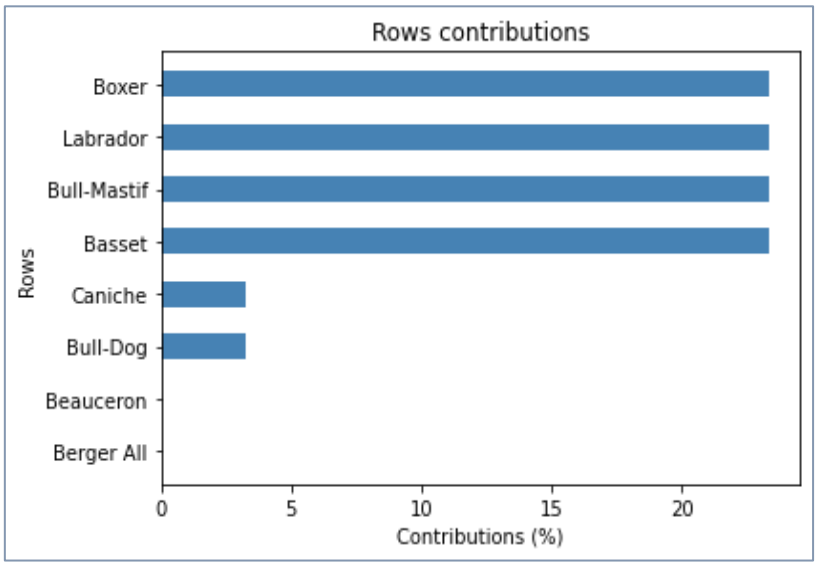


Figure 156 – Contributions des individus au 1er facteur – ACM – Données "Canidés"

Naturellement, les observations situés aux extrémités du facteur sont les plus contributives (Boxer, Labrador, Bull-Mastif, Basset). Le premier facteur repose principalement sur une opposition (Boxer, Labrador) vs. (Basset, Bull-Mastif).

Bis (et même plus) repetita. Les contributions des individus à un facteur s'additionnent. La somme est égale à 100% (si on raisonne en pourcentage) lorsque nous considérons l'ensemble des observations.

5.3.5.3 Cos2

Les COS2 traduit la qualité de représentation des individus sur les facteurs :

$$COS_{ih}^2 = \frac{F_{ih}^2}{d^2(i)}$$

```
#affichage des COS2
print(pandas.DataFrame(acm.row_cos2_[:,:2],index=X.index,columns=['Cos2.F1','Cos2.F2']))
```

```
                Cos2.F1   Cos2.F2
Chien
Beauceron    1.455025e-29  0.982924
Basset       6.272969e-01  0.302444
Berger All   1.361488e-29  0.982924
Boxer        7.945760e-01  0.088993
Bull-Dog     1.501677e-01  0.212538
Bull-Mastif  6.272969e-01  0.000393
Caniche      1.501677e-01  0.212538
Labrador     7.945760e-01  0.088993
```

Ils peuvent se cumuler sur les H premiers facteurs.

```
#affichage des COS2 cumulés
print(pandas.DataFrame(numpy.cumsum(acm.row_cos2_[:,:2],axis=1),index=X.index,colu
mns=['Cos2.F1','Cum(Cos2).F2']))
```

```
                Cos2.F1  Cum(Cos2).F2
Chien
Beauceron    1.455025e-29      0.982924
Basset       6.272969e-01      0.929741
Berger All   1.361488e-29      0.982924
Boxer        7.945760e-01      0.883569
Bull-Dog     1.501677e-01      0.362706
Bull-Mastif  6.272969e-01      0.627690
Caniche      1.501677e-01      0.362706
Labrador     7.945760e-01      0.883569
```

On note que seuls « Bull-Dog » et « Caniche » sont mal représentés sur le premier plan factoriel. Manifestement, ces individus présentent des spécificités qui ne sont pas captées par les 2 premiers facteurs.

5.3.5.4 Reconstitution des distances

Le principe de la reconstitution des distances dans le repère factoriel énoncé pour les modalités (section 5.3.4.6) est valable pour les observations.

5.3.6 Représentation simultanée – Relations de transition

Est-ce qu'on peut associer les caractéristiques (les modalités) aux individus dans le repère factoriel ? C'est tout l'enjeu de la représentation simultanée.

5.3.6.1 Relations de transition

A l'instar de l'AFC (section 4.5.2.54.3.3), ce n'est pas vraiment étonnant puisque les mécanismes internes de calcul sont les mêmes, il est possible d'obtenir les coordonnées des modalités (colonnes) à partir de celles des individus (lignes), et inversement.

Modalités → Individus. Pour les coordonnées des individus (F_{ih}) à partir des modalités (G_{kh}), la relation s'écrit :

$$F_{ih} = \frac{1}{\sqrt{\lambda_h}} \sum_{k=1}^{M} \frac{x_{ik}}{p} G_{kh}$$

On note que ($\frac{x_{ik}}{p}$) correspond au profil ligne de l'individu n°i.

Prenons un exemple pour les données « Canidés ». Voici le profil de l'observation « Basset ».

```
#profil du Basset
profil_Basset = X.loc["Basset"]/p
print(profil_Basset)
```

```
Taill-      0.333333
Taille+     0.000000
Taille++    0.000000
Velo-       0.333333
Veloc+      0.000000
Veloc++     0.000000
```

```
Affe-       0.333333
Affec+      0.000000
Name: Basset, dtype: float64
```

Rappelons les coordonnées des modalités sur le 1er facteur.

```
#rappel coordonnées factorielles des modalités - 1er facteur
print(acm.col_coord_[:,0])
```

```
[ 4.55873798e-01 -1.36762139e+00  4.55873798e-01  1.08146090e+00
 -1.08146090e+00  7.05894227e-15  1.36762139e+00 -4.55873798e-01]
```

Nous appliquons la formule.

```
#coordonnées calculée par la relation de transition du Basset
coord_Basset = (1/numpy.sqrt(acm.eig_[0][0]))*numpy.dot(profil_Basset,acm.col_coor
d_[:,0])
print(coord_Basset)
```

```
1.150779460602135
```

Et nous vérifions si nous matchons avec la coordonnée fournie nativement par l'ACM.

```
#vérification
print(pandas.DataFrame(acm.row_coord_[:,0],index=X.index).loc['Basset'])
```

```
0    1.150779
Name: Basset, dtype: float64
```

Yes !

Individus → Modalités. Le chemin inverse est tout aussi aisé. La relation de transition s'écrit :

$$G_{kh} = \frac{1}{\sqrt{\lambda_h}} \sum_{i=1}^{n} \frac{x_{ik}}{n_k} F_{ih}$$

Où $(\frac{x_{ik}}{n_k})$ représente le profil colonne pour la modalité « k ».

5.3.6.2 Représentation simultanée quasi-barycentrique

On parle de représentation quasi-barycentrique parce que la coordonnée de chaque point modalité correspond – au coefficient $(\frac{1}{\sqrt{\lambda_h}})$ près – à la moyenne des coordonnées des individus qui lui sont rattachés. Et, réciproquement, chaque point modalité est situé au barycentre des individus qui possède la caractéristique, toujours au coefficient $(\frac{1}{\sqrt{\lambda_h}})$ près.

Cette relation nous autorise à réaliser une représentation simultanée des points individus et modalités que l'on retrouve dans grand nombre de logiciels.

```
#représentation simultanée - relation quasi-barycentrique
fix,ax = plt.subplots(figsize=(7,7))
ax.axis([-2,+2,-2,+2])
ax.plot([-2,+2],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-2,+2],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Représentation simultanée - 1")

for i in range(X.shape[1]):
    ax.text(acm.col_coord_[i,0],acm.col_coord_[i,1],X.columns[i],color='dodgerblue')

for i in range(X.shape[0]):
    ax.text(acm.row_coord_[i,0],acm.row_coord_[i,1],X.index[i],color='firebrick')

plt.show()
```

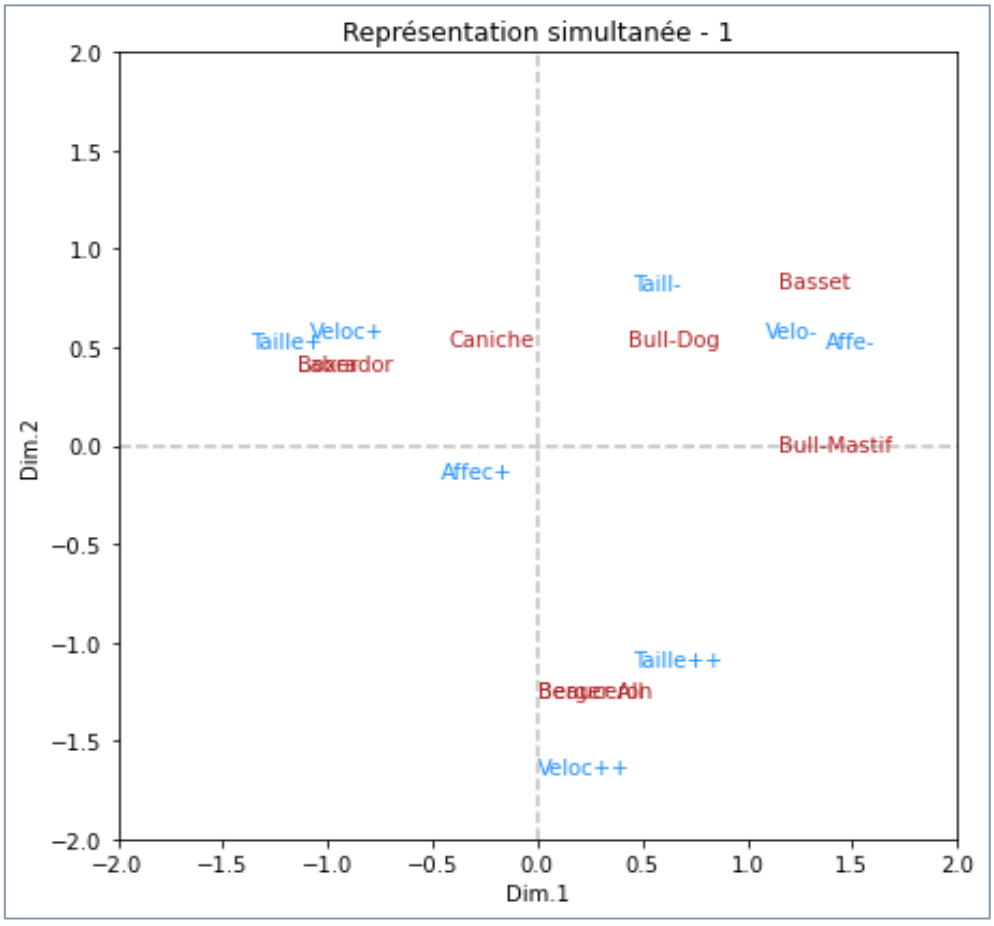


Figure 157 – Représentation simultanée 1 – ACM – Données "Canidés"

Toujours avec prudence lorsqu'il s'agit de transformer les proximités en associations, et en revenant à chaque fois aux données pour affermir les interprétations, nous remarquons que :

– « Beauceron » et « Berge All » partagent les caractéristiques (Taille++, Vélocité++).

Chien	Taille	Velocite	Affection
Beauceron	Taille++	Veloc++	Affec+
Berger All	Taille++	Veloc++	Affec+

– « Boxer » et « Labrador » partagent les modalités (Talille+, Veloc+).

Chien	Taille	Velocite	Affection
Boxer	Taille+	Veloc+	Affec+
Labrador	Taille+	Veloc+	Affec+

– « Basset » et « Bull-Mastif » partagent les caractéristiques (Vélocité-, Affection-)

Chien	Taille	Velocite	Affection
Basset	Taill-	Velo-	Affe-
Bull-Mastif	Taille++	Velo-	Affe-

5.3.6.3 Représentation simultanée barycentrique

Il existe une alternative pour faire figurer les individus et les modalités dans le même repère. Les individus sont placés normalement. Les modalités sont positionnées de manière à ce que leurs coordonnées correspondent réellement à la moyenne des coordonnées des individus qui leur sont rattachés (Saporta, 2006, page 225 ; Husson et al., 2009, page 138). Chaque modalité est située au barycentre des individus qui leur correspondent, d'où l'appellation « représentation barycentrique ».

On pourrait calculer explicitement les moyennes conditionnelles des modalités à partir des coordonnées factorielles. Mais au regard des relations de transition ci-dessus, il suffit en pratique de multiplier les coordonnées factorielles des points-modalités par la racine carrée de la valeur propre associée. Il s'agit donc d'une contraction des positions des modalités dans le repère puisque la valeur propre est nécessairement inférieure à 1 en ACM ($\lambda_h \leq 1, \forall h$).

Réalisons les calculs sous Python. Calculons tout d'abord les moyennes conditionnelles sur les deux premiers facteurs pour la variable « Taille ». Vérifions que nous obtenons les mêmes valeurs en corrigeant les coordonnées factorielles des points-modalités.

Nous créons un data frame intermédiaire pour associer la variable « Taille » aux facteurs (F1, F2). Nous utilisons ensuite un « pivot table » (les adeptes des tableaux croisés dynamiques sous Excel ne sont pas dépaysés) pour obtenir les moyennes conditionnelles.

```
#données temporaires liant Taille et Facteurs (1,2)
temp = pandas.DataFrame(acm.row_coord_[:,:2],columns=['F1','F2'],index=X.index)
temp['Taille'] = D.Taille
print(temp)
```

```
                     F1        F2    Taille
Chien
Beauceron    4.924471e-15 -1.279924  Taille++
Basset       1.150779e+00  0.799057    Taill-
Berger All   4.763557e-15 -1.279924  Taille++
Boxer       -1.150779e+00  0.385124   Taille+
Bull-Dog     4.284137e-01  0.509675    Taill-
Bull-Mastif  1.150779e+00 -0.028809  Taille++
Caniche     -4.284137e-01  0.509675    Taill-
Labrador    -1.150779e+00  0.385124   Taille+
```

```
#moyennes conditionnelles
print(pandas.pivot_table(temp,values=['F1','F2'],index='Taille',aggfunc='mean'))
```

```
               F1        F2
Taille
Taill-    0.383593  0.606136
Taille+  -1.150779  0.385124
Taille++  0.383593 -0.862886
```

Passons maintenant par la correction des coordonnées factorielles à l'aide des valeurs propres.

```
#coordonnées des modalités - relations barycentriques
coord_moda_2 = numpy.array([acm.col_coord_[:,0]*numpy.sqrt(acm.eig_[0][0]),acm.col_coord_[:,1]*numpy.sqrt(acm.eig_[0][1])])
print(pandas.DataFrame(numpy.transpose(coord_moda_2),index=X.columns,columns=['G_prim_1','G_prim_2']))
```

```
             G_prim_1  G_prim_2
Taill-    3.835932e-01  0.606136
Taille+  -1.150779e+00  0.385124
Taille++  3.835932e-01 -0.862886
Velo-     9.099909e-01  0.426641
Veloc+   -9.099909e-01  0.426641
Veloc++   5.939718e-15 -1.279924
Affe-     1.150779e+00  0.385124
Affec+   -3.835932e-01 -0.128375
```

Effectivement, les deux approches produisent les mêmes nouvelles coordonnées des modalités (pour « Taille », mais l'exercice est valable pour les autres variables).

Réalisons alors la nouvelle représentation graphique.

```
#représentation simultanée - relation barycentrique
fix,ax = plt.subplots(figsize=(7,7))
ax.axis([-2,+2,-2,+2])
```

```
ax.plot([-2,+2],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-2,+2],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Représentation simultanée - 2")

for i in range(X.shape[1]):
    ax.text(coord_moda_2[0,i],coord_moda_2[1,i],X.columns[i],color='dodgerblue')

for i in range(X.shape[0]):
    ax.text(acm.row_coord_[i,0],acm.row_coord_[i,1],X.index[i],color='firebrick')

plt.show()
```

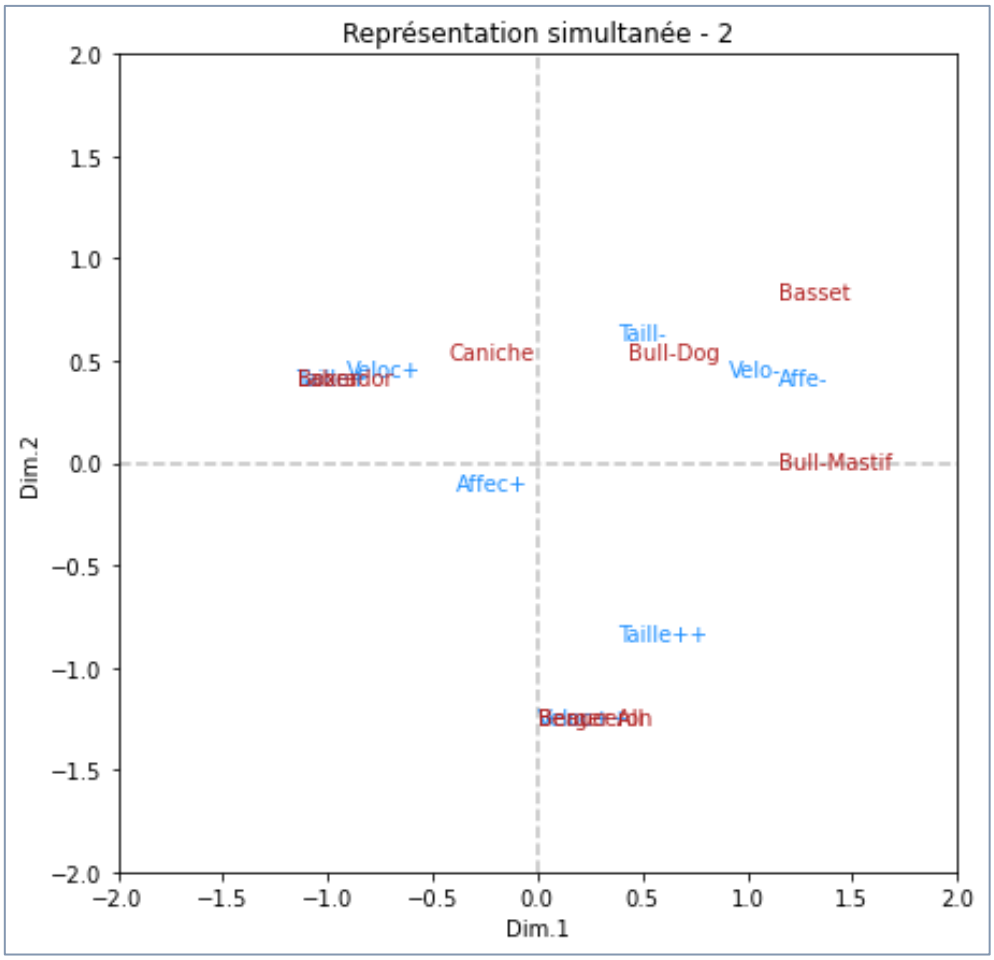


Figure 158 – Représentation simultanée 2 – ACM – Données "Canidés"

Le nouveau graphique (Figure 158) semble similaire au précédent (Figure 157), mais avec des différences qui méritent d'être soulignées. Par exemple :

- « Vélocité++ » a changé de place, il est maintenant situé exactement « sur » les individus « Beauceron » et « Berge All » parce que ce sont les deux seuls qui possèdent ces caractéristiques.

Chien	Taille	Velocite	Affection
Beauceron	Taille++	Veloc++	Affec+
Berger All	Taille++	Veloc++	Affec+

– Ce n'est pas le cas de « Taille++ » parce qu'un animal supplémentaire, « Bull-Mastif » possède cette propriété.

Chien	Taille	Velocite	Affection
Beauceron	Taille++	Veloc++	Affec+
Berger All	Taille++	Veloc++	Affec+
Bull-Mastif	Taille++	Velo-	Affe-

5.3.6.4 Formule de reconstitution du tableau disjonctif complet

Approcher le tableau initial des données à partir des coordonnées est une des fonctionnalités importantes des méthodes factorielles, avec une correspondance exacte lorsque nous utilisons l'ensemble des (H_{max}) facteurs. Dans le cas de l'analyse des correspondances multiples, c'est le tableau des indicatrices qui est reconstitué. Nous estimons la « propension » de l'individu n°i à posséder le caractère n°k. La formule de reconstitution pour les H premiers facteurs s'écrit :

$$\hat{x}_{ik} = \frac{n_k}{n} \times \left(1 + \sum_{h=1}^{H} \frac{F_{ih} \times G_{kh}}{\sqrt{\lambda_h}}\right)$$

La qualité de l'approximation dépend bien sûr du pouvoir de restitution des facteurs.

Attention, la valeur ($\hat{x}_{ik}$) peut être négative ou excéder 1. Elle ne peut pas être assimilée à l'estimation d'une probabilité. Une règle de décision possible pour la transformer en valeur binaire 0/1 (absence ou présence du caractère « k » pour l'individu « i ») serait de comparer ($\hat{x}_{ik}$) à la moyenne des valeurs dans le tableau des indicatrices initial, qui est obtenue facilement avec ($\frac{n \times p}{n \times M} = \frac{p}{M}$).

Rappelons quelques résultats de l'ACM sur les données « Canidés » que nous utilisons pour la reconstitution des données.

```
#rappel coordonnées factorielles des individus (Fih)
print(pandas.DataFrame(acm.row_coord_[:,:2],index=X.index))
```

```
                     0         1
Chien
Beauceron    4.924471e-15 -1.279924
Basset       1.150779e+00  0.799057
Berger All   4.763557e-15 -1.279924
Boxer       -1.150779e+00  0.385124
Bull-Dog     4.284137e-01  0.509675
Bull-Mastif  1.150779e+00 -0.028809
Caniche     -4.284137e-01  0.509675
Labrador    -1.150779e+00  0.385124
```

```
#fréquences des modalités (n_k/n)
freq_moda = acm.c_[0]/n
print(freq_moda)
```

```
[0.375 0.25  0.375 0.375 0.375 0.25  0.25  0.75 ]
```

```
#rappel des coordonnées des modalités (G_kh)
print(pandas.DataFrame(acm.col_coord_[:,:2],index=X.columns))
```

```
                     0         1
Taill-    4.558738e-01  0.788128
Taille+  -1.367621e+00  0.500758
Taille++  4.558738e-01 -1.121966
Velo-     1.081461e+00  0.554740
Veloc+   -1.081461e+00  0.554740
Veloc++   7.058942e-15 -1.664220
Affe-     1.367621e+00  0.500758
Affec+   -4.558738e-01 -0.166919
```

Nous affichons le tableau initial des indicatrices aux fins de vérification.

```
#tableau initial des indicatrices pour vérification
print(X)
```

```
            Taill-  Taille+  Taille++  Velo-  Veloc+  Veloc++  Affe-  Affec+
Chien
Beauceron        0        0         1      0       0        1      0       1
Basset           1        0         0      1       0        0      1       0
Berger All       0        0         1      0       0        1      0       1
Boxer            0        1         0      0       1        0      0       1
Bull-Dog         1        0         0      1       0        0      0       1
Bull-Mastif      0        0         1      1       0        0      1       0
Caniche          1        0         0      0       1        0      0       1
Labrador         0        1         0      0       1        0      0       1
```

Nous appliquons le calcul ci-dessus pour obtenir le tableau reconstitué.

```
#tableau reconstitué
temp = acm.row_coord_[:,:2]/numpy.sqrt(acm.eig_[0][:2])
temp = 1+numpy.dot(temp,numpy.transpose(acm.col_coord_[:,:2]))
Xhat = numpy.around(temp*freq_moda,2)
print(pandas.DataFrame(Xhat,index=X.index,columns=X.columns))
```

```
            Taill-  Taille+  Taille++  Velo-  Veloc+  Veloc++  Affe-  Affec+
Chien
Beauceron    -0.12     0.04      1.08   0.03    0.03     0.94   0.04    0.96
Basset        0.92    -0.09      0.17   1.15    0.04    -0.18   0.85    0.15
```

```
Berger All    -0.12     0.04      1.08   0.03    0.03     0.94   0.04    0.96
Boxer          0.29     0.78     -0.07  -0.08    1.03     0.04  -0.15    1.15
Bull-Dog       0.66     0.16      0.18   0.72    0.31    -0.03   0.51    0.49
Bull-Mastif    0.60    -0.22      0.62   0.92   -0.19     0.27   0.71    0.29
Caniche        0.48     0.51      0.01   0.31    0.72    -0.03   0.16    0.84
Labrador       0.29     0.78     -0.07  -0.08    1.03     0.04  -0.15    1.15
```

Pour (Beauceron, Taille-) par exemple :

$$\hat{x}_{Beauceron,Taille-} = 0.375 \times \left(1 + \frac{0.0 \times 0.456}{\sqrt{0.7080}} + \frac{(-1.280) \times 0.788}{\sqrt{0.5915}}\right) = -0.12$$

Nous pouvons convertir ce dernier tableau en estimation booléenne (0/1) en la comparant au seuil ($\frac{p}{M} = \frac{3}{8} = 0.375$).

```
#estimation booléenne à partir des 2 premiers facteurs
print(pandas.DataFrame((Xhat > (p/M)).astype(int),index=X.index,columns=X.columns))
```

```
             Taill-  Taille+  Taille++  Velo-  Veloc+  Veloc++  Affe-  Affec+
Chien
Beauceron         0        0         1      0       0        1      0       1
Basset            1        0         0      1       0        0      1       0
Berger All        0        0         1      0       0        1      0       1
Boxer             0        1         0      0       1        0      0       1
Bull-Dog          1        0         0      1       0        0      1       1
Bull-Mastif       1        0         1      1       0        0      1       0
Caniche           1        1         0      0       1        0      0       1
Labrador          0        1         0      0       1        0      0       1
```

L'approximation est plutôt de bonne qualité, nous n'observons que 3 mauvaises associations (valeurs en rouge) dans le tableau reconstitué : (Bull-Mastif, Taille-), (Caniche, Taille+), (Bull-Dog, Affection-).

