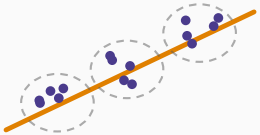
Analyse des données avancée

Chapitre 20 : Mise en œuvre sur R et Python



Jaouad Madkour

Faculté d'Économie et de Gestion de Tanger · 27 août 2026

Plan du chapitre

Lire une sortie

Rien de nouveau dans la sortie

```
Eigenvalues
                Dim.1   Dim.2   Dim.3
Variance        4.535   1.645   0.382
% of var.      56.690  20.566   4.775
Cumulative %   56.690  77.256  82.031

Variables
           Dim.1   ctr   cos2   Dim.2   ctr   cos2
prix       0.629  8.72  0.396   0.624 23.64  0.389
accueil    0.689 10.46  0.475  -0.580 20.48  0.336
```

les λ_k du chapitre 5

l'éboulis, en chiffres

la corrélation lue sur le cercle

contribution et $\cos^2$ du chapitre 8

Toute la sortie de `FactoMineR` tient dans les chapitres 5 à 8.

`Variance` est la valeur propre. `Dim.1` est la corrélation de la variable avec l'axe — ce que le cercle représente. `ctr` et `cos2` sont les deux colonnes du chapitre 8, et l'on sait laquelle se lit en ligne et laquelle en colonne.

Aucun nombre de cette sortie n'est nouveau. Le logiciel calcule vite ; il ne comprend rien.

Sur R

L'ACP avec FactoMineR

```
library(FactoMineR) ; library(factoextra)
cl <- read.csv("clients.csv", stringsAsFactors = TRUE)

acp <- PCA(cl, quanti.sup = c(3, 8), quali.sup = c(2, 4, 5, 6, 17),
           scale.unit = TRUE, ncp = 5, graph = FALSE)

acp$eig                       # 4.535 (56.69 %) puis 1.645 (20.57 %)
fviz_screeplot(acp)           # l'eboulis du chapitre 7
fviz_pca_var(acp)             # le cercle des correlations
fviz_pca_ind(acp, habillage = "frequence")
```

Les deux arguments qui font tout

`quanti.sup` et `quali.sup` désignent les colonnes **illustratives**. Tout ce qui n'y figure pas est actif.

C'est la décision du chapitre 2, réduite à deux vecteurs d'indices — et c'est la ligne qu'il faut relire trois fois avant de commenter quoi que ce soit.

Lire les aides à l'interprétation

```r
acp$var$cos2[, 1:2]          # qualite de representation des variables
acp$var$contrib[, 1:2]       # contribution : accueil 20.5 % sur l'axe 2
acp$ind$contrib[, 1]         # aucun individu au-dessus de 1.83 %

dimdesc(acp, axes = 1:2)     # description automatique des axes
```

La fonction la plus utile du paquet

`dimdesc` trie, pour chaque axe, les variables et les modalités les plus liées, avec leur corrélation.

Elle ne remplace pas la lecture du cercle — elle la vérifie. Un axe qu'on a mal nommé se repère immédiatement dans sa sortie.

```r
# CAH sur les axes factoriels, avec consolidation automatique
hc <- HCPC(acp, nb.clust = 4, consol = TRUE, graph = FALSE)

hc$desc.var$quanti           # ce qui caracterise chaque classe
hc$desc.var$test.chi2        # les variables qualitatives liees
hc$data.clust$clust          # la classe de chaque client

# Analyse discriminante
library(MASS)
afd <- lda(frequence ~ prix + choix + qualite + agencement +
             proprete + rapidite + conseil + accueil, data = cl)
table(cl$frequence, predict(afd)$class)          # 67.0 %
afdcv <- lda(frequence ~ ., data = cl[, 9:16], CV = TRUE)  # 65.3 %
```

AFC, ACM et MDS

```r
# AFC sur le tableau croise CSP x motif
tab <- table(cl$csp, cl$motif)
chisq.test(tab)              # X-squared = 110.41, df = 12
afc <- CA(tab, graph = FALSE)
afc$eig                      # 0.2267 (61.60 %) et 0.1030 (27.98 %)

# ACM sur les cinq variables qualitatives
acm <- MCA(cl[, c("csp","frequence","carte","motif","sexe")], graph = FALSE)

# MDS sur les dissimilarites entre enseignes
mds <- cmdscale(d_enseignes, k = 2, eig = TRUE)
sum(mds$eig[1:2]) / sum(mds$eig[mds$eig > 0])    # 97.34 %
```

Une ligne mérite d’être commentée

`chisq.test(tab)` puis `CA(tab)` : le test et l’analyse, l’un après l’autre, sur le même objet.