5.3.7 Individus supplémentaires

Nous avions longuement développé les motivations du traitement des individus supplémentaires précédemment (ex. ACP, section 1.5). Dans cette section, nous cherchons à situer un animal supplémentaire, le « Lévrier », par rapport à l'existant. Voici sa description :

ID	Chien	Taille	Velocite	Affection
Supplém.	Levrier	Taille++	Veloc++	Affe-

La relation de transition permet d'obtenir les coordonnées factorielles des individus supplémentaires à partir de leur description, plus précisément de leur profil. Pour l'individu i* sur le facteur n°h,

$$F_{i^*h} = \frac{1}{\sqrt{\lambda_h}} \sum_{k=1}^{M} \frac{x_{i^*k}}{p} G_{kh}$$

Où (x_{i^*k}) est l'indicatrice associé à l'individu supplémentaire i* pour la modalité « k », et par conséquent $(\frac{x_{i^*k}}{p})$ son profil.

Le tableau de calcul peut être résumé de la manière suivante :

Val.Propres	0.7080	0.5915		
	Axe.1	**Axe.2**	**Desc.Levrier**	**Profil.Levrier**
Taill-	0.456	0.788	**0**	0.000
Taille+	-1.368	0.501	**0**	0.000
Taille++	0.456	-1.122	**1**	0.333
Veloc-	1.081	0.555	**0**	0.000
Veloc+	-1.081	0.555	**0**	0.000
Veloc++	0.000	-1.664	**1**	0.333
Affe-	1.368	0.501	**1**	0.333
Affec+	-0.456	-0.167	**0**	0.000
		Somme	3	

Pour sa coordonnée sur le premier axe, nous formons l'équation :

$$\frac{1}{\sqrt{0.708}}(0.000 \times 0.456 + 0.000 \times (-1.368) + 0.333 \times 0.456 + \cdots) = 0.7224$$

Détaillons les calculs sous Python et plaçons l'individu dans le premier plan factoriel. Nous chargeons tout d'abord la description du « Lévrier » et nous construisons son profil.

```
#chargement de l'individu supplémentaire
DSupp = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="ACM_CANINES_SUPP",index_col=0)
print(DSupp)

           Taille Velocite Affection
Chien
Levrier  Taille++  Veloc++     Affe-


#vecteur pour individu supplémentaire
XSupp = numpy.zeros(M)

#codage 0/1 de l'individu supp
for v in DSupp.loc['Levrier'].values:
    XSupp[numpy.where(X.columns==v)] = 1

#profil de l'individu supplémentaire
```

```
XSupp = XSupp / p
print(XSupp)
```

```
[0.         0.         0.33333333 0.         0.         0.33333333
 0.33333333 0.        ]
```

Voici pour rappel les coordonnées factorielles des modalités et les valeurs propres.

```
#rappel des coordonnées des modalités
print(pandas.DataFrame(acm.col_coord_[:,:2],index=X.columns))
```

```
                     0         1
Taill-    4.558738e-01  0.788128
Taille+  -1.367621e+00  0.500758
Taille++  4.558738e-01 -1.121966
Velo-     1.081461e+00  0.554740
Veloc+   -1.081461e+00  0.554740
Veloc++   7.058942e-15 -1.664220
Affe-     1.367621e+00  0.500758
Affec+   -4.558738e-01 -0.166919
```

```
#rappel des valeurs propres
print(acm.eig_[0][:2])
```

```
[0.70803126 0.59148936]
```

Il ne nous reste plus qu'à appliquer la formule pour obtenir les coordonnées de l'individu supplémentaire et le placer dans le repère factoriel.

```
#projection
coord_supp = numpy.dot(numpy.reshape(XSupp,(1,M)),acm.col_coord_[:,:2])
coord_supp = coord_supp / numpy.sqrt(acm.eig_[0][:2])
print(coord_supp)
```

```
[[ 0.72236576 -0.99054221]]
```

```
#positionnement dans le plan factoriel
fix,ax = plt.subplots(figsize=(7,7))
ax.axis([-2,+2,-2,+2])
ax.plot([-2,+2],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-2,+2],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Individu supplémentaire")


#individus actifs
for i in range(X.shape[0]):
    ax.text(acm.row_coord_[i,0],acm.row_coord_[i,1],X.index[i],color='firebrick')


#individu supplémentaire
ax.text(coord_supp[0][0],coord_supp[0][1],'Levrier',color='magenta',fontsize=12)

plt.show()
```

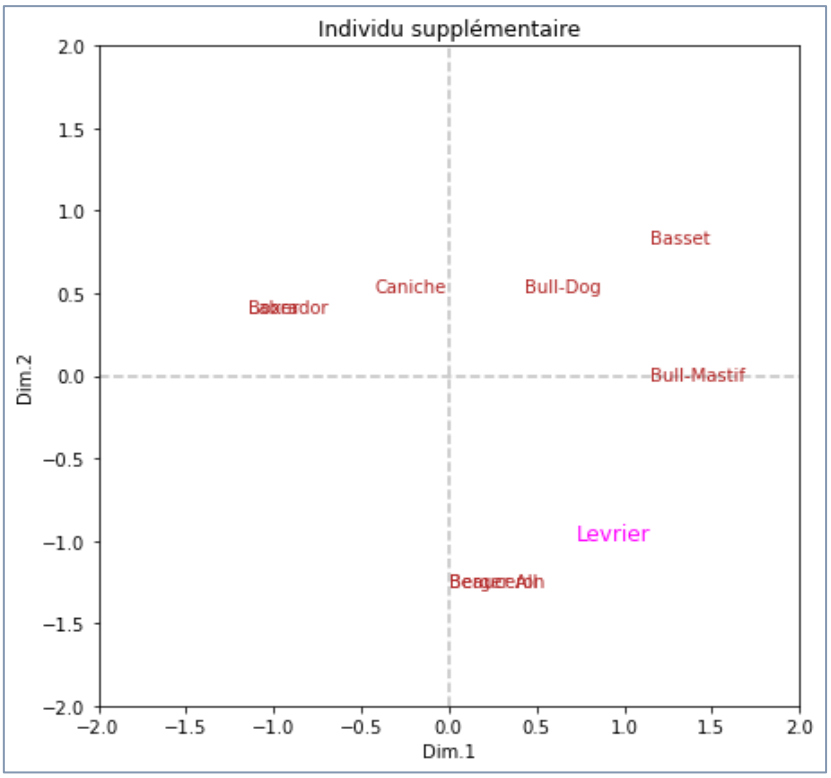


Figure 159 – Individu supplémentaire – ACM – Données "Canidés"

Le « Lévrier » est situé dans le voisinage du « Berger All » et « Beauceron » (Figure 160). Ce n'est pas étonnant, il partage avec eux les caractéristiques (Taille++, Vélocité++).

Chien	Taille	Velocite	Affection
Beauceron	Taille++	Veloc++	Affec+
Berger All	Taille++	Veloc++	Affec+

5.3.8 Variables illustratives

Le principe des variables illustratives est aussi valable pour l'analyse des correspondances multiples. Il s'agit de descripteurs, qualitatives ou quantitatives, non exploitées pour la construction des facteurs, mais que l'on utilise après coup pour mieux comprendre et commenter les résultats (cf. le principe pour l'ACP, section 1.6).

Dans le cas des données « Canidés », nous cherchons à approfondir les résultats de l'ACM à l'aide : (1) d'une cote d'amour subjective attribuée par des experts aux individus (totalement inventée pour la circonstance) (quantitative) ; (2) des fonctions qui leurs sont assignées (qualitative) (Figure 160).

Var. Illustrative quantitative

Var. Illustrative qualitative

ID	Chien	Cote	Fonction
1	Beauceron	2	utilite
2	Basset	4.5	chasse
3	Berger All	2.5	utilite
4	Boxer	3	compagnie
5	Bull-Dog	1.5	compagnie
6	Bull-Mastif	1	utilite
7	Caniche	4	compagnie
8	Labrador	3.5	chasse

Figure 160 – Variables illustratives – Données "Canidés"

5.3.8.1 Variables quantitatives

Le coefficient de corrélation est l'indicateur qui s'impose pour les variables illustratives quantitatives (section 1.6.1). Nous le calculons entre chaque facteur et les descripteurs additionnels. Il est possible, si on veut sophistiquer la présentation lorsque nous avons plusieurs variables, de les représenter dans une sorte de cercle des corrélations pour visualiser les liaisons.

Sous Python, nous affichons pour vérifications les variables illustratives. Nous faisons appel ensuite à la fonction corrcoef() de la librairie Numpy pour calculer la corrélation de « Cote » avec les 2 premiers facteurs :

```
#variables illustratives
print(D[['Cote','Fonction']])

             Cote   Fonction
Chien
Beauceron     2.0    utilite
Basset        4.5     chasse
Berger All    2.5    utilite
Boxer         3.0  compagnie
Bull-Dog      1.5  compagnie
Bull-Mastif   1.0    utilite
Caniche       4.0  compagnie
Labrador      3.5     chasse

#corrélation de Cote avec les 2 premiers facteurs
corr_cote = numpy.corrcoef(D.Cote,acm.row_coord_[:,:2],rowvar=False)[0,1:]
print(corr_cote)

[-0.28809902  0.44175308]

#positionnement dans un "cercle des corrélations"
fig, ax = plt.subplots(figsize=(5,5))
ax.axis([-1,+1,-1,+1])
```

```
ax.plot([-1,+1],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-1,+1],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Cercle des corrélations")

#variables illustratives
ax.text(corr_cote[0],corr_cote[1],'Cote',color='brown')
ax.arrow(0,0,corr_cote[0]+0.1,corr_cote[1]-0.1,color='r',head_width=0.05)

#cercle
from matplotlib.patches import Ellipse
ellipse = Ellipse((0,0),width=2,height=2,facecolor='none',edgecolor='silver',linestyle='--')
ax.add_patch(ellipse)

plt.show()
```

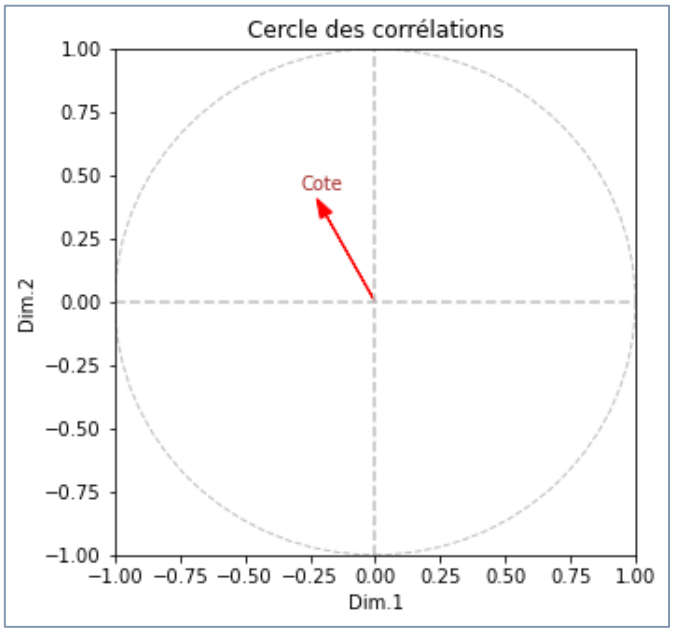


Figure 161 – Corrélations de la variable supplémentaire avec les facteurs (1, 2) – ACM – Données "Canidés"

La variable est plus corrélée avec le 2nd facteur qu'avec le 1er (en valeur absolue). Mais les liaisons restent faibles, il serait hasardeux d'en tirer des conclusions.

5.3.8.2 Variables qualitatives

L'objectif dans un premier temps est de pouvoir positionner les modalités des variables illustratives parmi les modalités actives, notamment pour identifier les proximités potentiellement intéressantes. Dans un second temps, pour apprécier l'éventuelle liaison entre la variable et le facteur étudié, il s'agit d'évaluer l'écartement des modalités sur les facteurs : soit pris globalement sous l'angle de la dispersion, on utilise alors le rapport de corrélation ; soit

pris individuellement par rapport à l'origine, on s'appuie alors sur la valeur-test. Pour ce second point, la démarche et la lecture des résultats ne diffèrent guère de ce que nous avions vu par ailleurs (voir ACP, section 1.6.2). Nous nous concentrons sur le premier objectif.

Calcul des coordonnées – Solution 1. Les moyennes conditionnelles constituent la voie simple pour obtenir les coordonnées des modalités. Attention néanmoins, si ($\bar{G}_{k^*h}$) est la moyenne des observations qui portent la modalité supplémentaire « k* » sur le facteur n°h, il faut la corriger par la valeur propre du facteur pour que la position soit raccord avec celles des modalités actives :

$$G_{k^*h} = \frac{\bar{G}_{k^*h}}{\sqrt{\lambda_h}}$$

Calcul des coordonnées – Solution 2. La seconde approche, plus classique, consiste à utiliser la relation de transition. Nous produisons les coordonnées factorielles de la modalité « k* » à partir de son profil et des coordonnées des individus (F_{ih}) (COURS ACM, page 44):

$$G_{k^*h} = \frac{1}{\sqrt{\lambda_h}} \sum_{i=1}^{n} \frac{x_{ik^*}}{n_{k^*}} F_{ih}$$

Intéressons-nous à la variable supplémentaire « Fonction » pour les données « Canidés ». Elle est composée de 3 modalités {chasse, compagnie, utilité}. Nous passons par la première solution pour calculer leurs coordonnées dans le 1er plan factoriel.

Pour ce faire, nous créons une structure temporaire associant les coordonnées factorielles des individus avec la variable « Fonction ». Nous calculons ensuite les moyennes conditionnelles.

```
#structure temporaire réunissant les coord. fact. (1,2) et fonction
df = pandas.DataFrame(acm.row_coord_[:,:2],index=X.index,columns=['F1','F2'])
df['Fonction'] = D.Fonction
print(df)
```

```
                      F1        F2   Fonction
Chien
Beauceron    4.924471e-15 -1.279924    utilite
Basset       1.150779e+00  0.799057     chasse
Berger All   4.763557e-15 -1.279924    utilite
Boxer       -1.150779e+00  0.385124  compagnie
Bull-Dog     4.284137e-01  0.509675  compagnie
Bull-Mastif  1.150779e+00 -0.028809    utilite
```

```
Caniche     -4.284137e-01  0.509675  compagnie
Labrador    -1.150779e+00  0.385124     chasse
```

```
#calcul des moyennes conditionnelles
GBar = pandas.pivot_table(df,values=['F1','F2'],index="Fonction",aggfunc='mean')
print(GBar)
```

```
                    F1        F2
Fonction
chasse    -2.109424e-15  0.592091
compagnie -3.835932e-01  0.468158
utilite    3.835932e-01 -0.862886
```

Nous dérivons les coordonnées à partir des moyennes en les divisant par la racine carrée des valeurs propres, conformément à la première formule.

```
#coordonnées factorielles des modalités - Solution 1
Gv1 = GBar.values/numpy.sqrt(acm.eig_[0][:2])
Gv1 = pandas.DataFrame(Gv1,index=GBar.index,columns=GBar.columns)
print(Gv1)
```

```
                    F1        F2
Fonction
chasse    -2.506903e-15  0.769866
compagnie -4.558738e-01  0.608722
utilite    4.558738e-01 -1.121966
```

Il ne reste plus qu'à les insérer dans le repère factoriel, parmi les modalités actives (Figure 162).

```
#représentation dans le plan
fix,ax = plt.subplots(figsize=(7,7))
ax.axis([-2,+2,-2,+2])
ax.plot([-2,+2],[0,0],color='silver',linestyle='--')
ax.plot([0,0],[-2,+2],color='silver',linestyle='--')
ax.set_xlabel("Dim.1")
ax.set_ylabel("Dim.2")
plt.title("Points-modalités supplémentaires")

#modalités actives
for i in range(X.shape[1]):
    ax.text(acm.col_coord_[i,0],acm.col_coord_[i,1],X.columns[i],color='dodgerblue')

#modalités illustratives de la variable "fonction"
for i in Gv1.index:
    ax.text(Gv1.loc[i][0],Gv1.loc[i][1],i,color='blue')

plt.show()
```

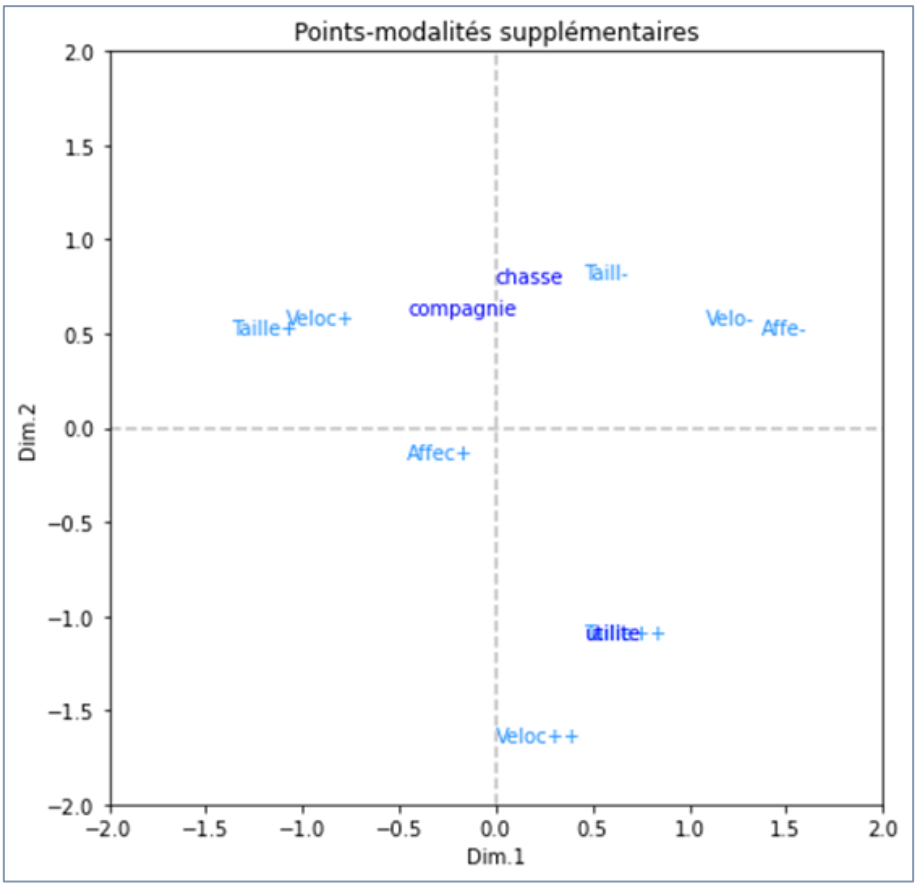


Figure 162 – Position des modalités de la variable illustrative "Fonction" – ACM – Données "Canidés"

Plusieurs commentaires :

- La variable « Fonction » est plus liée avec le 2nd facteur qui oppose (Vélocité++, Taille++) à (Taille-). Nous ne le réalisons pas ici, mais le calcul du rapport de corrélation devrait confirmer sans problème cette perception visuelle. Il semble donc que la fonction soit essentiellement tributaire du physique et de la rapidité des animaux.
- Est-ce que la position de « utilité » par rapport aux différentes modalités de « taille » et « vélocité » – surtout sur le 2nd axe factoriel – est évocateur d'une attraction avec « taille++ » d'une part (les points modalités sont superposés), et avec « vélocité++ » d'autre part ? Nous savons que les proximités peuvent être trompeuses dans les représentations graphiques. Pour nous en assurer, nous effectuons les croisements entre « fonction » et « taille », et entre « fonction » et « vélocité », en caractérisant le degré de liaison à travers le v de Cramer, et en faisant apparaître les contributions au χ^2 (Figure 163).

Row (Y)	Column (X)	Statistical indicator	
Fonction	Taille	Stat	Value
		d.f.	4
		Tschuprow's t	0.716860
		Cramer's v	0.716860
		Phi²	1.027778
		Chi² (p-value)	8.22 (0.0838)
		Lambda	0.600000
		Tau (p-value)	0.5556 (0.1001)
		U(R/C) (p-value)	0.6193 (0.0299)

Cross-tab

	Taille++	Taille-	Taille+	Sum
utilite	3 (+ 38 %)	0 (- 14 %)	0 (- 9 %)	3
chasse	0 (- 9 %)	1 (+ 1 %)	1 (+ 6 %)	2
compagnie	0 (- 14 %)	2 (+ 8 %)	1 (+ 1 %)	3
Sum	3	3	2	8 *100%*

Row (Y)	Column (X)	Statistical indicator	
Fonction	Velocite	Stat	Value
		d.f.	4
		Tschuprow's t	0.577350
		Cramer's v	0.577350
		Phi²	0.666667
		Chi² (p-value)	5.33 (0.2548)
		Lambda	0.400000
		Tau (p-value)	0.3651 (0.2761)
		U(R/C) (p-value)	0.3987 (0.1410)

Cross-tab

	Veloc++	Veloc-	Veloc+	Sum
utilite	2 (+ 39 %)	1 (- 0 %)	0 (- 21 %)	3
chasse	0 (- 9 %)	1 (+ 2 %)	1 (+ 2 %)	2
compagnie	0 (- 14 %)	1 (- 0 %)	2 (+ 13 %)	3
Sum	2	3	3	8 *100%*

Figure 163 – Lien entre « fonction » et « taille », et entre « fonction » et « vélocité » – Données « Canidés »

Nous avons une double réponse positive : (1) « fonction » est lié à « taille » et « vélocité » ; (2) ces relations reposent sur les attractions (utilité, taille++) et (utilité, vélocité++). Les positions relatives de la modalité « utilité » n'était pas trompeuse.

Remarque : Mais prudence encore une fois par rapport à ce type de lecture, nous noterons par exemple que la proximité (chasse, taille-) (Figure 162) ne signifie absolument rien quand nous inspectons le tableau de contingence. On ne saurait trop le répéter, il est vivement conseillé de revenir systématiquement aux données lorsque l'on souhaite tirer des conclusions à partir des représentations graphiques d'une analyse factorielle.

5.4 ACM avec d'autres outils (SAS, TANAGRA, R)

5.4.1 Données « Cars »

Dans cette section, nous reprenons les données issues de la documentation en ligne de SAS (PROC CORRESP). Les données proviennent d'un échantillon de ménages qui ont été invitées à

se décrire (Logement, Sexe, Revenu, Statut marital) et à décrire les caractéristiques de leurs véhicules (Origine, Taille, Type). Voici la liste des variables et modalités :

```
title1 'Automobile Owners and Auto Attributes';
proc format;
   value Origin  1 = 'American' 2 = 'Japanese' 3 = 'European';
   value Size    1 = 'Small'    2 = 'Medium'   3 = 'Large';
   value Type    1 = 'Family'   2 = 'Sporty'   3 = 'Work';
   value Home    1 = 'Own'      2 = 'Rent';
   value Sex     1 = 'Male'     2 = 'Female';
   value Income  1 = '1 Income' 2 = '2 Incomes';
   value Marital 1 = 'Single with Kids' 2 = 'Married with Kids'
                 3 = 'Single'           4 = 'Married';
run;
```

Le fichier comporte initialement 339 observations, dont 5 avec des valeurs manquantes. Avec les options par défaut, elles ont été omises lors de la mise en œuvre de l'analyse des correspondances multiples avec PROC CORRESP. Il a été décidé de supprimer ces lignes des données exportées présentées à TANAGRA et R dans les sections suivantes. Nous disposons ainsi de n = 334 observations et p = 7 variables actives. Voici les 10 premières lignes :

Origin	Size	Type	Home	Income	Marital	Sex
American	Large	Family	Own	1_Income	Married_With_Kids	Male
Japanese	Small	Sporty	Own	1_Income	Single	Male
Japanese	Small	Family	Own	2_Incomes	Married	Male
American	Large	Family	Rent	1_Income	Single	Male
American	Medium	Family	Own	2_Incomes	Married_With_Kids	Male
Japanese	Medium	Family	Own	2_Incomes	Single_With_Kids	Male
American	Large	Family	Own	2_Incomes	Married_With_Kids	Female
European	Medium	Family	Own	2_Incomes	Married_With_Kids	Female
American	Medium	Sporty	Own	2_Incomes	Married	Male
American	Medium	Family	Own	2_Incomes	Married	Female

Figure 164 – 10 premières lignes du fichier "Cars" – ACM

Remarque : Nous choisissons d'indiquer toutes les variables actives pour être raccord avec la documentation accessible en ligne, notamment pour pouvoir caler les résultats des différents outils. Mais considérant qu'un des objectifs possibles de l'étude est de mettre en relation les caractéristiques des ménages avec celles de leurs véhicules, nous aurions pu mettre en variables actives celles décrivant les ménages et utiliser en illustratives celles relatives à leurs véhicules.

5.4.2 ACM avec SAS – PROC CORRESP

La PROC CORRESP sait réaliser une analyse factorielle des correspondances simple ou multiple selon les options spécifiées. Dans notre cas, nous utilisons la commande suivante :

```
proc corresp mca benzecri data=Cars;
   tables Origin Size Type Income Home Marital Sex;
run;
```

L'option « mca » indique qu'une ACM a été demandée, « benzecri » pour introduire la correction éponyme (section 5.3.2), « tables » parce que SAS passe par le tableau de Burt dont on peut demander éventuellement l'affichage.

Remarque : PROC CORRESP se focalise sur l'analyse des points-modalités avec ces options. Il faut paramétrer différemment l'outil pour obtenir les informations (coordonnées factorielles, etc.) sur les individus.

5.4.2.1 Tableau des valeurs propres

Le premier tableau retrace les valeurs propres, uniquement celles supérieures à la moyenne puisque nous avons demandé la correction de Benzécri. Nous disposons des variances initiales et ajustées portées par les facteurs. Manifestement, les deux premiers suffisent largement pour décrire les données avec 94.52% d'inertie restituée (après correction).

Décomposition de l'inertie ajustée de Benzecri

Inertie principale	Inertie ajustée	Pourcentage	Pourcentage cumulé
0.32415	0.04474	75.52	75.52
0.23380	0.01126	19.00	94.52
0.18247	0.00214	3.60	98.12
0.16987	0.00099	1.68	99.80
0.15033	0.00008	0.13	99.93
0.14838	0.00004	0.07	100.00
	0.05924	100.00	

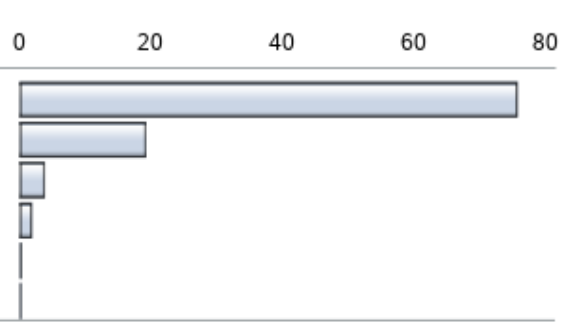


Figure 165 – Tableau des inerties – Proc Corresp – Données "Cars"

5.4.2.2 Analyse des modalités

Plusieurs tableaux décrivent les points-modalités. Nous les regroupons ci-dessous (Figure 166) :

$COS^2_{k(2)}$ ω_k I(k) G_{kh} CTR_{kh} COS^2_{kh}

Statistiques descriptives pour les points des colonnes			
	Qualité	Masse	Inertie
American	0.4925	0.0535	0.0521
European	0.0473	0.0188	0.0724
Japanese	0.3141	0.0706	0.0422
Large	0.4224	0.0180	0.0729
Medium	0.0548	0.0603	0.0482
Small	0.3825	0.0646	0.0457
Family	0.3330	0.0744	0.0399
Sporty	0.4112	0.0453	0.0569
Work	0.0052	0.0231	0.0699
1 Income	0.7991	0.0642	0.0459
2 Incomes	0.7991	0.0787	0.0374
Own	0.4208	0.1035	0.0230
Rent	0.4208	0.0393	0.0604
Married	0.3496	0.0432	0.0581
Married with Kids	0.3765	0.0466	0.0561
Single	0.6780	0.0466	0.0561
Single with Kids	0.0449	0.0064	0.0796
Female	0.1253	0.0637	0.0462
Male	0.1253	0.0791	0.0372

Coordonnées des colonnes		
	Dim1	Dim2
American	-0.4035	0.8129
European	-0.0568	-0.5552
Japanese	0.3208	-0.4678
Large	-0.6949	1.5666
Medium	-0.2562	0.0965
Small	0.4326	-0.5258
Family	-0.4201	0.3602
Sporty	0.6604	-0.6696
Work	0.0575	0.1539
1 Income	0.8251	0.5472
2 Incomes	-0.6727	-0.4461
Own	-0.3887	-0.0943
Rent	1.0225	0.2480
Married	-0.4169	-0.7954
Married with Kids	-0.8200	0.3237
Single	1.1461	0.2930
Single with Kids	0.4373	0.8736
Female	-0.3365	-0.2057
Male	0.2710	0.1656

Contributions partielles à l'inertie des points des colonnes		
	Dim1	Dim2
American	0.0268	0.1511
European	0.0002	0.0248
Japanese	0.0224	0.0660
Large	0.0268	0.1886
Medium	0.0122	0.0024
Small	0.0373	0.0764
Family	0.0405	0.0413
Sporty	0.0610	0.0870
Work	0.0002	0.0023
1 Income	0.1348	0.0822
2 Incomes	0.1099	0.0670
Own	0.0482	0.0039
Rent	0.1269	0.0103
Married	0.0232	0.1169
Married with Kids	0.0967	0.0209
Single	0.1889	0.0171
Single with Kids	0.0038	0.0209
Female	0.0223	0.0115
Male	0.0179	0.0093

Carrés du cosinus pour les points des colonnes		
	Dim1	Dim2
American	0.0974	0.3952
European	0.0005	0.0468
Japanese	0.1005	0.2136
Large	0.0695	0.3530
Medium	0.0480	0.0068
Small	0.1544	0.2281
Family	0.1919	0.1411
Sporty	0.2027	0.2085
Work	0.0006	0.0046
1 Income	0.5550	0.2441
2 Incomes	0.5550	0.2441
Own	0.3975	0.0234
Rent	0.3975	0.0234
Married	0.0753	0.2742
Married with Kids	0.3258	0.0508
Single	0.6364	0.0416
Single with Kids	0.0090	0.0359
Female	0.0912	0.0341
Male	0.0912	0.0341

Figure 166 – Informations sur les points-modalités – PROC CORRESP – Données "Cars"

Nous reconnaissons les différents éléments pour la plupart, mis à part « Qualité » qui, après investigations, s'avère correspondre à la somme des COS2 des modalités sur les facteurs sélectionnées, (H = 2) ici en l'occurrence.

5.4.2.3 Représentation graphique

La représentation des modalités dans le premier plan factoriel complète les sorties (Figure 167). Une analyse rapide semble montrer que :

- Sur le 1er facteur, qui restitue 75.52% de l'information après correction de Benzécri, les ménages composées de personnes mariées avec enfants (Marital = Married with kids), cumulant deux revenus (Income = 2_Incomes), plutôt propriétaires de leur logement (Home = Owen), font le choix de véhicules larges (Type = Large). Elles sont opposées aux célibataires (Marital = Single), ne bénéficiant forcément que d'un seul revenu (Income = 1_Income), louent leur logement (Home = Rent), lesquelles favorisent les véhicules sportifs (Type =

Sporty). Hé oui, quand on n'a pas la marmaille à transporter, les meubles à acheter chez Ikea, on peut se permettre de s'amuser en Fiat Coupé, trop facile.

- Le 2nd facteur (19%) semble opposer les personnes mariées (sans enfants) (Martial = Married) circulant au volant de petits véhicules (Size = Small) plutôt sportifs (Type = Sporty) à celles qui n'ont qu'un revenu (Income = 1_Income) et roulent avec des grosses (Size = Large) américaines (Origin = American), lesquelles sont principalement des hommes (Sex = Male) après investigations.

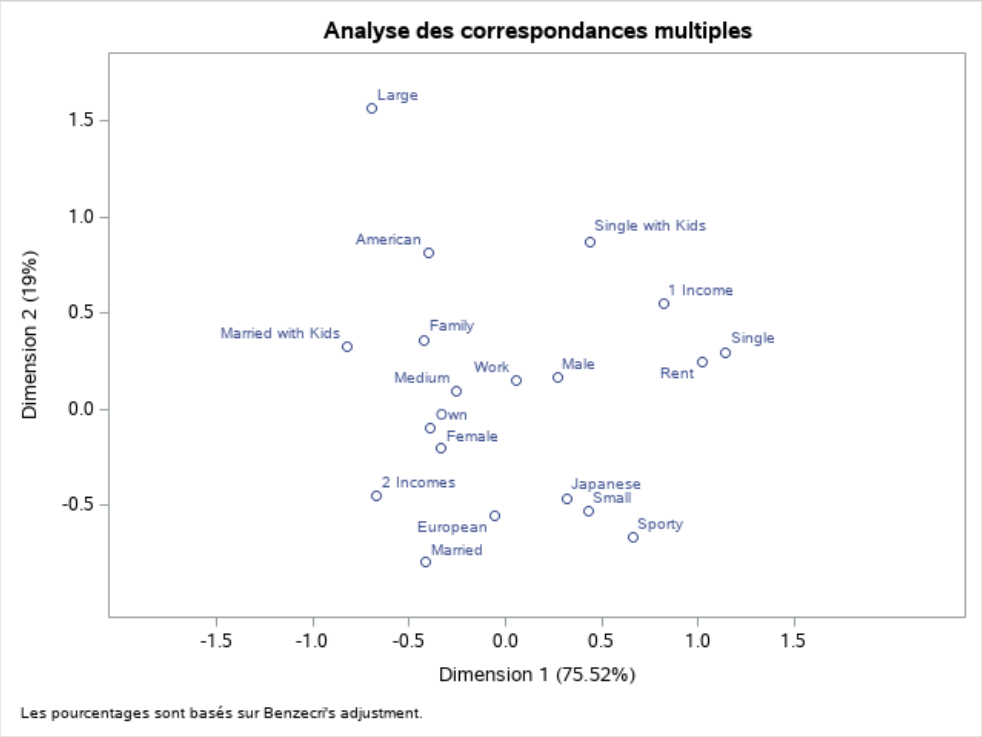


Figure 167 – Représentation des modalités dans le premier plan – Données "Cars"

5.4.3 ACM avec TANAGRA

Comme pour la majorité de ses outils d'analyse factorielle, TANAGRA a beaucoup bénéficié de la lecture attentive de la documentation de SAS, de PROC CORRESP en l'occurrence, pour l'implémentation de l'analyse des correspondances multiples. Le composant « Multiple Correspondence Analysis » (onglet « Factorial Analysis ») répond à la pratique usuelle de la méthode, avec la singularité de proposer deux formats de présentation des résultats.

L'importation des données et le paramétrage de l'outil sont décrits dans un tutoriel (TUTO 18). Pour les donnes « Cars » : nous limitons l'analyse à (H = 2) facteurs ; nous choisissons dans un

premier temps la présentation alternative des rapports qui incorpore l'ensemble des résultats dans un seul tableau, option pratique lorsque le nombre de facteurs reste réduit, nous trions les variables par facteur selon les contributions (TUTO 18, Figure 1).

5.4.3.1 Description du problème

TANAGRA regroupe les informations relatives aux données traitées (Figure 168).

Problem statement

# of instances	334
# of variables	7
# of variable values	19
Max # of factors	12
# of factors extracted	2
Total inertia	1.714286

Figure 168 – Description des données – TANAGRA – Données "Cars"

Nous disposons de (n = 334) observations, décrites par (p = 7) variables. Le nombre total de modalités est (M = 19). Le nombre total de facteur que l'on peut générer est (H_{max} = 12). Avec notre paramétrage, nous travaillons sur (H = 2) facteurs. L'inertie totale que nous décomposons sur les facteurs est ($I = 1.714286$).

5.4.3.2 Tableau des valeurs propres

Eigen values

	Original eigenvalues				Benzecri correction		
Axis	Eigenvalue	% explained	Histogram	% cumulated	Eigenvalue'	(%)	cumsum (%)
1	0.324153	18.91 %		18.91 %	0.044737	75.52 %	75.52 %
2	0.233796	13.64 %		32.55 %	0.011256	19.00 %	94.52 %
3	0.182466	10.64 %		43.19 %	0.002135	3.60 %	98.12 %
4	0.169871	9.91 %		53.10 %	0.000993	1.68 %	99.80 %
5	0.150334	8.77 %		61.87 %	0.000076	0.13 %	99.93 %
6	0.148375	8.66 %		70.52 %	0.000041	0.07 %	100.00 %
7	0.116052	6.77 %		77.29 %	-		
8	0.108785	6.35 %		83.64 %	-		
9	0.099330	5.79 %		89.43 %	-		
10	0.078786	4.60 %		94.03 %	-		
11	0.068197	3.98 %		98.01 %	-		
12	0.034139	1.99 %		100.00 %	-		

Figure 169 – Tableau des valeurs propres – ACM – TANAGRA – Données "Cars"

Le tableau des valeurs propres transcrit les inerties restituées par les axes (Figure 12). La correction de Benzécri d'applique aux facteurs portés par une variance supérieure à la moyenne. Comme sous SAS (Figure 165), nous constatons après coup que le premier plan restitue 94.52% de l'information disponible.

5.4.3.3 Analyse des variables et points modalités

Factors characterization (active variables)

Values	Overall			Factor 1				Factor 2			
Attribute = Value	Mass	Sq.Dist	Inertia	coord	v.test	cos2 (tot.)	ctr (%)	coord	v.test	cos2 (tot.)	ctr (%)
Marital = Married_With_Kids	0.0466	2.0642	0.0962	0.82003	10.415	0.326 (0.33)	9.67	-0.32375	-4.112	0.051 (0.38)	2.09
Marital = Single	0.0466	2.0642	0.0962	-1.14613	-14.557	0.636 (0.64)	18.89	-0.29305	-3.722	0.042 (0.68)	1.71
Marital = Married	0.0432	2.3069	0.0997	0.41688	5.009	0.075 (0.08)	2.32	0.79539	9.556	0.274 (0.35)	11.69
Marital = Single_With_Kids	0.0064	21.2667	0.1364	-0.43735	-1.731	0.009 (0.01)	0.38	-0.87357	-3.457	0.036 (0.04)	2.09
-	-	-	-	-	-	Tot.ctr.	31.26	-	-	Tot.ctr.	17.59
Income = 1_Income	0.0642	1.2267	0.0787	-0.82512	-13.595	0.555 (0.56)	13.48	-0.54719	-9.016	0.244 (0.80)	8.22
Income = 2_Incomes	0.0787	0.8152	0.0642	0.67265	13.595	0.555 (0.56)	10.99	0.44608	9.016	0.244 (0.80)	6.70
-	-	-	-	-	-	Tot.ctr.	24.46	-	-	Tot.ctr.	14.91
Home = Own	0.1035	0.3802	0.0393	0.38872	11.504	0.397 (0.40)	4.82	0.09427	2.790	0.023 (0.42)	0.39
Home = Rent	0.0393	2.6304	0.1035	-1.02249	-11.504	0.397 (0.40)	12.69	-0.24798	-2.790	0.023 (0.42)	1.03
-	-	-	-	-	-	Tot.ctr.	17.52	-	-	Tot.ctr.	1.43
Size = Large	0.0180	6.9524	0.1249	0.69489	4.809	0.069 (0.07)	2.68	-1.56656	-10.842	0.353 (0.42)	18.86
Size = Small	0.0646	1.2119	0.0783	-0.43255	-7.170	0.154 (0.15)	3.73	0.52583	8.716	0.228 (0.38)	7.64
Size = Medium	0.0603	1.3688	0.0825	0.25624	3.997	0.048 (0.05)	1.22	-0.09649	-1.505	0.007 (0.05)	0.24
-	-	-	-	-	-	Tot.ctr.	7.63	-	-	Tot.ctr.	26.73
Origin = American	0.0535	1.6720	0.0894	0.40347	5.694	0.097 (0.10)	2.68	-0.81286	-11.472	0.395 (0.49)	15.11
Origin = Japanese	0.0706	1.0242	0.0723	-0.32082	-5.785	0.100 (0.10)	2.24	0.46776	8.434	0.214 (0.31)	6.60
Origin = European	0.0188	6.5909	0.1240	0.05684	0.404	0.000 (0.00)	0.02	0.55518	3.946	0.047 (0.05)	2.48
-	-	-	-	-	-	Tot.ctr.	4.94	-	-	Tot.ctr.	24.20
Type = Family	0.0744	0.9195	0.0684	0.42012	7.995	0.192 (0.19)	4.05	-0.36017	-6.854	0.141 (0.33)	4.13
Type = Sporty	0.0453	2.1509	0.0975	-0.66036	-8.216	0.203 (0.20)	6.10	0.66962	8.332	0.208 (0.41)	8.70
Type = Work	0.0231	5.1852	0.1198	-0.05747	-0.461	0.001 (0.00)	0.02	-0.15391	-1.233	0.005 (0.01)	0.23
-	-	-	-	-	-	Tot.ctr.	10.18	-	-	Tot.ctr.	13.06
Sex = Male	0.0791	0.8054	0.0637	-0.27103	-5.511	0.091 (0.09)	1.79	-0.16565	-3.368	0.034 (0.13)	0.93
Sex = Female	0.0637	1.2416	0.0791	0.33651	5.511	0.091 (0.09)	2.23	0.20567	3.368	0.034 (0.13)	1.15
-	-	-	-	-	-	Tot.ctr.	4.02	-	-	Tot.ctr.	2.08

Figure 170 – Caractérisation des facteurs – Points-modalités – TANAGRA – Données "Cars"

Ce mode d'affichage regroupe dans un seul emplacement toutes les informations relatives aux variables-modalités actives (Figure 170). Il évite à avoir jongler avec plusieurs tableaux disséminées à différents endroits. Il est surtout pratique lorsque le nombre de facteurs est réduit. Sinon, il est possible de passer par une présentation en tableaux séparés (section 5.4.3.5) lorsque nous devons manipuler un nombre plus élevé de facteurs.

La partie « Overall » indique le rôle de chaque modalité dans l'analyse, avec le poids, la distance au carré à l'origine et l'inertie. Ensuite, nous disposons pour chaque facteur : de la coordonnée, de la valeur-test, du COS2 et de la contribution. Pour évaluer l'influence des variables, les contributions des modalités homologues sont additionnées. Nous constatons ainsi que la variable « Marital » est celle qui pèse le plus sur le premier facteur avec une contribution de 31.26%, suivie de « Income » avec 24.46% et « Home » avec 17.52%.

Le déchiffrage de la profusion d'informations dans les tableaux peut vite se révéler fastidieux. En plus du tri selon les contributions, TANAGRA surligne les coordonnées des modalités, avec une couleur différente selon le signe, lorsqu'elle répondent aux conditions suivantes : son COS2 est supérieur à $(1/H_{max})$, sa contribution est supérieure à son poids, sa valeur test est supérieure à un seuil que l'on peut spécifier en paramètre (valeur-test seuil = 3) par défaut.

5.4.3.4 Carte des modalités

La carte des modalités est accessible dans l'onglet « Chart » (Figure 171).



Figure 171 – Carte des modalités – ACM – TANAGRA – Données "Cars"

Nous pouvons sélectionner les facteurs à placer en abscisse et ordonnée. Il est possible de filtrer les modalités selon les CTR et les COS2 pour éviter l'enchevêtrement des points-modalités lorsqu'ils sont très nombreux.

5.4.3.5 Présentation alternative des points modalités

Lorsque le nombre de facteurs retenu est élevé, TANAGRA propose un affichage en tableaux séparés inspiré de SAS pour faciliter la lecture. Nous avons tout d'abord le tableau des coordonnées des points-modalités accompagnées des valeurs tests (Figure 172). Le tri selon les contributions reste de mise.

Coordinates and test-values

Values	Overall			Coordinate		Test-Value	
Attribute = Value	Mass	Sq.Dist	Inertia	coord_1	coord_2	v-test_1	v-test_2
Marital = Married_With_Kids	0.0466	2.0642	0.0962	0.82003	-0.32375	10.415	-4.112
Marital = Single	0.0466	2.0642	0.0962	-1.14613	-0.29305	-14.557	-3.722
Marital = Married	0.0432	2.3069	0.0997	0.41688	0.79539	5.009	9.556
Marital = Single_With_Kids	0.0064	21.2667	0.1364	-0.43735	-0.87357	-1.731	-3.457
Income = 1_Income	0.0642	1.2267	0.0787	-0.82512	-0.54719	-13.595	-9.016
Income = 2_Incomes	0.0787	0.8152	0.0642	0.67265	0.44608	13.595	9.016
Home = Own	0.1035	0.3802	0.0393	0.38872	0.09427	11.504	2.790
Home = Rent	0.0393	2.6304	0.1035	-1.02249	-0.24798	-11.504	-2.790
Size = Large	0.0180	6.9524	0.1249	0.69489	-1.56656	4.809	-10.842
Size = Small	0.0646	1.2119	0.0783	-0.43255	0.52583	-7.170	8.716
Size = Medium	0.0603	1.3688	0.0825	0.25624	-0.09649	3.997	-1.505
Origin = American	0.0535	1.6720	0.0894	0.40347	-0.81286	5.694	-11.472
Origin = Japanese	0.0706	1.0242	0.0723	-0.32082	0.46776	-5.785	8.434
Origin = European	0.0188	6.5909	0.1240	0.05684	0.55518	0.404	3.946
Type = Family	0.0744	0.9195	0.0684	0.42012	-0.36017	7.995	-6.854
Type = Sporty	0.0453	2.1509	0.0975	-0.66036	0.66962	-8.216	8.332
Type = Work	0.0231	5.1852	0.1198	-0.05747	-0.15391	-0.461	-1.233
Sex = Male	0.0791	0.8054	0.0637	-0.27103	-0.16565	-5.511	-3.368
Sex = Female	0.0637	1.2416	0.0791	0.33651	0.20567	5.511	3.368

Figure 172 – Coordonnées et valeurs-test – TANAGRA – Données "Cars"

Puis un second tableau regroupant les COS2 et les CTR (contributions) est affiché (Figure 173). Le tri des modalités est coordonné avec celui du tableau précédent.

Cos² and contributions

Values	Cos²		CTR (%)	
Attribute = Value	cos2_1 (tot.)	cos2_2 (tot.)	ctr_1	ctr_2
Marital = Married_With_Kids	0.326 (0.33)	0.051 (0.38)	9.671	2.090
Marital = Single	0.636 (0.64)	0.042 (0.68)	18.893	1.712
Marital = Married	0.075 (0.08)	0.274 (0.35)	2.316	11.690
Marital = Single_With_Kids	0.009 (0.01)	0.036 (0.04)	0.379	2.094
-	-	Tot. ctr.	31.259	17.586
Income = 1_Income	0.555 (0.56)	0.244 (0.80)	13.475	8.217
Income = 2_Incomes	0.555 (0.56)	0.244 (0.80)	10.985	6.698
-	-	Tot. ctr.	24.460	14.915
Home = Own	0.397 (0.40)	0.023 (0.42)	4.825	0.393
Home = Rent	0.397 (0.40)	0.023 (0.42)	12.691	1.035
-	-	Tot. ctr.	17.516	1.428
Size = Large	0.069 (0.07)	0.353 (0.42)	2.676	18.856
Size = Small	0.154 (0.15)	0.228 (0.38)	3.728	7.638
Size = Medium	0.048 (0.05)	0.007 (0.05)	1.222	0.240
-	-	Tot. ctr.	7.625	26.735
Origin = American	0.097 (0.10)	0.395 (0.49)	2.685	15.110
Origin = Japanese	0.100 (0.10)	0.214 (0.31)	2.241	6.605
Origin = European	0.000 (0.00)	0.047 (0.05)	0.019	2.481
-	-	Tot. ctr.	4.945	24.196
Type = Family	0.192 (0.19)	0.141 (0.33)	4.052	4.129
Type = Sporty	0.203 (0.20)	0.208 (0.41)	6.099	8.695
Type = Work	0.001 (0.00)	0.005 (0.01)	0.024	0.234
-	-	Tot. ctr.	10.175	13.059
Sex = Male	0.091 (0.09)	0.034 (0.13)	1.793	0.929
Sex = Female	0.091 (0.09)	0.034 (0.13)	2.226	1.153
-	-	Tot. ctr.	4.020	2.082

Figure 173 – COS2 et CTR – TANAGRA – Données "Cars"

Remarque : Notre exemple ne s'y prête pas mais, TANAGRA, tout comme SAS et R, sait appréhender les variables supplémentaires. Lorsqu'elles sont qualitatives, les modalités associées sont positionnées dans la représentation factorielle (TUTO 18), les COS2 et valeur-test sont fournies. Lorsqu'elle sont quantitatives, il calcule la corrélation et la statistique de Fisher pour expertiser la significativité du lien avec les facteurs (COURS ACM, page 43).

5.4.3.6 Coefficients des fonctions scores

Les coefficients des fonctions score sont retracés dans un dernier tableau (Figure 174). Ils permettent de déployer le système de représentation sur les individus à partir de leur description sous forme de vecteurs d'indicatrices.

Factor Score Coefficients

Applied to indicator matrix i.e. columns are dummy variables

Attribute = Value	Axis_1	Axis_2
Origin = American	0.1012374	-0.2401601
Origin = Japanese	-0.0804985	0.1381984
Origin = European	0.0142629	0.1640288
Size = Large	0.1743574	-0.4628382
Size = Small	-0.1085343	0.1553551
Size = Medium	0.0642955	-0.0285065
Type = Family	0.1054152	-0.1064110
Type = Sporty	-0.1656936	0.1978393
Type = Work	-0.0144209	-0.0454712
Home = Own	0.0975346	0.0278530
Home = Rent	-0.2565583	-0.0732655
Income = 1_Income	-0.2070357	-0.1616674
Income = 2_Incomes	0.1687791	0.1317941
Marital = Married_With_Kids	0.2057567	-0.0956517
Marital = Single	-0.2875802	-0.0865810
Marital = Married	0.1046021	0.2349981
Marital = Single_With_Kids	-0.1097374	-0.2580964
Sex = Male	-0.0680056	-0.0489408
Sex = Female	0.0844365	0.0607654

Figure 174 – Coefficients des fonctions scores – TANAGRA – Données "Cars"

Ce dispositif peut être mis en œuvre notamment pour le traitement des individus supplémentaires.

5.4.3.7 Représentation des individus

Concernant les individus actifs, TANAGRA produit automatiquement les coordonnées factorielles à la suite du composant. Puisque notre effectif ne permet pas de les distinguer individuellement, nous utilisons le composant « Scatterplot ». Si nous illustrons les points selon la variable « Martial » qui est la plus déterminante pour le 1er facteur, nous distinguons les personnes célibataires à gauche de l'origine, en couple à droite (Figure 175).

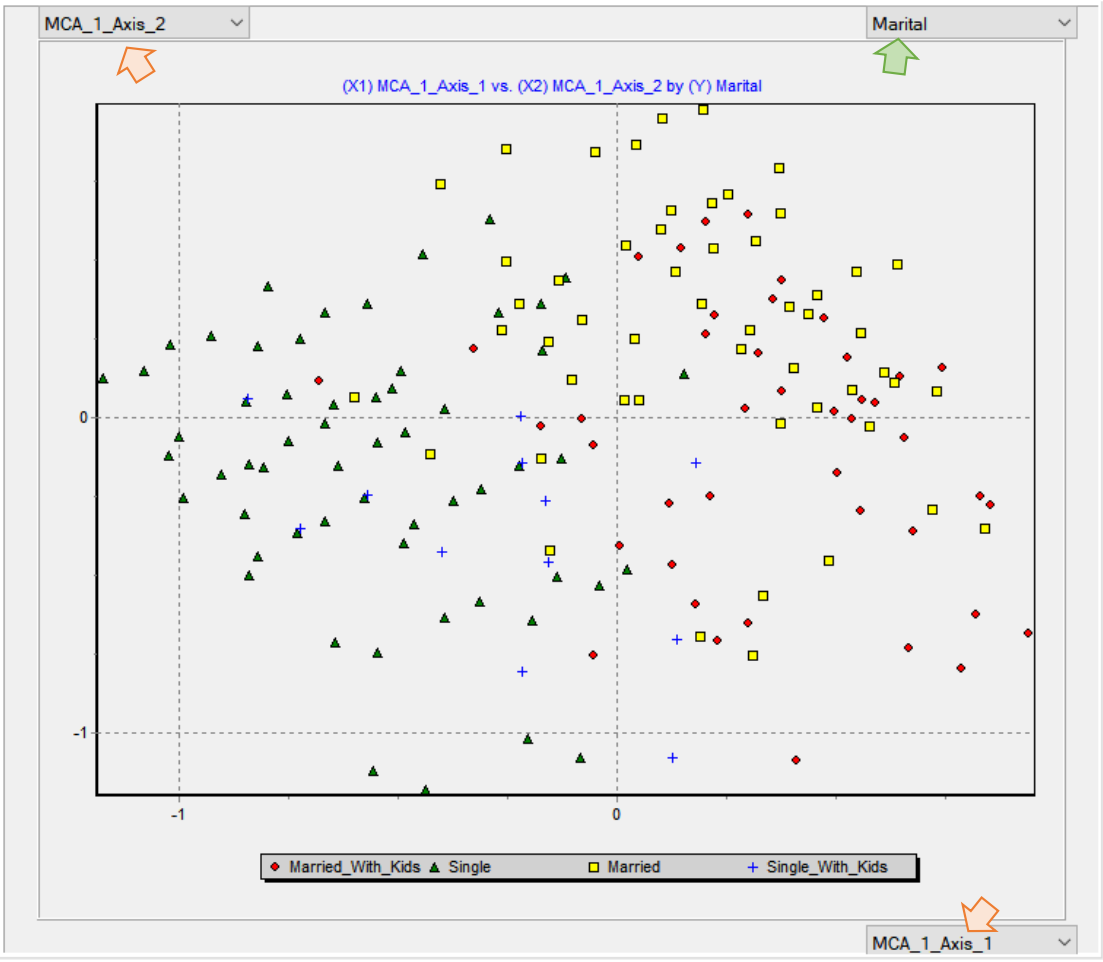


Figure 175 – Projection des individus dans (F1, F2) illustrés par "Marital" – TANAGRA – Données "Cars"

5.4.4 ACM avec R – Package « ca »

Après « factominer » pour l'ACP (section 1.7.4) et « ade4 » pour l'AFC (section 4.5.4), nous utilisons le package « ca » pour l'analyse des correspondances multiples, un troisième package phare de l'analyse factorielle sous R.

5.4.4.1 Importation des données

L'affaire commence toujours par le chargement des données. Elle ne présente aucune difficulté sous R via la librairie « openxlsx ».

```
#charger les données
library(openxlsx)
D <- read.xlsx("Data_Methodes_Factorielles.xlsx",sheet=10)
str(D)

## 'data.frame':    334 obs. of  7 variables:
##  $ Origin : chr  "American" "Japanese" "Japanese" "American" ...
```

```
##  $ Size   : chr  "Large" "Small" "Small" "Large" ...
##  $ Type   : chr  "Family" "Sporty" "Family" "Family" ...
##  $ Home   : chr  "Own" "Own" "Own" "Rent" ...
##  $ Income : chr  "1_Income" "1_Income" "2_Incomes" "1_Income" ...
##  $ Marital: chr  "Married_With_Kids" "Single" "Married" "Single" ...
##  $ Sex    : chr  "Male" "Male" "Male" "Male" ...
```

5.4.4.2 ACM sur tableau des indicatrices

Après avoir importé la librairie « ca », nous faisons appel à la fonction mjca(). Nous lui passons l'ensemble de données (D), nous demandons le calcul des informations pour les (nd = 2) premiers facteurs. Avec l'option « lambda = indicator », l'outil réalise l'ACM à partir d'une AFC sur la matrice des indicatrices (section 5.2.1). Nous affichons les résultats.

```
#importer la librairie
library(ca)

#lancer les calculs
#ACM travaille sur le tableau des indicatrices
acm <- mjca(D,nd=2,lambda='indicator')
print(acm)
```

```
##
##  Eigenvalues:
##            1        2        3        4        5        6        7
## Value      0.324153 0.233796 0.182466 0.169871 0.150334 0.148375 0.116052
## Percentage 18.91%   13.64%   10.64%   9.91%    8.77%    8.66%    6.77%
##            8        9       10       11       12
## Value      0.108785 0.09933 0.078786 0.068197 0.034139
## Percentage 6.35%    5.79%   4.6%     3.98%    1.99%
##
##
##  Columns:
##         Origin:American Origin:European Origin:Japanese Size:Large Size:Medium
## Mass           0.053464        0.018820        0.070573   0.017964    0.060308
## ChiDist        0.549437        0.994373        0.421438   1.108682    0.459400
## Inertia        0.016140        0.018608        0.012534   0.022081    0.012728
## Dim. 1        -0.708662       -0.099840        0.563489  -1.220502   -0.450069
## Dim. 2         1.681121       -1.148201       -0.967389   3.239867    0.199545
##         Size:Small Type:Family Type:Sporty Type:Work  Home:Own Home:Rent
## Mass      0.064585    0.074423    0.045338  0.023097  0.103507  0.039350
## ChiDist   0.467101    0.406058    0.630610  0.877304  0.276656  0.727726
## Inertia   0.014091    0.012271    0.018029  0.017777  0.007922  0.020839
## Dim. 1    0.759740   -0.737906    1.159855  0.100946 -0.682742  1.795908
## Dim. 2   -1.087486    0.744877   -1.384875  0.318299 -0.194971  0.512859
##         Income:1_Income Income:2_Incomes Marital:Married
## Mass           0.064157         0.078700        0.043199
## ChiDist        0.553991         0.451623        0.638147
## Inertia        0.019690         0.016052        0.017592
## Dim. 1         1.449250        -1.181454       -0.732215
## Dim. 2         1.131672        -0.922558       -1.644987
##         Marital:Married_With_Kids Marital:Single Marital:Single_With_Kids
## Mass                     0.046621       0.046621                 0.006416
## ChiDist                  0.638742       0.712317                 1.827178
## Inertia                  0.019021       0.023655                 0.021419
```

```
## Dim. 1                 -1.440297       2.013061                 0.768162
## Dim. 2                  0.669562       0.606067                 1.806674
##         Sex:Female Sex:Male
## Mass      0.063730 0.079127
## ChiDist   0.449453 0.361992
## Inertia   0.012874 0.010369
## Dim. 1   -0.591055 0.476039
## Dim. 2   -0.425358 0.342586
```

Nous disposons avec l'affichage par défaut :

- Des valeurs propres et des pourcentages d'inertie rapportées par les facteurs, tous deux non corrigés. Les (H = 2) premiers facteurs semblent restituer (18.91 + 13.64) 32.55% de l'information disponible. Nous savons que ce n'est pas vrai, qu'il faudra corriger cette estimation (section 5.4.4.3).
- Pour chaque modalité, les informations initiales (poids, distance au carré à l'origine, inertie) et les coordonnées factorielles dans le repère demandé (H = 2 dimensions).

5.4.4.3 Corrections de Benzécri et Greenacre

Essayons de calculer les valeurs propres de Benzécri (section 5.3.2) et les pourcentages d'inertie de Greenacre (section 5.3.3) à partir des informations fournies par l'objet. Nous démarrons à partir des valeurs singulières.

```
#les valeurs singulières
print(acm$sv)

##  [1] 0.5693446 0.4835250 0.4271610 0.4121545 0.3877288 0.3851954 0.3406646
##  [8] 0.3298263 0.3151665 0.2806883 0.2611455 0.1847662

#valeurs propres
lambada <- acm$sv^2
print(lambada)

##  [1] 0.32415331 0.23379638 0.18246649 0.16987130 0.15033363 0.14837550
##  [7] 0.11605236 0.10878538 0.09932990 0.07878592 0.06819698 0.03413856

#nombre de variables
p <- 7

#nombre de modalités
M <- 19

#valeurs propres corrigées de Benzécri
adjLambada <- ((p/(p-1))*(lambada[lambada>1/p]-1/p))^2
print(adjLambada)
```

```
## [1] 4.473741e-02 1.125631e-02 2.135448e-03 9.932906e-04 7.608312e-05
## [6] 4.144890e-05

#S'' pour correction de Greenacre
correction <- (p/(p-1))*(sum(lambada^2)-(M-p)/p^2)
print(correction)

## [1] 0.08026229

#pourcentage corrigés Greenacre
percent <- adjLambada/correction
print(percent)

## [1] 0.5573900828 0.1402441184 0.0266058679 0.0123755575 0.0009479310
## [6] 0.0005164181
```

Les (H = 2) premiers facteurs restitueraient en réalité (55.74 + 14.02) = 69.76% de l'information non redondante portée par les données. Vérifions nos opérations en refaisant appel à la fonction mjca() avec l'option « lambda = adjusted » qui permet justement de produire ces corrections.

```
#vérification avec l'option 'adjusted'
print(mjca(D,nd=2,lambda='adjusted'))

##
##  Eigenvalues:
##            1        2        3        4        5       6
## Value      0.044737 0.011256 0.002135 0.000993 7.6e-05 4.1e-05
## Percentage 55.74%   14.02%   2.66%    1.24%    0.09%   0.05%
##
##
##  Columns:
##         Origin:American Origin:European Origin:Japanese Size:Large Size:Medium
## Mass           0.053464        0.018820        0.070573   0.017964    0.060308
## ChiDist        0.549437        0.994373        0.421438   1.108682    0.459400
## Inertia        0.016140        0.018608        0.012534   0.022081    0.012728
## Dim. 1        -0.708662       -0.099840        0.563489  -1.220502   -0.450069
## Dim. 2         1.681121       -1.148201       -0.967389   3.239867    0.199545
##         Size:Small Type:Family Type:Sporty Type:Work  Home:Own Home:Rent
## Mass      0.064585    0.074423    0.045338  0.023097  0.103507  0.039350
## ChiDist   0.467101    0.406058    0.630610  0.877304  0.276656  0.727726
## Inertia   0.014091    0.012271    0.018029  0.017777  0.007922  0.020839
## Dim. 1    0.759740   -0.737906    1.159855  0.100946 -0.682742  1.795908
## Dim. 2   -1.087486    0.744877   -1.384875  0.318299 -0.194971  0.512859
##         Income:1_Income Income:2_Incomes Marital:Married
## Mass           0.064157         0.078700        0.043199
## ChiDist        0.553991         0.451623        0.638147
## Inertia        0.019690         0.016052        0.017592
## Dim. 1         1.449250        -1.181454       -0.732215
## Dim. 2         1.131672        -0.922558       -1.644987
##         Marital:Married_With_Kids Marital:Single Marital:Single_With_Kids
## Mass                     0.046621       0.046621                 0.006416
## ChiDist                  0.638742       0.712317                 1.827178
## Inertia                  0.019021       0.023655                 0.021419
## Dim. 1                  -1.440297       2.013061                 0.768162
## Dim. 2                   0.669562       0.606067                 1.806674
##         Sex:Female Sex:Male
## Mass      0.063730 0.079127
```

```
## ChiDist   0.449453 0.361992
## Inertia   0.012874 0.010369
## Dim. 1   -0.591055 0.476039
## Dim. 2   -0.425358 0.342586
```

Les résultats concordent. C'est toujours rassurant. Nous remarquons que la correction des valeurs propres et pourcentages d'inerties n'affectent en aucune manière les coordonnées factorielles des modalités.

5.4.4.4 Représentation des modalités

Par défaut, l'appel à plot() sur l'objet produit la carte des modalités (Figure 176).

```
#représentation des modalités
plot(acm)
```

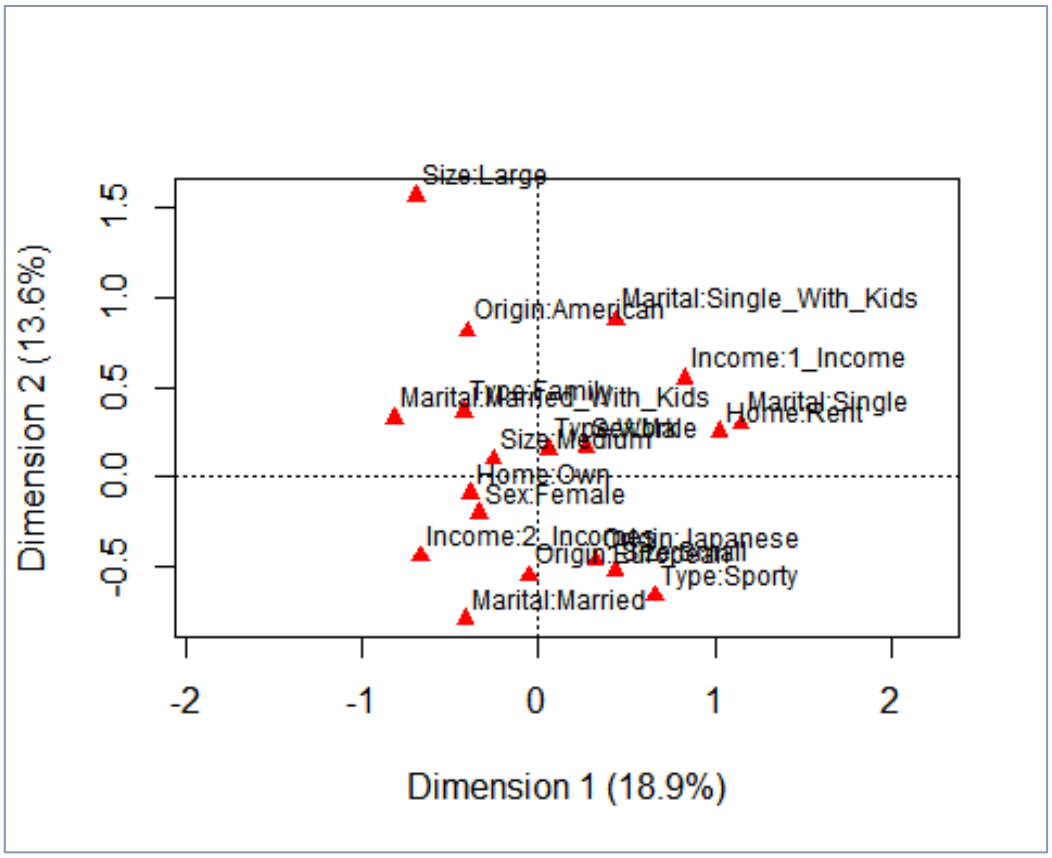


Figure 176 – Carte des modalités – ACM – R, package "ca" – Données "Cars"

5.4.4.5 Représentation des individus

Pour la carte des individus, nous exploitons le champ « $rowcoord » de l'objet « acm ». Comme sous TANAGRA (Figure 175), nous illustrons les points à l'aide de la variable « Marital »

```r
#codes couleurs
couleurs = c('deepskyblue','blue','yellowgreen','springgreen4')

#représentation des individus
plot(acm$rowcoord,col=couleurs[factor(D$Marital)],pch=19,xlab='Dim.1',ylab='Dim.2')
legend('bottomright',legend=levels(factor(D$Marital)),col=couleurs,pch=19)
```

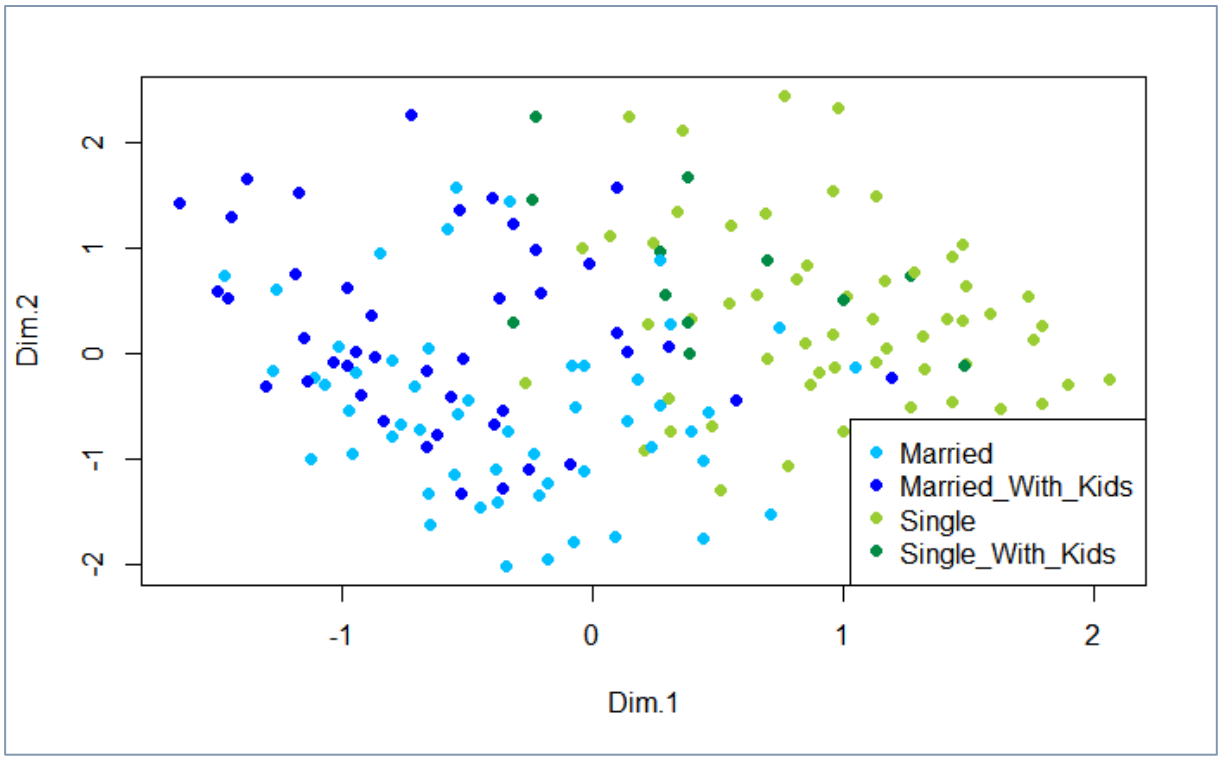


Figure 177 – Carte des individus – ACM – R, package "ca" – Données "Cars"

6 Analyse factorielle des données mixtes (AFDM)

On utilise habituellement l'analyse en composantes principales (ACP) lorsque toutes les variables actives sont quantitatives, l'analyse des correspondances multiples (ACM) lorsqu'elles sont toutes catégorielles. Mais que faire lorsque nous avons un mix des deux types de variables ?

Une première piste consiste à mettre en actives les quantitatives, utiliser une ACP, puis introduire en illustratives les qualitatives. Ou a contrario, mettre respectivement en actives et illustratives les qualitatives et quantitatives en passant par une ACM. Très facile à mettre en œuvre dans les logiciels, ces solutions ne sont pas adaptées parce qu'une seule partie seulement de l'information participe à la construction des facteurs. Les résultats sont faussés.

Une seconde stratégie consiste à découper en classes les variables quantitatives et de passer par une ACM. Elle est néanmoins peu avantageuse lorsque (1) le nombre d'observations est faible, en effet l'ACM donne des résultats peu stables dans ce cas ; (2) le nombre de variables catégorielles est faible par rapport aux variables quantitatives. De toute manière, le découpage en classes n'est jamais anodin. Il faut définir le nombre d'intervalles pour chaque variable à traiter, choisir un algorithme pour obtenir les bornes de discrétisation. Les approches usuelles (intervalles de largeur ou de fréquences égales..., voir « La discrétisation des variables quantitatives », octobre 2014) ne sont pas toujours les plus adaptées. Le processus est susceptible de dénaturer l'information traitée.

Une troisième approche consiste à remplacer les variables catégorielles par une série d'indicatrices via un codage disjonctif complet. L'information n'est pas dénaturée car il y a une

bijection exacte entre les variables initiales et les variables recodées. Mais il n'est pas très judicieux de traiter directement avec une ACP des données mélangeant des colonnes de valeurs numériques (continues) et des indicatrices (0/1). Les dispersions étant différentes, il y a déséquilibre dans l'influence des variables, les résultats peuvent être biaisés.

L'analyse factorielle des données mixtes (AFDM) de Jérôme Pagès (« Analyse factorielle des données mixtes », Revue de Statistique Appliquée, Tome 52, N°4, 2004 ; pages 93-111) s'inscrit dans cette dernière piste. Mais elle introduit une subtilité supplémentaire. A l'instar de l'ACP normée où l'on réduit les variables (c'est une forme de recodage) pour uniformiser leurs influences, il propose de substituer au codage 0/1 des variables qualitatives un codage 0/x où « x » est savamment calculé à partir des fréquences des modalités. On peut dès lors utiliser un programme usuel d'ACP pour mener l'analyse. Les calculs sont bien maîtrisés. L'interprétation des résultats requiert en revanche un effort supplémentaire puisqu'elle sera différente selon que l'on étudie le rôle d'une variable active quantitative ou qualitative.

L'AFDM est associé au package « factominer » pour R. Mais en creusant un peu plus le domaine, je me suis rendu compte que l'approche, sous différentes appellations, était présente dans plusieurs packages pour R (ade4, pcamixdata), lesquels font état d'autres références bibliographiques. La comparaison des implémentations et des sorties sur plusieurs jeux de données montrent qu'ils (ces packages) produisent des résultats identiques. Le fondement et l'expression des techniques sont donc bien les mêmes.

Enfin, l'AFDM est une véritable généralisation aux données mixtes dans le sens où nous retrouvons les résultats de l'ACP normée lorsque toutes les variables sont quantitatives, ceux de l'ACM lorsqu'elles sont qualitatives.

6.1 Principe de l'analyse factorielle des données mixtes

6.1.1 Tableau de données

L'analyse factorielle des données mixtes traite les tableaux individus-variables, lesquelles sont composées d'un mix de quantitatives et qualitatives. Nous utilisons un extrait des données « Autos 2005 » accessible sur la page de cours de Pierre-Louis Gonzalez (STA101) au CNAM.

L'organisation des données étant spécifique, nous inaugurons de nouvelles notations dans ce chapitre. Nous disposons de (n = 10) observations, (C = 5) variables quantitatives, (D = 3) qualitatives.

Variables « actives » quantitatives et /ou qualitatives

C = 5 quantitatives D = 3 qualitatives

$i : 1, \ldots, n = 10$

Individus actifs

Modele	puissanc	longueu	hauteur	poids	CO2	origine	carburant	4X4
GOLF	75	421	149	1217	143	Europe	Diesel	non
CITRONC4	138	426	146	1381	142	France	Diesel	non
P607	204	491	145	1723	223	France	Diesel	non
VELSATIS	150	486	158	1735	188	France	Diesel	non
CITRONC2	61	367	147	932	141	France	Essence	non
CHRYS300	340	502	148	1835	291	Autres	Essence	non
AUDIA3	102	421	143	1205	168	Europe	Essence	non
OUTLAND	202	455	167	1595	237	Autres	Diesel	oui
PTCRUISER	223	429	154	1595	235	Autres	Essence	non
SANTA_FE	125	450	173	1757	197	Autres	Diesel	oui

Figure 178 - Données "Autos 2005"

Encore une fois, nous nous posons les questions usuelles de l'analyse factorielle :

- Quels sont les véhicules qui se ressemble c.-à-d. qui présentent des caractéristiques similaires ? Il sera de nouveau question de proximités entre les individus.
- 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.

- Quelles sont les relations entre les variables ? Entre quantitatives, l'idée de la corrélation s'impose ; entre qualitatives, le χ^2 du tableau de contingence ; mais comment faire entre quantitatives et qualitatives ?
- La question peut être étendue aux modalités des variables qualitatives.

De nouveau avec l'analyse factorielle, il faut définir un espace de dimension réduite qui préserve au mieux l'information du tableau de données initial.

6.1.2 Objectif de l'analyse factorielle des données mixtes

Schématisons la démarche de l'analyse factorielle pour comprendre la nature des calculs qui seront proposés par la suite. Rappelons qu'elle a pour objectif de produire un premier facteur F_1 qui soit le plus lié possible avec les variables originelles. Si la liaison n'est pas parfaite, un second facteur est produit. Elle explique l'information résiduelle, non prise en compte sur le précédent (les précédents lorsque nous traitons le $h^{ème}$ facteur [h > 2]). Souvent les H premiers facteurs – H est faible par rapport au nombre de variables, mais il est à déterminer pour chaque base à traiter – suffisent pour obtenir un résumé satisfaisant des données.

ACP. Avec l'analyse en composantes principales lorsque les variables sont toutes quantitatives, la variance restituée par le premier facteur s'écrit :

$$\lambda_1 = \sum_j r^2(F_1, X_j)$$

Où $r^2(\,)$ est le carré du coefficient de corrélation linéaire entre la variable X_j et le facteur F_1.

ACM. Avec l'analyse des correspondances multiples, les variables sont toutes qualitatives, la variance restituée s'écrit :

$$\lambda_1 = \frac{1}{p} \sum_j \eta^2(F_1, X_j)$$

Où $\eta^2(\,\,)$ est le carré du rapport de corrélation entre la variable X_j et le facteur F_1 c.-à-d. on souhaite que, globalement, les modalités des variables soient le plus dispersés possible.

AFDM. La généralisation pour l'analyse factorielle des données mixtes consiste à définir un critère qui réunit ces spécifications pour appréhender simultanément les deux types de variables

$$\lambda_1 = \sum_{j=1}^{C} r^2(F_1, X_j) + \sum_{j=C+1}^{C+D} \eta^2(F_1, X_j)$$

Puisque $(0 \leq r^2 \leq 1)$ et $(0 \leq \eta^2 \leq 1)$, les deux types de variables pèsent de manière équilibrée dans l'analyse, cet aspect est primordial pour la praticabilité de l'approche.

6.2 Organisation des calculs

Pour parvenir à ce résultats, l'AFDM passe par l'application d'une analyse en composante sur des données transformées judicieusement. Deux étapes sont nécessaires.

6.2.1 Transformation des données

Codage disjonctif complet des variables qualitatives. Les variables quantitatives sont laissées en l'état, les qualitatives sont codées en indicatrices 0/1. Voici les notations adoptées après ces modifications en considérant que les quantitatives sont préalablement regroupées dans les premières colonnes.

$$k : 1, \ldots, C + M = P = 12$$

$$\sum_{j=C+1}^{C+D} m_j = M = 7$$

$$j : 1, \ldots, C = 5 \qquad m_1 = 3 \qquad m_2 = 2 \qquad m_3 = 2$$

Modele	puissance	longueur	hauteur	poids	CO2		orig_Autres	orig_Europ	orig_Franc	carb_Diese	carb_Essen	4X4_non	4X4_oui
GOLF	75	421	149	1217	143		0	1	0	1	0	1	0
CITRONC4	138	426	146	1381	142		0	0	1	1	0	1	0
P607	204	491	145	1723	223		0	0	1	1	0	1	0
VELSATIS	150	486	158	1735	188		0	0	1	1	0	1	0
CITRONC2	61	367	147	932	141	x_{ik}	0	0	1	0	1	1	0
CHRYS300	340	502	148	1835	291		1	0	0	0	1	1	0
AUDIA3	102	421	143	1205	168		0	1	0	0	1	1	0
OUTLAND	202	455	167	1595	237		1	0	0	1	0	0	1
PTCRUISER	223	429	154	1595	235		1	0	0	0	1	1	0
SANTA_FE	125	450	173	1757	197		1	0	0	1	0	0	1
Moyenne	162	444.8	153	1497.5	196.5	**N_k**	4	2	4	6	4	8	2
Ecart-Type	78.9	38.8	9.5	283.7	47.5	**p_k**	0.4	0.2	0.4	0.6	0.4	0.8	0.2

Figure 179 – AFDM – Transformation 1 des données "Autos 2005"

Nous disposons d'une matrice (x_{ik}) où i = 1, ..., n et k = 1, ..., P. Selon la nature de la colonne, nous calculons :

- Pour les variables quantitatives, les moyennes (μ_k) et écarts-type (σ_k).
- Pour les modalités, les fréquences absolues (n_k) et relatives $(p_k = \frac{n_k}{n})$.

Une seconde manipulation est opérée.

Standardisation différenciée des colonnes. Elle consiste à standardiser les colonnes, mais avec des calculs différents selon leur nature.

Modele	puissance	longueur	hauteur	poids	CO2	orig_Autres	orig_Europe	orig_France	carb_Diesel	carb_Essenc	4X4_non	4X4_oui
GOLF	-1.103	-0.614	-0.419	-0.989	-1.127	0.000	2.236	0.000	1.291	0.000	1.118	0.000
CITRONC4	-0.304	-0.485	-0.733	-0.411	-1.148	0.000	0.000	1.581	1.291	0.000	1.118	0.000
P607	0.532	1.192	-0.838	0.795	0.558	0.000	0.000	1.581	1.291	0.000	1.118	0.000
VELSATIS	-0.152	1.063	0.524	0.837	-0.179	0.000	0.000	1.581	1.291	0.000	1.118	0.000
CITRONC2	-1.280	-2.007	-0.628	-1.993	-1.169	0.000	0.000	1.581	0.000	1.581	1.118	0.000
CHRYS300	2.256	1.476	-0.524	1.189	1.990	1.581	0.000	0.000	0.000	1.581	1.118	0.000
AUDIA3	-0.761	-0.614	-1.047	-1.031	-0.600	0.000	2.236	0.000	0.000	1.581	1.118	0.000
OUTLAND	0.507	0.263	1.466	0.344	0.853	1.581	0.000	0.000	1.291	0.000	0.000	2.236
PTCRUISER	0.773	-0.408	0.105	0.344	0.811	1.581	0.000	0.000	0.000	1.581	1.118	0.000
SANTA_FE	-0.469	0.134	2.094	0.915	0.011	1.581	0.000	0.000	1.291	0.000	0.000	2.236

$$z_{ik} = \frac{x_{ik} - \mu_k}{\sigma_k} \; ; k = 1, \ldots, C \qquad z_{ik} = \frac{x_{ik}}{\sqrt{p_k}} \; ; k = C+1, \ldots, P$$

Figure 180 - AFDM - Transformation 2 - Données "Autos 2005"

Ainsi :

- Les variables quantitatives sont centrées et réduites.
- Les indicatrices des modalités sont normalisées par la racine carrée de la proportion.

Ce tableau de données sera présenté à l'algorithme d'analyse en composantes principales pour obtenir les résultats de l'analyse factorielle des données mixtes (section 6.2.3).

6.2.2 Equivalences avec l'ACP et l'ACM

L'analyse factorielle des données mixtes est une vraie généralisation dans le sens où : lorsque les variables sont toutes quantitatives, nous disposons des résultats d'une ACP normée ; lorsqu'elles sont qualitatives, nous avons ceux de l'ACM.

En effet,

- Comme en ACP normée, l'inertie d'une variable est égale à 1, et l'inertie totale est égale au nombre de variables.
- Comme en ACM (à un coefficient multiplicatif près),

$$I(modalité) = 1 - p_k$$
$$I(variable) = m_j - 1$$

Le lecteur peut se rendre compte très facilement de ces équivalences en lançant un programme d'AFDM (un de ceux que nous décrirons dans ce chapitre) sur les données des parties consacrées à l'ACP (chapitre 1) et l'ACM (chapitre 5), et en comparant les résultats.

6.2.3 ACP sur tableau de données modifié

Nous travaillons sous Python dans cette section. Nous importons les données « Autos 2005 » et nous effectuons les transformations nécessaires pour faire appel ensuite l'ACP du package « fanalysis ». Les résultats obtenus correspondent à ceux de l'analyse factorielle des données mixtes. Nous vérifions alors si la première valeur propre correspond bien à la somme des carrés des corrélations et des rapports de corrélation des variables avec le 1er facteur.

6.2.3.1 Importation et préparation des données

```
#chargement des données
import pandas
DataALL = pandas.read_excel("Data_Methodes_Factorielles.xlsx",sheet_name="AFDM_AUTOS",index_col=0)

#affichage des caractéristiques
print(DataALL.info())

#nombre d'observations
n = DataALL.shape[0]
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, GOLF to SANTA_FE
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   puissance  10 non-null     int64
 1   longueur   10 non-null     int64
 2   hauteur    10 non-null     int64
 3   poids      10 non-null     int64
 4   CO2        10 non-null     int64
 5   origine    10 non-null     object
 6   carburant  10 non-null     object
```

```
 7   4X4        10 non-null     object
dtypes: int64(5), object(3)
memory usage: 720.0+ bytes
```

La transformation des variables quantitatives ne pose aucune difficulté. C'est l'écart-type de la population (σ_k) qui est utilisée pour réduire les valeurs.

```
#récupérer les variables quantitatives
DC = DataALL.iloc[:,:5]
print(DC)
```

```
             puissance  longueur  hauteur  poids  CO2
Modele
GOLF                75       421      149   1217  143
CITRONC4           138       426      146   1381  142
P607               204       491      145   1723  223
VELSATIS           150       486      158   1735  188
CITRONC2            61       367      147    932  141
CHRYS300           340       502      148   1835  291
AUDIA3             102       421      143   1205  168
OUTLAND            202       455      167   1595  237
PTCRUISER          223       429      154   1595  235
SANTA_FE           125       450      173   1757  197
```

```
#librairie numpy
import numpy

#lesquelles (variables quanti) sont centrées et réduites
ZC = (DC.values-numpy.mean(DC.values,axis=0))/numpy.std(DC.values,axis=0)
print(ZC)
```

```
[[-1.1028751  -0.61403051 -0.41885391 -0.98856229 -1.12656547]
 [-0.30424141 -0.4850325  -0.73299434 -0.41057935 -1.14762277]
 [ 0.53242246  1.19194158 -0.83770782  0.79472655  0.55801841]
 [-0.1521207   1.06294357  0.52356739  0.83701799 -0.17898704]
 [-1.28034925 -2.00720898 -0.62828086 -1.99298388 -1.16868007]
 [ 2.2564571   1.47573719 -0.52356739  1.18944661  1.98991471]
 [-0.76060352 -0.61403051 -1.04713477 -1.03085373 -0.60013301]
 [ 0.50706901  0.26315593  1.46598868  0.34361791  0.85282059]
 [ 0.77328024 -0.4076337   0.10471348  0.34361791  0.81070599]
 [-0.46903884  0.13415793  2.09426954  0.91455228  0.01052865]]
```

Pour les variables qualitatives, après le codage 0/1, nous calculons les fréquences relatives (proportions) des modalités pour pouvoir effectuer la transformation.

```
#récupérer les variables qualitatives
DD = DataALL.iloc[:,5:]
print(DD)
```

```
            origine carburant  4X4
Modele
GOLF         Europe    Diesel  non
CITRONC4     France    Diesel  non
P607         France    Diesel  non
VELSATIS     France    Diesel  non
```

```
CITRONC2      France   Essence  non
CHRYS300      Autres   Essence  non
AUDIA3        Europe   Essence  non
OUTLAND       Autres    Diesel  oui
PTCRUISER     Autres   Essence  non
SANTA_FE      Autres    Diesel  oui
```

```
#codage 0/1
DDcoding = pandas.get_dummies(DD)
print(DDcoding)
```

```
             origine_Autres  origine_Europe  origine_France  \
Modele
GOLF                      0               1               0
CITRONC4                  0               0               1
P607                      0               0               1
VELSATIS                  0               0               1
CITRONC2                  0               0               1
CHRYS300                  1               0               0
AUDIA3                    0               1               0
OUTLAND                   1               0               0
PTCRUISER                 1               0               0
SANTA_FE                  1               0               0

             carburant_Diesel  carburant_Essence  4X4_non  4X4_oui
Modele
GOLF                        1                  0        1        0
CITRONC4                    1                  0        1        0
P607                        1                  0        1        0
VELSATIS                    1                  0        1        0
CITRONC2                    0                  1        1        0
CHRYS300                    0                  1        1        0
AUDIA3                      0                  1        1        0
OUTLAND                     1                  0        0        1
PTCRUISER                   0                  1        1        0
SANTA_FE                    1                  0        0        1
```

```
#proportions
p_k = numpy.mean(DDcoding.values,axis=0)
print(p_k)
```

```
[0.4 0.2 0.4 0.6 0.4 0.8 0.2]
```

```
#standardisation des indicatrices
ZD = DDcoding.values/numpy.sqrt(p_k)
print(ZD)
```

```
[[0.         2.23606798 0.         1.29099445 0.         1.11803399
  0.        ]
 [0.         0.         1.58113883 1.29099445 0.         1.11803399
  0.        ]
 [0.         0.         1.58113883 1.29099445 0.         1.11803399
  0.        ]
 [0.         0.         1.58113883 1.29099445 0.         1.11803399
  0.        ]
 [0.         0.         1.58113883 0.         1.58113883 1.11803399
  0.        ]
 [1.58113883 0.         0.         0.         1.58113883 1.11803399
  0.        ]
 [0.         2.23606798 0.         0.         1.58113883 1.11803399
  0.        ]
 [1.58113883 0.         0.         1.29099445 0.         0.
```

```
 2.23606798]
[1.58113883 0.         0.         0.         1.58113883 1.11803399
 0.        ]
[1.58113883 0.         0.         1.29099445 0.         0.
 2.23606798]]
```

Les deux blocs (ZC, ZD) sont assemblés dans une seule structure (z_{ik}) où i = 1, ..., n ; j = 1, ..., P.

```
#assembler les deux ensembles Z
Z = numpy.concatenate((ZC,ZD),axis=1)
print(Z)
```

```
[[-1.1028751  -0.61403051 -0.41885391 -0.98856229 -1.12656547  0.
   2.23606798  0.          1.29099445  0.          1.11803399  0.        ]
 [-0.30424141 -0.4850325  -0.73299434 -0.41057935 -1.14762277  0.
   0.          1.58113883  1.29099445  0.          1.11803399  0.        ]
 [ 0.53242246  1.19194158 -0.83770782  0.79472655  0.55801841  0.
   0.          1.58113883  1.29099445  0.          1.11803399  0.        ]
 [-0.1521207   1.06294357  0.52356739  0.83701799 -0.17898704  0.
   0.          1.58113883  1.29099445  0.          1.11803399  0.        ]
 [-1.28034925 -2.00720898 -0.62828086 -1.99298388 -1.16868007  0.
   0.          1.58113883  0.          1.58113883  1.11803399  0.        ]
 [ 2.2564571   1.47573719 -0.52356739  1.18944661  1.98991471  1.58113883
   0.          0.          0.          1.58113883  1.11803399  0.        ]
 [-0.76060352 -0.61403051 -1.04713477 -1.03085373 -0.60013301  0.
   2.23606798  0.          0.          1.58113883  1.11803399  0.        ]
 [ 0.50706901  0.26315593  1.46598868  0.34361791  0.85282059  1.58113883
   0.          0.          1.29099445  0.          0.          2.23606798]
 [ 0.77328024 -0.4076337   0.10471348  0.34361791  0.81070599  1.58113883
   0.          0.          0.          1.58113883  1.11803399  0.        ]
 [-0.46903884  0.13415793  2.09426954  0.91455228  0.01052865  1.58113883
   0.          0.          1.29099445  0.          0.          2.23606798]]
```

6.2.3.2 ACP sur données transformées

Nous pouvons dès lors instancier et appliquer l'objet de calcul ACP de « fanalysis ».

```
#classe de calcul
from fanalysis.pca import PCA

#instanciation - les données sont déjà standardisées
afdm = PCA(std_unit=False,row_labels=DataALL.index,col_labels=[DC.columns,DDcoding.columns])

#lancement des calculs
afdm.fit(Z)
```

Nous affichons les valeurs propres, nous notons que seules les (H_{max} = 9) premières sont non-nulles. En effet, 3 colonnes parmi les 12 sont redondantes dans les données présentées à l'ACP.

```
#affichage des valeurs propres
print(afdm.eig_[0])
```

```
[4.27313563e+00 2.12188732e+00 1.43871787e+00 8.36367144e-01
 1.64030083e-01 1.14470293e-01 3.36251524e-02 1.58052203e-02
 1.96128129e-03 1.08736851e-32]
```

6.2.3.3 Qualité de restitution du 1er facteur

Nous disposons des coordonnées des individus sur le 1er facteur (F_1).

```
#coordonnées des individus sur le 1er facteur (F1)
print(afdm.row_coord_[:,0])
```

```
[ 2.31780488  1.44536838 -0.77973114 -0.54106289  3.27586403 -2.95770457
  2.31613266 -2.25591897 -0.93180905 -1.88894333]
```

Nous pouvons les exploiter pour calculer le carré des corrélations des variables quantitatives avec (F_1).

```
#carré corrélations des variables quantitatives avec F1
corr2 = (numpy.corrcoef(afdm.row_coord_[:,0],DC.values,rowvar=False)**2)[0,1:]
print(corr2)
```

```
[0.67132603 0.6339052  0.33451975 0.86400031 0.79327801]
```

Pour les carrés des rapports de corrélation, nous effectuons les calculs variable par variable, sans oublier que les facteurs sont centrés, et que la variance totale d'un facteur correspond à sa valeur propre. Ainsi, pour « Origine », « Carburant » et « 4X4 » :

```
#data frame temporaire
temp = DD.copy()
temp['F1'] = afdm.row_coord_[:,0]

#carré du rapport de corrélation pour origine
eta_origine = numpy.sum((pandas.pivot_table(temp,values='F1',index='origine').valu
es[:,0]**2)*p_k[:3])/afdm.eig_[0][0]
print(eta_origine)
```

```
0.6965665387607467
```

```
#carré du rapport de corrélation pour carburant
eta_carburant = numpy.sum((pandas.pivot_table(temp,values='F1',index='carburant').
values[:,0]**2)*p_k[3:5])/afdm.eig_[0][0]
print(eta_carburant)
```

```
0.028262312028001453
```

```
#carré du rapport de corrélation pour 4X4
eta_4X4 = numpy.sum((pandas.pivot_table(temp,values='F1',index='4X4').values[:,0]*
*2)*p_k[5:])/afdm.eig_[0][0]
print(eta_4X4)
```

```
0.2512774715903162
```

Nous effectuons les sommes idoines :

$$\lambda_1 = \sum_{j=1}^{C} r^2(F_1, X_j) + \sum_{j=C+1}^{C+D} \eta^2(F_1, X_j)$$
$$= (0.671 + 0.634 + \cdots) + (0.697 + 0.028 + 0.251)$$
$$= \mathbf{4.271}$$

```
#somme des carrés des corrélations et rapports de corrélation
sum_sq = numpy.sum(corr2) + eta_origine + eta_carburant + eta_4X4
print(sum_sq)

4.27313562699487

#première valeur propre de l'AFDM
print(afdm.eig_[0][0])

4.27313562699487
```

Nous réaffichons la première valeur propre de l'ACP sur variables préparées au cas où nous aurions des doutes. L'inertie portée par le 1er facteur correspond bien au critère de l'AFDM. La méthode consiste à maximiser la liaison globale des variables avec les facteurs, au sens de la somme des carrés de la corrélation linéaire pour les quantitatives, et des carrés du rapport de corrélation pour les qualitatives.

6.2.3.4 Coordonnées des individus et des variables-modalités

Nous en aurons l'usage dans les sections suivantes lorsqu'il faudra décortiquer les sorties des logiciels. Nous noterons (F_{ih}) les coordonnées de l'individu n°i (i = 1, …, n) pour le facteur n°h (h = 1, …, H_{max}) fournies par l'ACP sur des données préparées (z_{ik}). Pour les données « Autos 2005 », nous avons :

```
#coordonnées des individus
print(pandas.DataFrame(afdm.row_coord_[:,:2],columns=['F1','F2'],index=DataALL.index))

                  F1        F2
Modele
GOLF        2.317805 -0.687271
CITRONC4    1.445368 -0.122286
P607       -0.779731  1.018563
VELSATIS   -0.541063 -0.160059
CITRONC2    3.275864  0.025117
CHRYS300   -2.957705  2.628112
AUDIA3      2.316133  0.795341
```

```
OUTLAND      -2.255919 -1.840426
PTCRUISER    -0.931809  1.090852
SANTA_FE     -1.888943 -2.747943
```

Et (G_{kh}) les coordonnées de la variable-modalité n°k (soit une variable quantitative, soit une indicatrice de modalité issue d'une variable qualitative) pour le facteur n°h.

```
#coordonnées des variables-modalités
print(pandas.DataFrame(afdm.col_coord_[:,:2],columns=['F1','F2'],index=[list(DC.columns)+list(DDcoding.columns)]))
```

```
                         F1        F2
puissance         -0.819345  0.539186
longueur          -0.796182  0.244889
hauteur           -0.578377 -0.763585
poids             -0.929516  0.061131
CO2               -0.890662  0.378984
origine_Autres    -0.614538 -0.094369
origine_Europe     0.501259  0.016589
origine_France     0.260095  0.082639
carburant_Diesel  -0.106325 -0.402313
carburant_Essence  0.130221  0.492730
4X4_non            0.224177  0.352170
4X4_oui           -0.448355 -0.704340
```

6.3 Pratique de l'AFDM sous TANAGRA

Il n'y a pas de librairie Python proposant l'AFDM telle que nous la présentons dans ce chapitre à l'heure actuelle (juin 2020), à ma connaissance tout du moins. J'avais cru trouver mon bonheur avec le package « prince ». Mais à l'épreuve des données, je me suis rendu compte que les résultats étaient différents. J'ai préféré jouer la sécurité en m'appuyant sur des outils dont je connais parfaitement les tenants et aboutissants pour présenter les résultats de la méthode et décrire les formules sous-jacentes. J'utilise en l'occurrence le logiciel TANAGRA dans cette section. Nous nous pencherons sur plusieurs packages R plus bas (section 6.4).

6.3.1 Nombre de facteurs

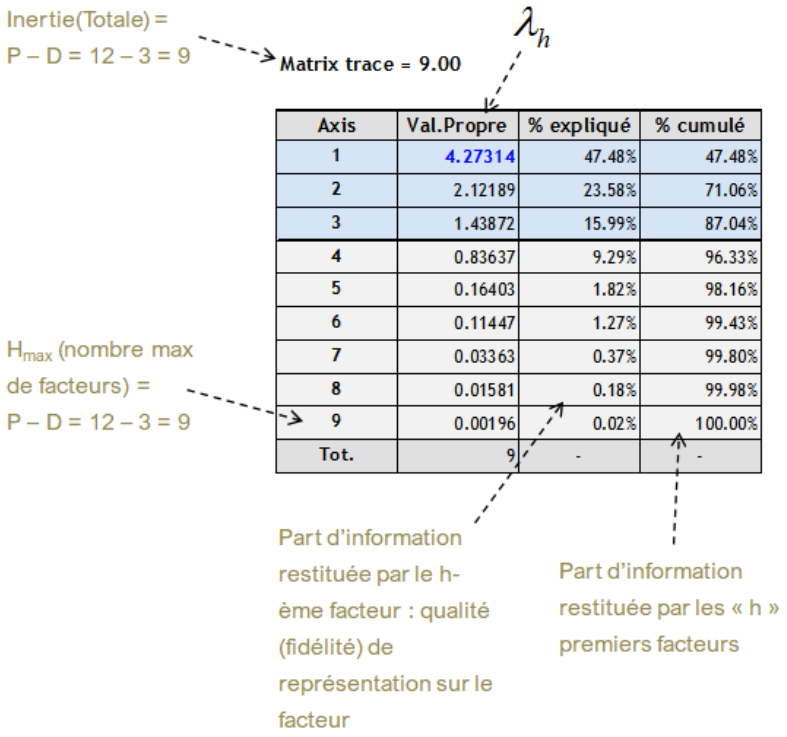
Le nombre de facteurs pertinents « H » est l'arlésienne de l'analyse factorielle. On en parle tout le temps mais personne n'a de solution qui s'impose véritablement, un peu comme le nombre de groupes dans la classification automatique.

Nous en avons longuement parlé notamment pour l'ACP et l'ACM. Le problème est peut-être encore plus complexe avec l'AFDM. Nous savons au moins que le nombre maximum de facteurs est égal à :

$$H_{max} = P - D$$

Puisque nous avons introduit artificiellement D redondances (D est le nombre de variables qualitatives) quand les données recodées puisque la dernière indicatrices peut être déduite des autres dans le codage disjonctif complet.

Voici le tableau des valeurs propres (λ_h) pour les données « Autos 2005 ».



Axis	Val.Propre	% expliqué	% cumulé
1	4.27314	47.48%	47.48%
2	2.12189	23.58%	71.06%
3	1.43872	15.99%	87.04%
4	0.83637	9.29%	96.33%
5	0.16403	1.82%	98.16%
6	0.11447	1.27%	99.43%
7	0.03363	0.37%	99.80%
8	0.01581	0.18%	99.98%
9	0.00196	0.02%	100.00%
Tot.	9	-	-

Figure 181 – Tableau des valeurs propres – AFDM – Données "Autos 2005"

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

- Avec la règle de Kaiser (section 1.3.1.3), nous sélectionnerons les facteurs tels que ($\lambda_h \geq 1$) c.-à-d. 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 redondante comme

nous avons pu le voir pour l'ACM (sections 5.3.1, 5.3.2 et 5.3.3). La question de la correction à introduire reste un problème ouvert pour l'heure.

- Avec la règle de Karlis-Saporta-Spinaki (section 1.3.1.3), la valeur seuil est surestimée parce que le nombre de colonnes (P) 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 à ce bon vieux le diagramme des valeurs propres et la recherche de « coudes », annonciateurs de changement significatif de structure dans les données.

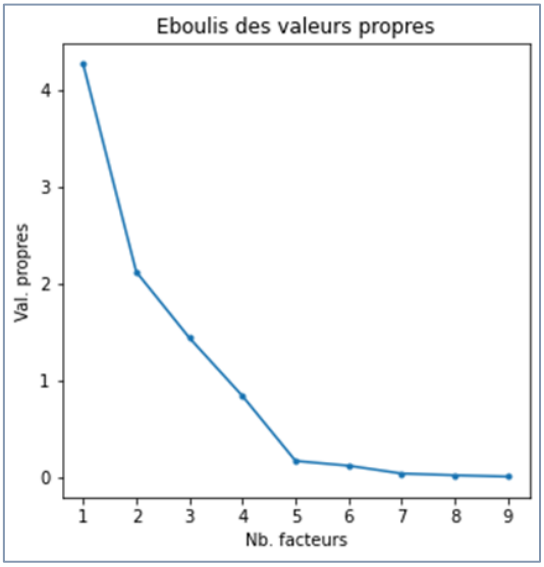


Figure 182 - Diagramme des valeurs propres - AFDM - Données "Autos 2005"

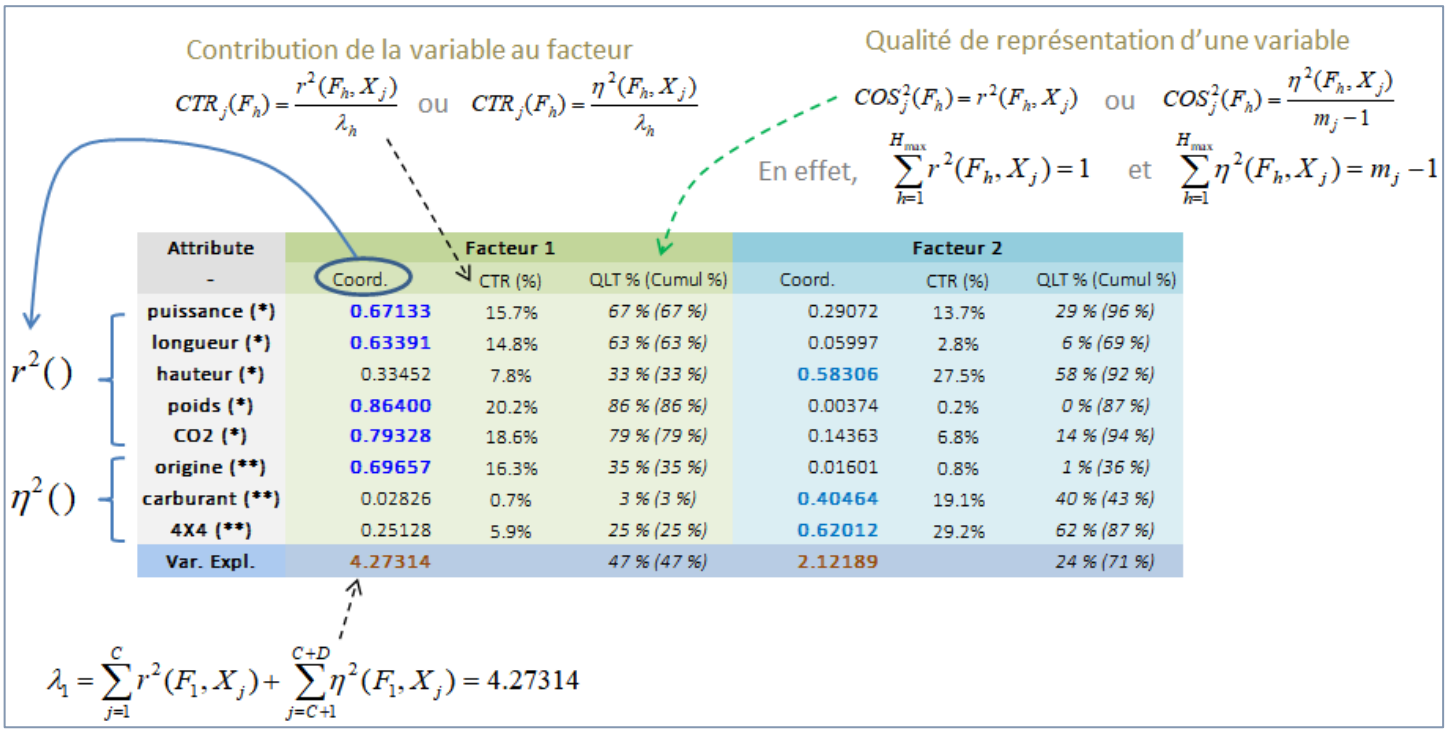
Nous avons un « coude » assez marqué au niveau de « h = 2 ». Sachant que le second facteur (F2) porte 23.58% de l'information disponible, nous choisissons de retenir (H = 2) axes factoriels pour l'analyse. C'est heureux. En dessous il aurait été difficile de réaliser des cartes factorielles.

6.3.2 Représentation des variables

6.3.2.1 Analyse simultanée des variables quantitatives et qualitatives

La représentation des variables quantitatives et qualitatives dans le même repère est une vraie gageure. La solution adoptée par une grande partie des outils est de mêler dans un tableau les carrés des corrélations (variables quantitatives) et des rapports de corrélation (qualitatives) avec les facteurs. Rappelons que ce sont là deux indicateurs d'intensité de relation qui varient entre 0 et 1. Leurs valeurs sont comparables. Ce dispositif permet de faire apparaître les contributions (CTR) des variables et leurs qualité de représentation (COS2) (Figure 183).

Il n'est pas nécessaire de calculer les coordonnées ex-post à partir de l'expression des facteurs. Les coordonnées des variables-modalités (G_{kh}) (section 6.2.3.4) suffisent : nous avons la correspondance avec le carré de la corrélation [$r^2(F_h, X_k) = G_{kh}^2$] pour les variables quantitatives ; les carrés des rapports de corrélation des variables qualitatives peuvent en être déduites [$\eta^2(F_h, X_j) = \sum_{k \in X_j} G_{kh}^2$].



Attribute	Facteur 1			Facteur 2		
-	Coord.	CTR (%)	QLT % (Cumul %)	Coord.	CTR (%)	QLT % (Cumul %)
puissance (*)	0.67133	15.7%	67 % (67 %)	0.29072	13.7%	29 % (96 %)
longueur (*)	0.63391	14.8%	63 % (63 %)	0.05997	2.8%	6 % (69 %)
hauteur (*)	0.33452	7.8%	33 % (33 %)	0.58306	27.5%	58 % (92 %)
poids (*)	0.86400	20.2%	86 % (86 %)	0.00374	0.2%	0 % (87 %)
CO2 (*)	0.79328	18.6%	79 % (79 %)	0.14363	6.8%	14 % (94 %)
origine (**)	0.69657	16.3%	35 % (35 %)	0.01601	0.8%	1 % (36 %)
carburant (**)	0.02826	0.7%	3 % (3 %)	0.40464	19.1%	40 % (43 %)
4X4 (**)	0.25128	5.9%	25 % (25 %)	0.62012	29.2%	62 % (87 %)
Var. Expl.	4.27314		47 % (47 %)	2.12189		24 % (71 %)

Figure 183 - Coordonnées des variables, contributions et COS2 - AFDM - Données "Autos 2005"

Pour récapituler, si la variable est quantitative (resp. qualitative) : (**Coord**) sa coordonnée sur le facteur F_h correspond au carré de la corrélation (resp. carré du rapport de corrélation) ; (**CTR**) sa contribution est la ratio entre la coordonnée et la variance restituée par le facteur ; (COS2) sa qualité de représentation est égale à sa coordonnée (resp. ramenée au nombre de ses modalités moins 1).

Pour les données « Autos 2005 », si nous nous en tenons aux deux premiers facteurs :

- (Poids, CO2, Origine, Puissance, Longueur) pèsent (intensité) sur le 1er facteur qui restitue 47% de l'information disponible. Mais on ne saurait pas dire quelles relations (sens) entretiennent ces variables pour l'instant.
- Sur le 2nd (24% de l'information), (4X4, Hauteur, Carburant, Puissance) sont les plus déterminants.

Sur ces 2 premiers facteurs, les variables sont plutôt bien représentées à l'exception de « Origine » (COS2 = 36%) et « Carburant » (43%).

Lorsque le nombre de variables est élevé, il est pratique de passer par une représentation graphique. Voici ce qu'il en est dans le premier plan factoriel (Figure 184).

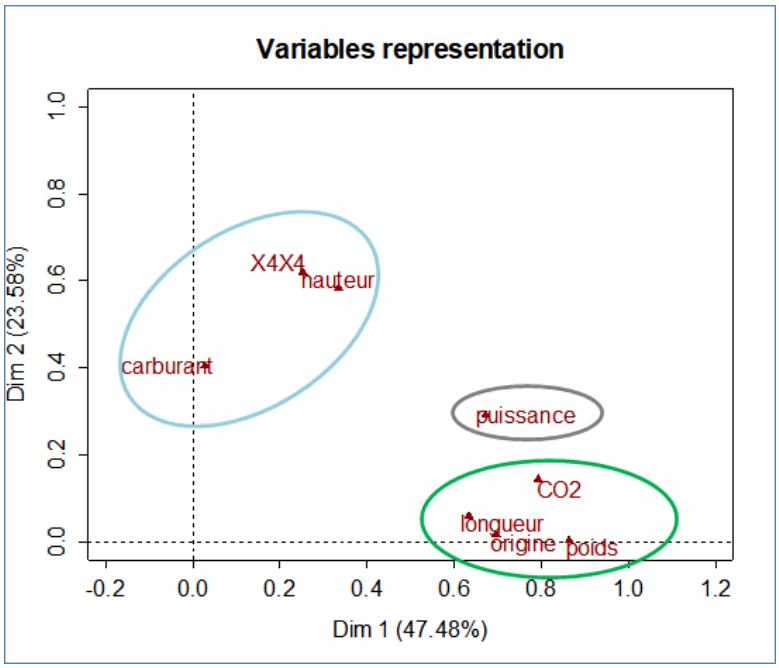


Figure 184 – Représentation des variables dans le 1er plan factoriel – AFDM – Données "Autos 2005"

Mis à part (Puissance), le rattachement des variables aux facteurs est assez tranché. Mais, à ce stade, nous ne pouvons pas décrire réellement la nature de ces derniers (ex. « Poids » est corrélé positivement avec « Longueur » ? Négativement ? Quel type de « Carburant » est associé aux « 4X4 » sur le 2nd facteur ? Etc.).

6.3.2.2 Analyse des variables quantitatives

Pour spécifiquement les variables quantitatives, la corrélation est un indicateur naturel pour caractériser les facteurs (voir ACP, section 1.3.2).

Variables quantitatives - Corrélations

Attribute	Fact.1	Fact.2
puissance	-0.81935	0.53919
longueur	-0.79618	0.24489
hauteur	-0.57838	-0.76359
poids	-0.92952	0.06113
CO2	-0.89066	0.37898

Figure 185 – Corrélations avec les facteurs – AFDM – Données "Autos 2005"

Nous savons maintenant qu'il y a un « effet taille » sur le premier facteur : les véhicules imposants sont les plus puissants, les plus polluants. Sur le 2nd, nous comprenons qu'à « taille » égale, il y a une opposition entre la puissance et la hauteur. Nous pouvons utiliser un cercle des corrélations pour représenter les variables (Figure 186).

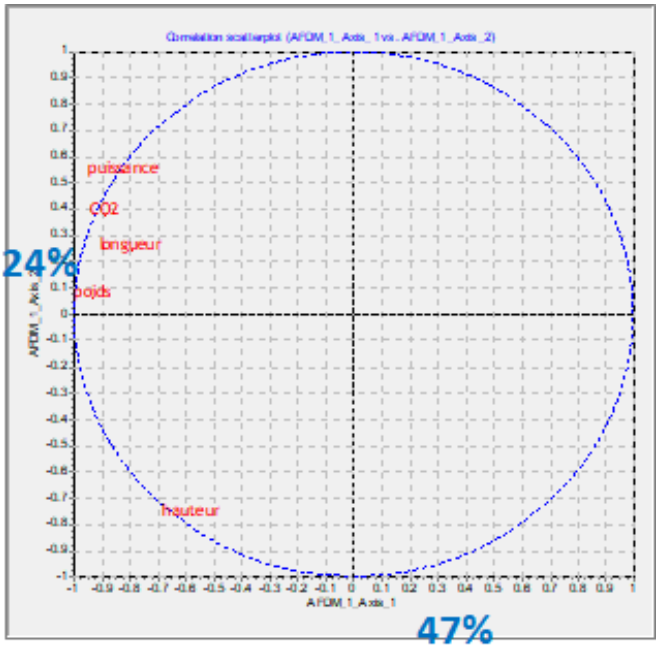


Figure 186 – Cercle des corrélation des variables quantitatives – AFDM – Données "Autos 2005"

6.3.2.3 Analyse des variables qualitatives – Les points modalités

L'analyse précédente est approfondie avec l'étude des modalités des variables qualitatives. Leurs coordonnées correspondent aux moyennes conditionnelles sur les facteurs (μ_{kh}) (Figure 187).

Discrete Attributes - Conditional means and contributions

Attribute		Axis_1			Axis_2		
-		Mean	CTR (%)	v.test	Mean	CTR (%)	v.test
origine	Europe	2.3170	5.88	1.681	0.0540	0.01	0.056
	France	0.8501	1.58	1.007	0.1903	0.32	0.320
	Autres	-2.0086	8.84	-2.380	-0.2174	0.42	-0.365
	Tot.	-	16.30	-	-	0.75	-
carburant	Diesel	-0.2837	0.26	-0.504	-0.7566	7.63	-1.908
	Essence	0.4256	0.40	0.504	1.1349	11.44	1.908
	Tot.	-	0.66	-	-	19.07	-
4X4	non	0.5181	1.18	1.504	0.5735	5.84	2.362
	oui	-2.0724	4.70	-1.504	-2.2942	23.38	-2.362
	Tot.	-	5.88	-	-	29.22	-

Figure 187 – Tableau des modalités – AFDM – Données "Autos 2005"

Ici non plus il n'est pas nécessaire de les calculer après coup, nous pouvons exploiter les coordonnées factorielles des points-modalités :

$$\mu_{kh} = G_{kh} \times \sqrt{\frac{\lambda_h}{p_k}}$$

Aux modalités peuvent être associés des contributions, lesquelles s'additionnent pour aboutir aux contributions des variables cohérentes avec celles du tableau dédié (Figure 183).

Nous comprenons maintenant pour les données « Autos 2005 » que l'effet taille caractéristique du 1er facteur repose sur l'opposition entre les véhicules d'origine européennes et autres. Ces derniers étant plutôt associés aux véhicules tous-terrains (4X4). Pour (F2), les véhicules hauts sont plutôt des 4X4, ils carburent préférentiellement au gazole (diesel).

Attention dans ce genre de lecture, les valeurs des corrélations ne peuvent pas être rapprochés aux moyennes conditionnelles des modalités. Il faut plutôt raisonner en termes de directions. Je le précise parce que certains outils proposent des « biplot » mélangeant

allègrement les individus, les variables quantitatives, les modalités des qualitatives. Pourquoi pas. Nous devons simplement être très attentifs à bien comprendre les idées qui sous-tendent ce type de représentation. Ce sont les directions (j'ai bien insisté là) qui importent alors et non pas les proximités.

6.3.3 Représentation des individus

Les coordonnées factorielles (F_{ih}) des individus sont directement fournies par l'ACP sur données transformées (section 6.2.3.4). Nous pouvons les accompagner des aides à l'interprétation usuelles : les contributions (CTR) et les qualités de représentation (COS2) (Figure 188).

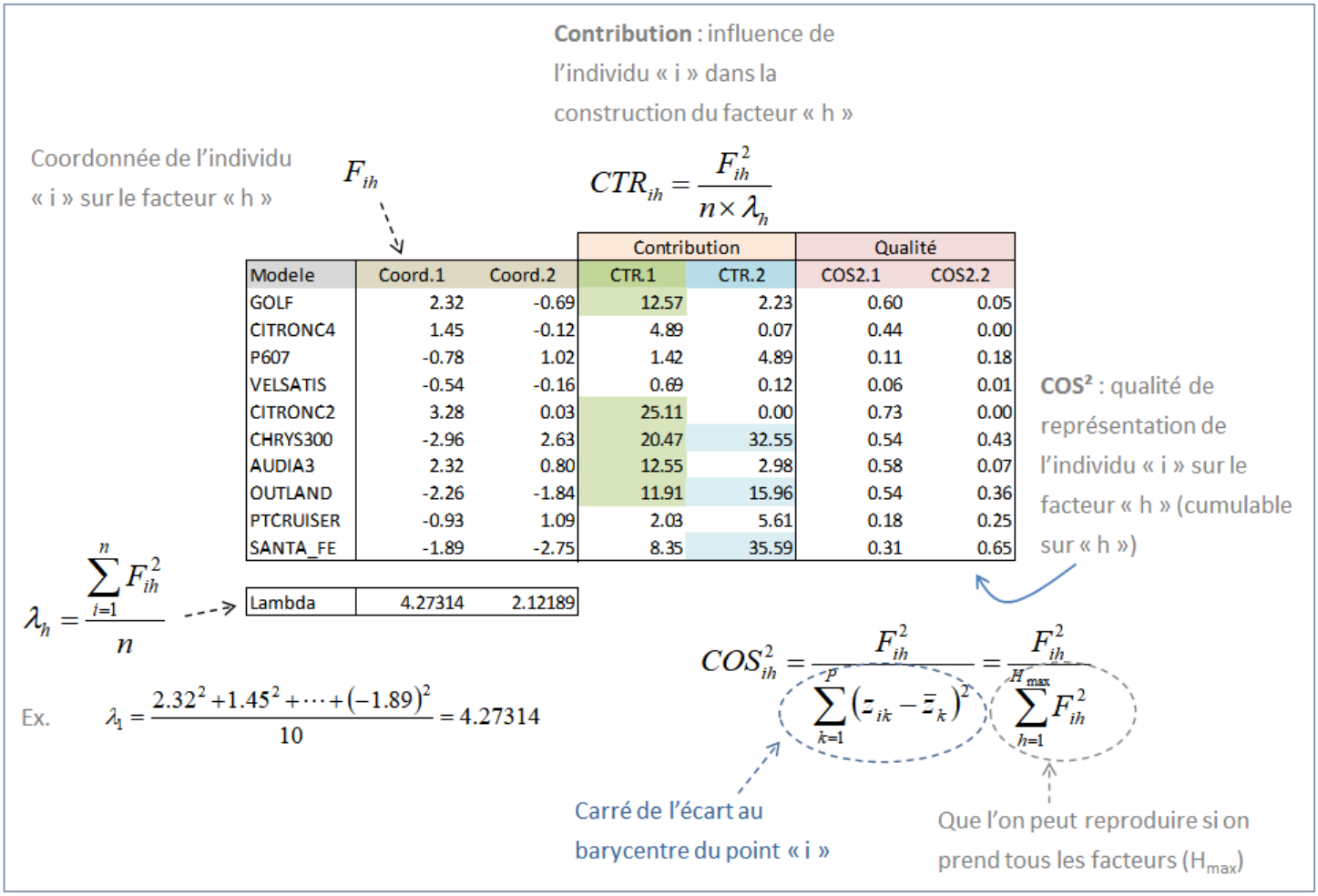


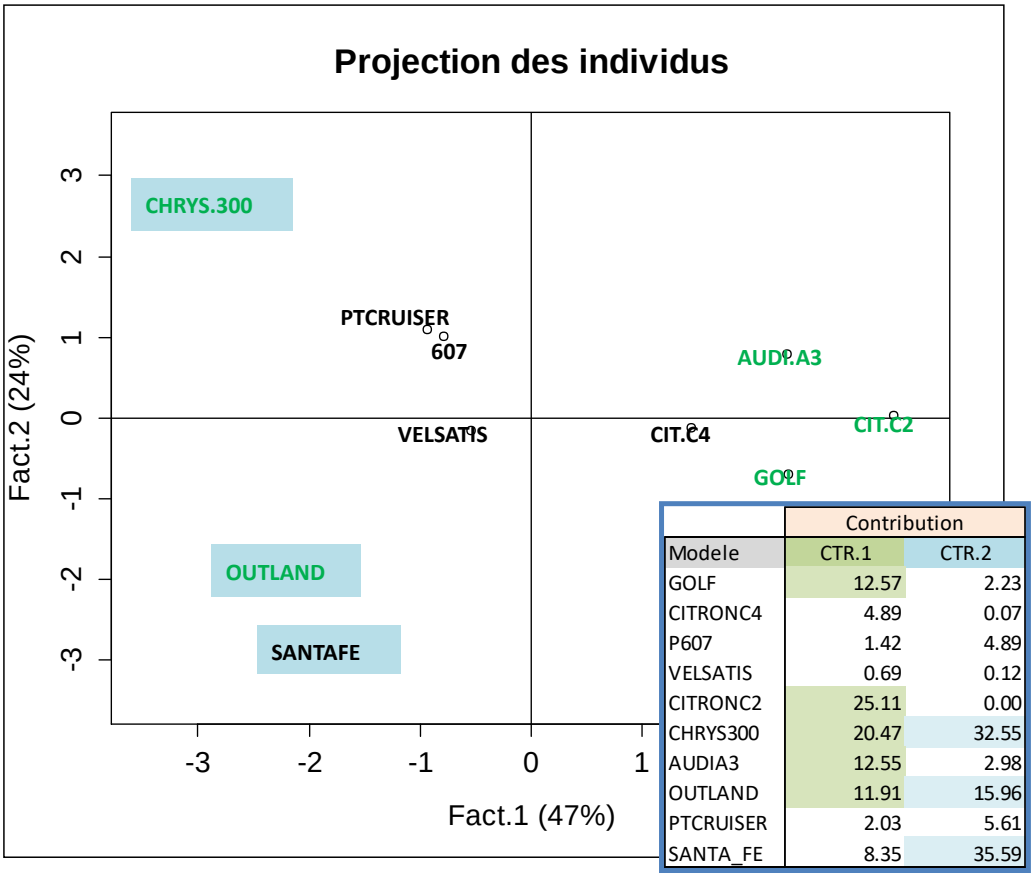
Figure 188 – Coordonnées des individus et aides à l'interprétation – AFDM – Données "Autos 2005"

Ainsi :

- Sur le 1er facteur, les véhicules emblématiques de l'opposition basée sur l'encombrement sont (Citroën C2, VW Golf, Audi A3) vs. (Chrysler 300, Outlander).

- Sur le 2nd, nous constatons plutôt l'opposition (Chrysler 300) vs. (Outlander, Santa Fe).

Observer ces configurations dans un graphique est autrement plus sympathique. Nous avons adopté un code couleur pour identifier les contributions importantes selon les axes (Figure 189).



	Contribution	
Modele	CTR.1	CTR.2
GOLF	12.57	2.23
CITRONC4	4.89	0.07
P607	1.42	4.89
VELSATIS	0.69	0.12
CITRONC2	25.11	0.00
CHRYS300	20.47	32.55
AUDIA3	12.55	2.98
OUTLAND	11.91	15.96
PTCRUISER	2.03	5.61
SANTA_FE	8.35	35.59

Figure 189 – Carte des individus illustrés selon leur contribution – AFDM – Données "Autos 2005"

6.3.4 Individus supplémentaires

A l'instar de l'ACP (section 1.5) et l'ACM (section 5.3.7), il est possible de projeter un individu supplémentaire dans le repère factoriel. Pour notre exemple « Autos 2005 », nous souhaitons situer la « Nissan X-TRAIL » par rapport aux individus actifs. Voici ses caractéristiques :

Modele	puissance	longueur	hauteur	poids	CO2	origine	carburant	4X4
X-TRAIL	136	446	168	1520	190	Autres	Diesel	oui

Puisque le mécanisme interne repose sur une ACP, nous pouvons exploiter les vecteurs propres issus de la diagonalisation matricielle ou de la décomposition en valeurs singulières (section 1.2), non sans avoir préparer préalablement le vecteur de description :

- Il faut tout d'abord coder en indicatrices de modalités les variables qualitatives.

Modele	puissance	longueur	hauteur	poids	CO2	orig_Autres	orig_Europe	orig_France	carb_Diesel	carb_Essence	4X4_non	4X4_oui
X-TRAIL	136	446	168	1520	190	1	0	0	1	0	0	1

- Puis appliquer les paramètres de transformation pour le centrage-réduction.

Variable	puissance	longueur	hauteur	poids	CO2	orig_Autres	orig_Europe	orig_France	carb_Diesel	carb_Essence	4X4_non	4X4_oui
Moyenne	162	444.8	153	1497.5	196.5	0.4	0.2	0.4	0.6	0.4	0.8	0.2
Ecart-type	78.88	38.76	9.55	283.75	47.49	0.63	0.45	0.63	0.77	0.63	0.89	0.45

- Appliquer enfin les coefficients des vecteurs pour obtenir les coordonnées factorielles.

Variable	puissance	longueur	hauteur	poids	CO2	orig_Autres	orig_Europe	orig_France	carb_Diesel	carb_Essence	4X4_non	4X4_oui
Fact.1	-0.396	-0.385	-0.280	-0.450	-0.431	-0.297	0.242	0.126	-0.051	0.063	0.108	-0.217
Fact.2	0.370	0.168	-0.524	0.042	0.260	-0.065	0.011	0.057	-0.276	0.338	0.242	-0.484

Sur les 2 facteurs, voici le détail des opérations :

$$F_{X-TRAIL,1} = -0.389 \times \left(\frac{136-162}{78.88}\right) + \cdots - 0.217 \times \left(\frac{1-0.2}{0.45}\right) = -1.32$$

$$F_{X-TRAIL,2} = 0.370 \times \left(\frac{136-162}{78.88}\right) + \cdots - 0.484 \times \left(\frac{1-0.2}{0.45}\right) = -2.51$$

Le nouveau véhicule se place parmi ses congénères tous-terrains (Figure 190).

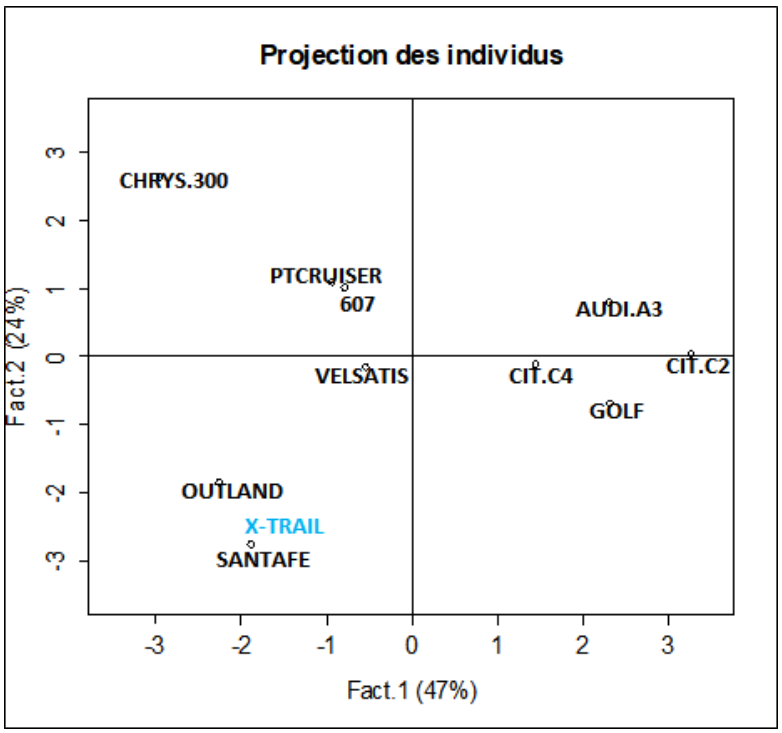


Figure 190 – Individu supplémentaire – AFDM – Données "Autos 2005"

Pour faciliter le déploiement en dehors de l'outil, via un tableur par exemple, TANAGRA fournit les paramètres de transformation et les coefficients de projection dans le tableau « Factor Scores » (Figure 191).

Eigen vectors - Factor Scores

Attribute	Center	Scale	Axis_1	Axis_2
puissance	162.000000	78.884726	-0.396363	0.370150
longueur	444.800000	38.760289	-0.385158	0.168116
hauteur	153.000000	9.549869	-0.279793	-0.524199
poids	1497.500000	283.745396	-0.449659	0.041966
CO2	196.500000	47.489473	-0.430863	0.260172
origine = Europe	0.200000	0.447214	0.242487	0.011389
origine = France	0.400000	0.632456	0.125822	0.056731
origine = Autres	0.400000	0.632456	-0.297287	-0.064784
carburant = Diesel	0.600000	0.774597	-0.051435	-0.276187
carburant = Essence	0.400000	0.632456	0.062995	0.338258
4X4 = non	0.800000	0.894427	0.108447	0.241764
4X4 = oui	0.200000	0.447214	-0.216894	-0.483527

Figure 191 – Tableau Factor Scores de TANAGRA – AFDM – Données "Autos 2005"

6.3.5 Variables illustratives

Les variables illustratives ne servent pas pour la construction des facteurs, elles sont exploitées après coup pour mieux les comprendre et interpréter (ACP, section 1.6 ; ACM, section 5.3.8).

Modele	prix	surtaxe
GOLF	19140	non
CITRONC4	23400	non
P607	40550	oui
VELSATIS	38250	oui
CITRONC2	10700	non
CHRYS300	54900	oui
AUDIA3	21630	non
OUTLAND	29990	oui
PTCRUISER	27400	oui
SANTA_FE	27990	oui

Figure 192 – Variables illustratives – Données "Autos 2005"

Pour les données « Autos 2005 », nous souhaitons comprendre la nature des facteurs à la lumière des variables « Prix » et « Surtaxe » (Figure 192). Les calculs n'ont plus de secrets pour nous : pour les quantitatives, nous utilisons le coefficient de corrélation pour mesurer les liaisons

avec les facteurs ; pour les qualitatives, nous utilisons le rapport de corrélation pour évaluer la liaison globale, les moyennes conditionnelles et les valeurs-test pour situer les positions relatives des modalités. Sans expliciter les calculs que nous connaissons bien maintenant (COURS AFDM, pages 27 et 28). Nous constatons que la variable « Prix » est surtout corrélée avec le premier facteur comme en témoigne sa position dans le cercle des corrélations (Figure 193).

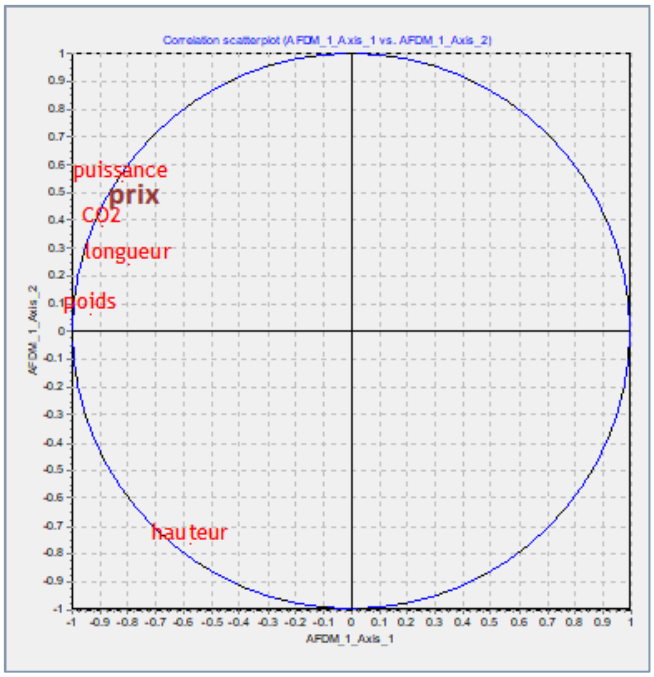


Figure 193 – Variable supplémentaire "Prix" dans le cercle des corrélations – AFDM – Données "Autos 2005"

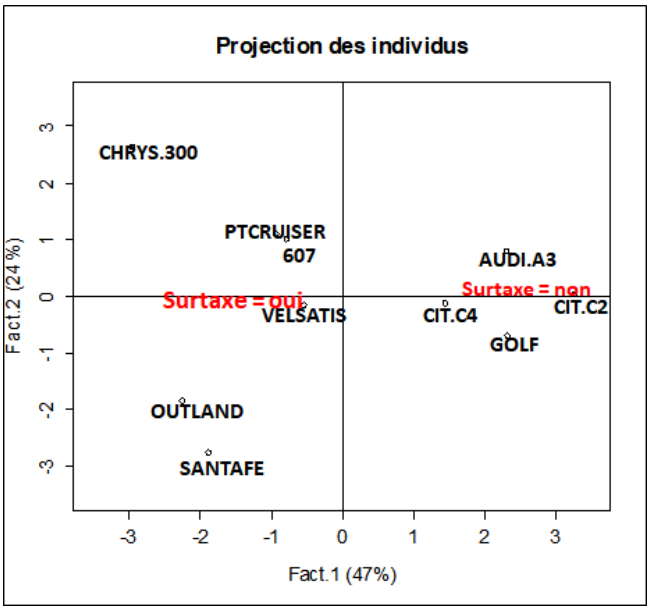


Figure 194 – Position des modalités supplémentaires – AFDM – Données "Autos 2005"

La surtaxe est surtout liée à l'opposition « grosses » vs. « petites » voitures (Figure 194).

6.4 AFDM avec d'autres outils – R et les packages spécialisés

Plusieurs packages implémentent l'analyse factorielle des données mixtes sous R. Ils s'appuient sur des références bibliographiques spécifiques mais, c'est ce qui nous importe en pratique, présentent des résultats numériques identiques. Ils se différencient finalement par le mode de présentation des sorties (COURS ACM, pages 36 à 39 ; TUTO 19, section 4).

6.4.1 Données « Joueurs de Tennis »

Nous utilisons un jeu de données de mon cru pour illustrer la méthode. Il recense les caractéristiques de 20 joueurs de tennis qui ont été classés numéro 1 mondial ces dernières décennies (la question peut se poser pour Guillermo Vilas, mais c'est un tel champion qu'il a tout à fait sa place ici) et qui ont gagné au moins 2 tournois du Grand Chelem.

Joueur	Taille	Lateralite	MainsRevers	Titres	Finales	TitresGC	RolandGarros	BestClassDouble
Agassi	180	droitier	deux	60	30	8	vainqueur	123
Becker	191	droitier	une	49	28	6	demi	6
Borg	180	droitier	deux	64	25	11	vainqueur	890
Connors	178	gaucher	deux	109	52	8	demi	370
Courier	185	droitier	deux	23	13	4	vainqueur	20
Djokovic	188	droitier	deux	79	34	17	vainqueur	114
Edberg	187	droitier	une	41	36	6	finale	1
Federer	185	droitier	une	103	54	20	vainqueur	24
Kafelnikov	190	droitier	deux	26	20	2	vainqueur	4
Kuerten	190	droitier	une	20	9	3	vainqueur	38
Lendl	187	droitier	une	94	50	8	vainqueur	20
McEnroe	180	gaucher	une	77	31	7	finale	1
Murray	191	droitier	deux	46	22	3	finale	51
Nadal	185	gaucher	deux	85	37	19	vainqueur	26
Nastase	180	droitier	une	58	38	2	vainqueur	59
Rafter	185	droitier	une	11	14	2	demi	6
Safin	193	droitier	deux	15	12	2	demi	71
Sampras	185	droitier	une	64	24	14	demi	27
Vilas	180	gaucher	une	62	40	4	vainqueur	175
Wilander	182	droitier	deux	33	27	7	vainqueur	3

Figure 195 – Données "Joueurs de Tennis"

Les joueurs sont décrits par : leur taille, leur latéralité, le fait d'utiliser ou non une prise à deux mains pour leur revers, le nombre de titres, le nombre de finales, le nombre de titres en Grand Chelem, le meilleur résultat à Roland-Garros, le meilleur classement en double. Notre objectif est de mettre en lumière les principales dimensions qui réunissent ou distinguent ces champions.

Opération commune aux outils que nous utilisons dans cette section, nous chargeons et inspectons les données sous R.

```r
#charger les données
library(xlsx)
D <- read.xlsx("Data_Methodes_Factorielles.xlsx",sheetName="AFDM_TENNIS")
rownames(D) <- D$Joueur
D$Joueur <- NULL
print(D)
```

```
##            Taille Lateralite MainsRevers Titres Finales TitresGC RolandGarros
## Agassi        180   droitier        deux     60      30        8    vainqueur
## Becker        191   droitier         une     49      28        6         demi
## Borg          180   droitier        deux     64      25       11    vainqueur
## Connors       178    gaucher        deux    109      52        8         demi
## Courier       185   droitier        deux     23      13        4    vainqueur
## Djokovic      188   droitier        deux     79      34       17    vainqueur
## Edberg        187   droitier         une     41      36        6       finale
## Federer       185   droitier         une    103      54       20    vainqueur
## Kafelnikov    190   droitier        deux     26      20        2    vainqueur
## Kuerten       190   droitier         une     20       9        3    vainqueur
## Lendl         187   droitier         une     94      50        8    vainqueur
## McEnroe       180    gaucher         une     77      31        7       finale
## Murray        191   droitier        deux     46      22        3       finale
## Nadal         185    gaucher        deux     85      37       19    vainqueur
## Nastase       180   droitier         une     58      38        2    vainqueur
## Rafter        185   droitier         une     11      14        2         demi
## Safin         193   droitier        deux     15      12        2         demi
## Sampras       185   droitier         une     64      24       14         demi
## Vilas         180    gaucher         une     62      40        4    vainqueur
## Wilander      182   droitier        deux     33      27        7    vainqueur
##            BestClassDouble
## Agassi                 123
## Becker                   6
## Borg                   890
## Connors                370
## Courier                 20
## Djokovic               114
## Edberg                   1
## Federer                 24
## Kafelnikov               4
## Kuerten                 38
## Lendl                   20
## McEnroe                  1
## Murray                  51
## Nadal                   26
## Nastase                 59
## Rafter                   6
## Safin                   71
## Sampras                 27
## Vilas                  175
## Wilander                 3
```

6.4.2 FactoMineR

La fonction **FAMD()** du package « FactoMineR » cite explicitement l'article de Pagès (2004) (COURS AFDM, page 36). Ses sorties sont directement raccords avec notre présentation de la méthode qui s'appuie sur la même référence.

Après avoir importé la librairie, nous faisons appel à FAMD() en lui passant le data frame à traiter et le nombre de facteurs à générer (ncp = 2).

```
#importation
library(FactoMineR)

#lancement
afdm1 <- FAMD(D,ncp=2)

print(summary(afdm1))

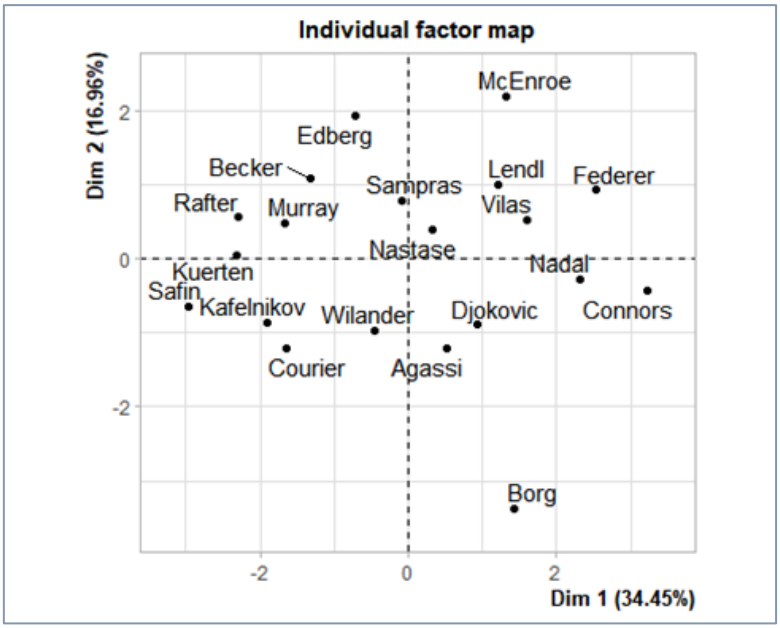
##
## Call:
## FAMD(base = D, ncp = 2)
##
##
## Eigenvalues
##                        Dim.1  Dim.2
## Variance               3.101  1.526
## % of var.             34.454 16.958
## Cumulative % of var. 34.454 51.412
##
## Individuals (the 10 first)
##                      Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2
## Agassi             |  1.817 |  0.526  0.446  0.084 | -1.207  4.770  0.441 |
## Becker             |  2.540 | -1.320  2.809  0.270 |  1.089  3.887  0.184 |
## Borg               |  4.409 |  1.429  3.292  0.105 | -3.375 37.316  0.586 |
## Connors            |  4.345 |  3.220 16.714  0.549 | -0.426  0.595  0.010 |
## Courier            |  2.359 | -1.634  4.308  0.480 | -1.204  4.746  0.260 |
## Djokovic           |  2.432 |  0.926  1.383  0.145 | -0.895  2.623  0.135 |
## Edberg             |  2.820 | -0.705  0.802  0.063 |  1.923 12.120  0.465 |
## Federer            |  3.638 |  2.523 10.267  0.481 |  0.921  2.779  0.064 |
## Kafelnikov         |  2.471 | -1.908  5.871  0.596 | -0.863  2.440  0.122 |
## Kuerten            |  2.860 | -2.321  8.686  0.659 |  0.047  0.007  0.000 |
##
## Continuous variables
##                     Dim.1    ctr   cos2    Dim.2    ctr   cos2
## Taille            | -0.685 15.150  0.470 |  0.153  1.533  0.023 |
## Titres            |  0.934 28.119  0.872 |  0.143  1.341  0.020 |
## Finales           |  0.860 23.840  0.739 |  0.283  5.233  0.080 |
## TitresGC          |  0.671 14.520  0.450 | -0.066  0.288  0.004 |
## BestClassDouble   |  0.374  4.501  0.140 | -0.665 28.935  0.442 |
##
## Categories
##                     Dim.1    ctr   cos2 v.test    Dim.2    ctr   cos2 v.test
## droitier          | -0.527  2.310  0.657 -2.608 | -0.125  0.540  0.037 -0.885 |
## gaucher           |  2.107  9.238  0.657  2.608 |  0.501  2.159  0.037  0.885 |
## deux              | -0.022  0.002  0.000 -0.054 | -0.940 18.971  0.734 -3.317 |
## une               |  0.022  0.002  0.000  0.054 |  0.940 18.971  0.734  3.317 |
## demi              | -0.690  1.239  0.144 -0.986 |  0.269  0.778  0.022  0.549 |
## finale            | -0.354  0.196  0.020 -0.368 |  1.525 14.977  0.370  2.260 |
## vainqueur         |  0.376  0.883  0.175  1.140 | -0.493  6.273  0.300 -2.132 |
```

Summary() affiche tour à tour :

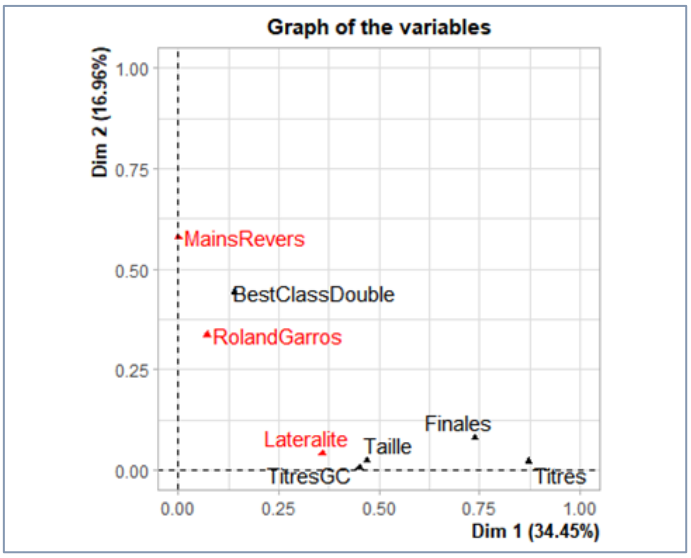
- Les variances expliquées par les 2 premiers facteurs, le cumul restitue 51.412% de l'information disponible dans les données.
- Les coordonnées des 10 premiers individus.
- Les coordonnées des variables quantitatives c.-à-d. les corrélations avec les facteurs, accompagnées des contributions et des COS2.
- Les coordonnées des modalités des variables qualitatives c.-à-d. les moyennes conditionnelles, avec les contributions, COS2 et valeurs-test.

Une série de graphiques sont automatiquement générés avec ce paramétrage. Nous observons :

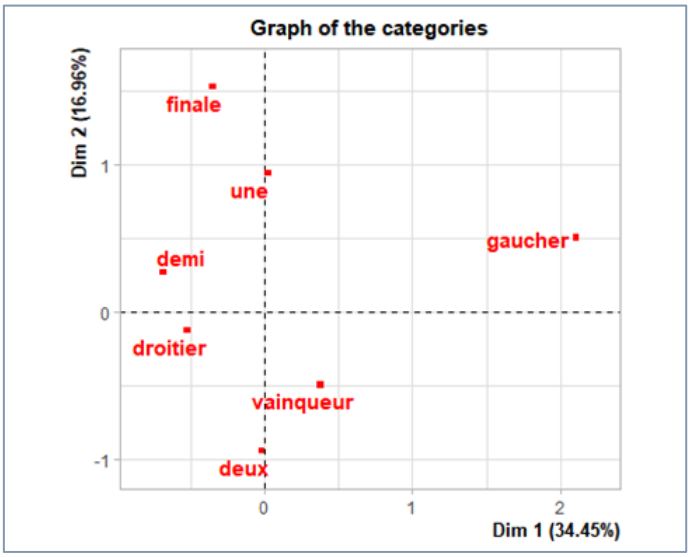
- La carte des individus. Sur le premier facteur, elle situe à droite les joueurs qui ont accumulés des titres durant une longue carrière, le plus emblématique à l'est étant Jimmy Connors, géant parmi les géants. Les moins prolixes sont à l'ouest, ils n'ont pas un palmarès à la hauteur de leur talent à cause notamment des blessures qui ont perturbé leurs carrières, avec Marat Safin, Patrick Rafter, Gustavo Kuerten. Le second facteur paraît opposer les joueurs ayant un revers à une main (au nord, Stefan Edberg, Roger Federer, John McEnroe, le meilleur d'entre tous) à ceux qui ont une petite faiblesse au poignet et ont besoin de s'aider d'une seconde main (au sud, Björn Borg, Jim Courier, André Agassi) (ok, ok ! peut-être que je ne suis pas très objectif quand il s'agit de tennis).



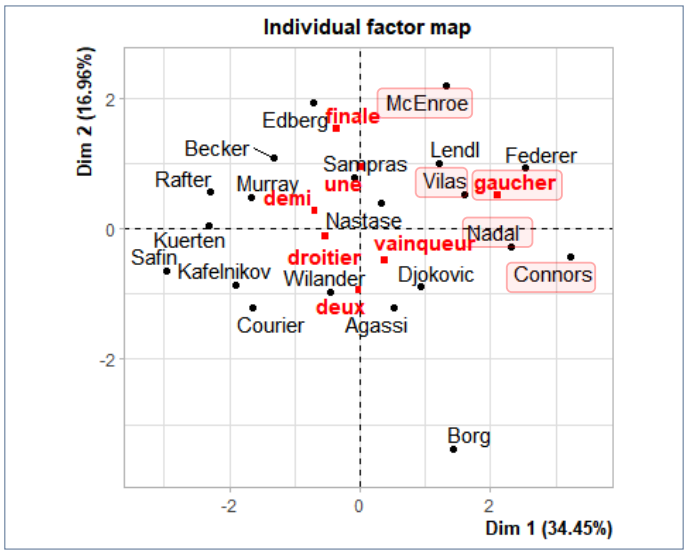
- La carte des variables c.-à-d. carré des corrélations des variables quantitatives (noir) ou carré des rapports de corrélation des qualitatives (rouge). Elle montre que « Titre » est parmi les plus déterminants sur le 1er facteur, mais on ne sait pas encore comment. « MainRevers » est la variable qui pèse le plus sur le 2nd.



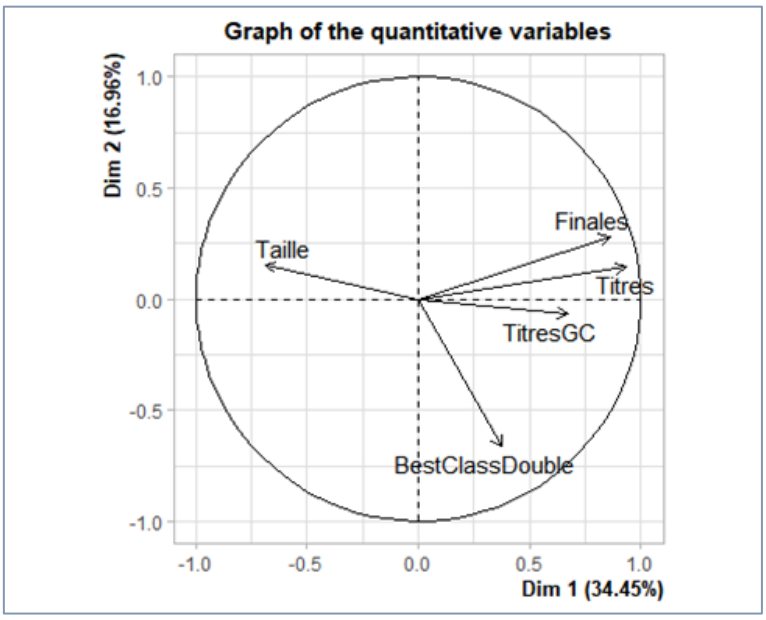
- La carte des modalités semble indiquer que la « latéralité » joue un rôle important sur le premier facteur (droitier vs. gaucher). Il faut relativiser cela lorsque nous inspectons le tableau des contributions ci-dessus, la latéralité présente une contribution de 11.5% (2.310% + 9.238%) . L'opposition « revers à une main » vs. « revers à deux mains » sur le second facteur est, elle, une caractéristique importante sur le 2nd avec une contribution cumulée de (18.971 + 18.971) = 37.9%.



- Les cartes des individus et des modalités peuvent être combinées. Dans ce cas, chaque modalité est située au barycentre des observations qui lui sont associées. Les gauchers qui ont été numéro 1 mondial ne sont pas légions, mais ils ont marqué à jamais l'histoire du tennis (McEnroe, Nadal, Connors, Vilas).



- Le cercle des corrélations enfin indique le sens de la relation des variables quantitatives avec les facteurs. Le nombre de titres est déterminant sur le 1er facteur, dans le sens croissant de l'ouest vers l'est. On se rend compte aussi sur le 2nd facteur que ceux qui ont des revers à deux mains ne se sont pas beaucoup préoccupés de leur classement en double.



Nous ne l'avons pas montré ici parce que nos données ne s'y prêtent pas. Il est très facile d'appréhender les individus supplémentaires et les variables illustratives avec « FactoMineR ». Il s'agit simplement de paramétrer la fonction à bon escient en indiquant les numéros de lignes et colonnes concernés dans le data frame.

6.4.3 Ade4

Nous utilisons la fonction **dudi.mix()** avec le package « ade4 » (COURS AFDM, page 37). L'affichage par défaut intègre les valeurs propres et pourcentages d'inertie restituées par les facteurs.

```
#importation
library(ade4)

#lancement
afdm2 <- dudi.mix(D,scannf=FALSE,nf=2)
summary(afdm2)
```

```
## Class: mix dudi
## Call: dudi.mix(df = D, scannf = FALSE, nf = 2)
##
## Total inertia: 9
##
## Eigenvalues:
##     Ax1     Ax2     Ax3     Ax4     Ax5
##  3.1009  1.5262  1.1778  1.0075  0.8404
##
## Projected inertia (%):
##     Ax1     Ax2     Ax3     Ax4     Ax5
##  34.454  16.958  13.087  11.195   9.337
##
## Cumulative projected inertia (%):
##     Ax1   Ax1:2   Ax1:3   Ax1:4   Ax1:5
##   34.45   51.41   64.50   75.69   85.03
##
## (Only 5 dimensions (out of 9) are shown)
```

Le tableau des variables regroupe les carrés des corrélations des variables quantitatives et les carrés des rapport de corrélation des qualitatives.

```
#coordonnées des variables (carrés)
print(afdm2$cr)
```

```
##                          RS1         RS2
## Taille          0.4697697732 0.023396470
## Lateralite      0.3580789624 0.041186583
## MainsRevers     0.0001525342 0.579087301
## Titres          0.8719419280 0.020469719
## Finales         0.7392389035 0.079867293
## TitresGC        0.4502596147 0.004395626
```

```
## RolandGarros    0.0718592111 0.336208885
## BestClassDouble 0.1395841978 0.441619398
```

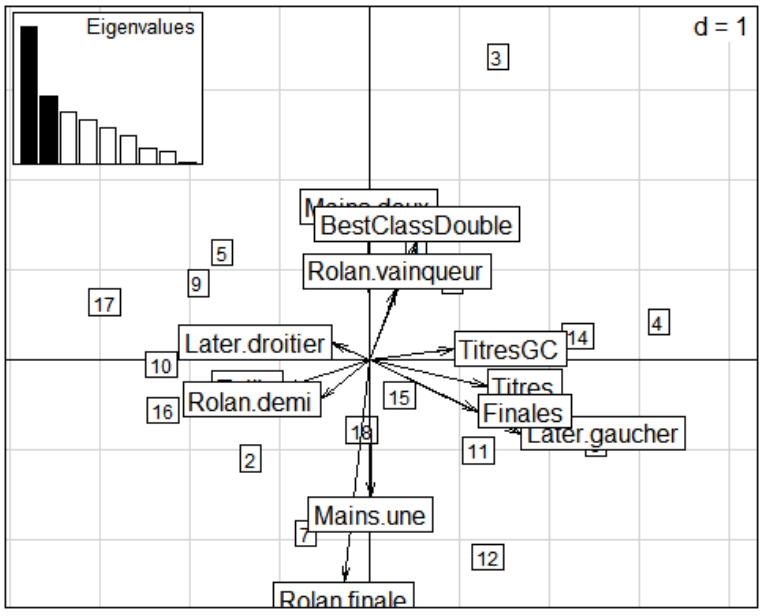
Les coordonnées des points modalités regroupe (mélange) les corrélations des variables quantitatives avec les coordonnées des modalités, lesquelles correspondent à une variante des moyennes conditionnelles ($G^*_{kh} = \frac{\mu_{kh}}{\sqrt{\lambda_h}}$).

```
#coordonnées des modalités
print(afdm2$co)
```

```
##                       Comp1       Comp2
## Taille           -0.68539753 -0.15295905
## Later.droitier   -0.29919850  0.10147239
## Later.gaucher     1.19679399 -0.40588956
## Mains.deux       -0.01235047  0.76097786
## Mains.une         0.01235047 -0.76097786
## Titres            0.93377831 -0.14307243
## Finales           0.85979003 -0.28260802
## TitresGC          0.67101387  0.06629952
## Rolan.demi       -0.39196030 -0.21800239
## Rolan.finale     -0.20119596 -1.23447640
## Rolan.vainqueur   0.21361578  0.39945343
## BestClassDouble   0.37360969  0.66454450
```

Nous retrouvons ces informations dans un graphique « biplot ».

```
#graphique
biplot(afdm2)
```



Nous l'avons répété à plusieurs reprises, il faut surtout raisonner en termes de « direction » dans ce genre de graphiques.

6.4.4 PCAmixdata

Nous utilisons **PCAmix()** avec le package « PCAmixdata » (COURS AFDM, pages 38 et 39). Il faut distinguer explicitement les variables quantitatives et qualitatives lors de l'appel de la fonction.

```
#importation
library(PCAmixdata)

#lancement
afdm3 <- PCAmix(D[c(1,4,5,6,8)],D[c(2,3,7)],ndim=2,graph=TRUE)

print(summary(afdm3))

##
## Call:
## PCAmix(X.quanti = D[c(1, 4, 5, 6, 8)], X.quali = D[c(2, 3, 7)],     ndim = 2, graph = TRUE)
##
## Method = Factor Analysis of mixed data (FAmix)
##
## Data:
##    number of observations:  20
##    number of  variables:  8
##         number of numerical variables:  5
##         number of categorical variables:  3
##
## Squared loadings :
##                  dim 1 dim 2
## Taille            0.47  0.02
## Titres            0.87  0.02
## Finales           0.74  0.08
## TitresGC          0.45  0.00
## BestClassDouble  0.14  0.44
## Lateralite        0.36  0.04
## MainsRevers       0.00  0.58
## RolandGarros      0.07  0.34
##
##


#corrélations des variables avec les facteurs
print(afdm3$quanti.cor)

##                      dim 1       dim 2
## Taille           -0.6853975  0.15295905
## Titres            0.9337783  0.14307243
## Finales           0.8597900  0.28260802
## TitresGC          0.6710139 -0.06629952
## BestClassDouble   0.3736097 -0.66454450

#coordonnées des modalités
print(afdm3$categ.coord)

##                 dim 1      dim 2
## droitier  -0.29919850 -0.1014724
## gaucher    1.19679399  0.4058896
```

```
## deux       -0.01235047 -0.7609779
## une         0.01235047  0.7609779
## demi       -0.39196030  0.2180024
## finale     -0.20119596  1.2344764
## vainqueur   0.21361578 -0.3994534
```

Les différents affichage n'appelle pas de commentaires particuliers.

Point important à souligner, contrairement aux autres outils, « PCAmixdata » intègre la rotation VARIMAX des facteurs pour l'analyse factorielle des données mixtes (COURS AFDM, page 39). Nous le savons (section 2.2), ce dispositif permet d'améliorer l'interprétation des facteurs dans certaines situations.

7 Références

7.1 Références

- Bouroche J-M., Saporta G., « L'analyse de données », Collection « Que Sais-je ? », Presses Universitaires de France (PUF), 4ème édition, 1994.
- Diday E., Lemaire J., Pouget J., Testu F., « Eléments d'Analyse de Données », Dunod, 1982.
- Husson F., Lê S., Pagès J., « Analyse de données avec R », Collection « Pratique de la Statistique », Presses Universitaires de Rennes (PUR), 2009.
- Lebart L., Morineau A., Piron M., « Statistique Exploratoire Multidimensionnelle », 3ème édition, Dunod, 2000.
- Saporta G., « Probabilités, Analyse de données et Statistique », 2ème édition, Technip, 2006.
- Tenenhaus M., « Statistique – Méthodes pour décrire, expliquer et prévoir », Dunod, 2007.
- Tufféry S., « Data Mining et Statistique Décisionnelle – L'intelligence des données », Technip, 2012.

7.2 Supports de cours

Cette section regroupe les supports de cours que j'ai rédigé.

- [COURS ACP] « Analyse en composantes principales », http://tutoriels-data-mining.blogspot.com/2013/07/analyse-en-composantes-principales.html
- [COURS MDS] « Positionnement multidimensionnel », http://tutoriels-data-mining.blogspot.com/2019/04/positionnement-multidimensionnel-diapos.html

- [COURS AFC] « Analyse factorielle des correspondances », http://tutoriels-data-mining.blogspot.com/2013/07/analyse-factorielle-des-correspondances.html
- [COURS ACM] « Analyse des correspondances multiples », http://tutoriels-data-mining.blogspot.com/2013/08/analyse-des-correspondances-multiples.html
- [COURS AFDM] « Analyse factorielle des données mixtes », http://tutoriels-data-mining.blogspot.com/2013/08/analyse-factorielle-de-donnees-mixtes.html

7.3 Tutoriels

J'ai écrit de nombreux tutoriels où il a été question de l'analyse factorielle. Les principaux sont regroupés ci-dessous. Ils abordent des aspects spécifiques de la méthode ou mettent en avant les fonctionnalités de certains outils. Pour obtenir une liste exhaustive, voir http://tutoriels-data-mining.blogspot.com/search/label/Analyse%20factorielle (52 tutoriels en juin 2020).

- (TUTO 1) « ACP avec Python », juin 2018 ; http://tutoriels-data-mining.blogspot.com/2018/06/acp-avec-python.html
- (TUTO 2) « ACP avec TANAGRA – Nouveaux outils », juin 2012 ; http://tutoriels-data-mining.blogspot.com/2012/06/acp-avec-tanagra-nouveaux-outils.html
- (TUTO 3) « ACP avec R – Détection du nombre d'axes », juin 2012 ; http://tutoriels-data-mining.blogspot.com/2012/06/acp-avec-r-detection-du-nombre-daxes.html
- (TUTO 4) « Analyse en composantes principales avec R », mai 2009 ; http://tutoriels-data-mining.blogspot.com/2009/05/analyse-en-composantes-principales-avec.html
- (TUTO 5) « ACP sous Excel avec Xnumbers », mars 2018 ; http://tutoriels-data-mining.blogspot.com/2018/03/acp-sous-excel-avec-xnumbers.html
- (TUTO 6) « ACP – Description de véhicules », décembre 2006 ; http://tutoriels-data-mining.blogspot.com/2008/03/acp-description-de-vhicules.html
- (TUTO 7) « ACP sous R – Indice KMO et Test de Barlett », mai 2012 ; http://tutoriels-data-mining.blogspot.com/2012/05/acp-sous-r-indice-kmo-et-test-de.html
- (TUTO 8) « Rotation VARIMAX en ACP », septembre 2006 ; http://tutoriels-data-mining.blogspot.com/2008/04/rotation-varimax-en-acp.html

- (TUTO 9) « Analyse en facteurs principaux », septembre 2012, http://tutoriels-data-mining.blogspot.com/2012/09/analyse-en-facteurs-principaux.html
- (TUTO 10) « ACP sur corrélations partielles (suite) », juin 2012 ; http://tutoriels-data-mining.blogspot.com/2012/06/acp-sur-correlations-partielles-suite.html
- (TUTO 11) « Classification de variables », mars 2007 ; http://tutoriels-data-mining.blogspot.com/2008/03/classification-de-variables.html
- (TUTO 12) « La page Excel'Ense de la Revue Modulad (comment réaliser une ACP et une AFC sous Excel avec une macro VBA spécialisée) », novembre 2014 ; http://tutoriels-data-mining.blogspot.com/2014/11/la-page-excelense-de-modulad.html
- (TUTO 13) « AFC – Association médias et professions », décembre 2016 ; http://tutoriels-data-mining.blogspot.com/2008/03/afc-association-mdias-et-professions.html
- (TUTO 14) « Analyse factorielle des correspondances avec R », mai 2009 ; http://tutoriels-data-mining.blogspot.com/2009/05/analyse-factorielle-des-correspondances.html
- (TUTO 15) « Analyse des correspondances – Comparaisons de logiciels », décembre 2012 ; http://tutoriels-data-mining.blogspot.com/2012/12/analyse-des-correspondances-comparaisons.html
- (TUTO 16) « ACM – Races canines », décembre 2006 ; http://tutoriels-data-mining.blogspot.com/2008/03/afcm-races-canines.html
- (TUTO 17) « ACM avec R – Package FactoMineR », mai 2009 ; http://tutoriels-data-mining.blogspot.com/2009/05/analyse-de-corresponsances-multiples.html
- (TUTO 18) « ACM – Analyse des correspondances multiples – Outils », décembre 2012 ; http://tutoriels-data-mining.blogspot.com/2012/12/analyse-des-correspondances-multiples.html
- (TUTO 19) « Analyse factorielle des données mixtes », septembre 2012 ; http://tutoriels-data-mining.blogspot.com/2012/09/analyse-factorielle-de-donnees-mixtes.html
- (TUTO 20) « Multidimensional scaling sous R », avril 2019 ; http://tutoriels-data-mining.blogspot.com/2019/04/multidimensional-scaling-sous-r.html

8 Annexes

8.1 Gestion des versions

- Version 1.0. Mise en ligne le 19 juillet 2020.

8.2 Fichier de données

Le fichier « **Data_Methodes_Factorielles.xlsx** » contient les différents jeux de données utilisées pour illustrer les techniques présentées dans cet ouvrage. Il comporte les feuilles suivantes (avec les sections concernées) :

- DATA_ACP_ACTIF (section 1.1, « Principe de l'analyse en composantes principales » ; section 1.3, « Pratique de l'ACP avec « fanalysis » sous Python » ; section 2.1, « Techniques avancées pour identifier le nombre de facteurs » ; section 2.3, « Indices de compressibilité de l'information »).
- DATA_ACP_IND_SUP (section 1.5, « Projection des individus supplémentaires »).
- DATA_ACP_VAR_ILLUS (section 1.6, « Traitement des variables illustratives »).
- BURGER_ACP (section 2.5, « Clustering de variables – VARCLUS »).
- AUTOS_AFP (section 2.6, « Analyse en facteurs principaux »).
- AFC_ETUDES (section 4.1, « Principe de l'analyse factorielle des correspondances » ; section 4.3, « Pratique de l'AFC avec « fanalysis » sous Python »).
- AFC_FOODS (section 4.5, « AFC avec d'autres outils (SAS, TANAGRA, R) »).

- ACM_CANINES (section 5.1, « Principe de l'analyse des correspondances multiples » ; section 5.3, « Pratique de l'ACM avec « fanalysis » sous Python »).
- ACM_CANINES_SUPP (section 5.3, « Pratique de l'ACM avec « fanalysis » sous Python »).
- ACM_CARS (section 5.4, « ACM avec d'autres outils (SAS, TANAGRA, R) »).
- AFDM_AUTOS (section 6.2, « Organisation des calculs » ; section 6.3, « Pratique de l'AFDM sous TANAGRA »).
- AFDM_TENNIS (section 6.4, « AFDM avec d'autres outils – R et les packages spécialisés »).
- AUTOS_MDS (section 3.2, « Positionnement multidimensionnel classique »).
- AUTOS_MDS_SUPP (section 3.3, « Traitement des individus supplémentaires »).
- AUTOS_MDS_SOURCE (section 3.4, « Positionnement multidimensionnel classique et ACP »).
- MDS_MADAGASCAR (section 3.6, « Un exemple – Distances routières entre villes de Madagascar »).

8.3 Notebooks

Plusieurs « notebooks » Jupyter (fichiers **.ipynb**) m'ont servi durant la rédaction de cet ouvrage. Ils sont regroupés dans le fichier archive « **PMFP_NOTEBOOKS.zip** »). Je les liste ici, les noms de fichier sont préfixés par le chapitre concerné :

- ACM_COURS
- ACM_FANALYSIS
- ACP_BOOK_AUTOS
- ACP_FANALYSIS
- ACP_KMO
- ACP_NB_FACTEURS_SIMUL
- ACP_PARTIAL
- AFC_COURS
- AFC_FANALYSIS
- AFDM_AUTOS
- AFP_AUTOS

- MDS_AUTOS
- MDS_MADAGASCAR

A chacun de ces projets est associé à fichier archive « .zip » (ex. ACM_COURS.zip ») qui correspond à une exportation « Markdown » (« .md ») du code et de ses sorties. Il est possible de les convertir en format Word (.docx) (ou autre format à votre convenance) avec l'outil PANDOC (https://pandoc.org/).