Le premier dit qu’il y a une liaison ; le second dit laquelle. Les deux lignes ensemble résument tout le chapitre 9.

Sur Python

Les équivalents

```python
import pandas as pd, numpy as np, prince
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

cl = pd.read_csv("clients.csv")
notes = ["prix","choix","qualite","agencement",
         "proprete","rapidite","conseil","accueil"]

acp = prince.PCA(n_components=5).fit(cl[notes])
acp.eigenvalues_summary        # 56.69 % puis 20.57 %
F = acp.transform(cl[notes])   # les coordonnees des individus
```

```
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

Z = linkage(F.iloc[:, :2], method="ward")
dendrogram(Z, no_labels=True)
g = fcluster(Z, t=4, criterion="maxclust")            # 4 classes
```

Consolidation et discrimination

```
# k-means partant des centres de la CAH
centres = np.array([F.iloc[:, :2][g == k].mean() for k in range(1, 5)])
km = KMeans(n_clusters=4, init=centres, n_init=1).fit(F.iloc[:, :2])

afd = LDA().fit(StandardScaler().fit_transform(cl[notes]), cl.frequence)
```

La correspondance des commandes

R et Python, côte à côte

R	Python
`PCA` (FactoMineR)	`prince.PCA`
`CA`	`prince.CA`
`MCA`	`prince.MCA`
`hclust(..., "ward.D2")`	`scipy.linkage(..., "ward")`
`kmeans`	`sklearn.KMeans`
`lda` (MASS)	`LinearDiscriminantAnalysis`
`cmdscale`	`sklearn.manifold.MDS`

Lequel choisir

L'avantage de Python est ailleurs

Il commence dès que les données sont volumineuses, ou qu'il faut **industrialiser** le calcul du score — le recalculer chaque nuit sur la base clients.

Pour une étude, R. Pour un traitement récurrent, Python.

Un avantage réel de R sur ce terrain

`FactoMineR` sort d'emblée contributions, $\cos^2$ et éléments illustratifs. En Python, il faut souvent les recalculer.

Ce n'est pas un détail : ce sont exactement les indicateurs sans lesquels un plan factoriel ne s'interprète pas.

Les pièges

Ceux que le logiciel ne signale jamais

Le piège	**Ce qu'il faut faire**
Interpréter une flèche courte	regarder le $\cos^2$ avant de commenter
Lire un plan sans regarder λ_3	vérifier ce que le plan laisse dehors
Croire que 15 % est faible en ACM	comparer à $1/p$, pas à 100 %
Rapprocher une ligne d'une colonne en ACP	seule l'AFC autorise cette lecture
Prendre une classe pour une population	c'est une hypothèse, pas un fait établi
Nommer un segment avant de l'avoir décrit	sortir d'abord le tableau des moyennes
Juger l'AFD sur son taux d'apprentissage	publier le taux en validation croisée

Six de ces sept erreurs consistent à faire dire aux données plus qu'elles ne disent.

L'analyse des données ne teste rien, ne prouve rien, ne prédit rien à elle seule. Elle *montre*. **Sa force est de faire voir ; sa faiblesse est qu'on la croit trop vite.**

La question à se poser devant tout plan factoriel : *quelle part de l'information ai-je sous les yeux, et qu'est-ce que je ne vois pas ?*

Ce que le logiciel ne fait pas

Il ne choisit pas les variables actives

`PCA(cl)` sans argument met **tout** en actif, y compris le panier moyen et l'âge — et sort un résultat parfaitement propre, parfaitement faux.

Aucun message d'erreur ne viendra.

Il ne dit pas si les classes existent

`HCPC(acp, nb.clust = 4)` produira quatre classes sur trois cents points tirés au hasard, avec un dendrogramme d'apparence convaincante.

C'est au lecteur du saut d'inertie de trancher, pas à la fonction.

Ce qu’il faut retenir

Le résumé du chapitre

Six points

1. PCA, CA, MCA, HCPC, `lda`, `cmdscale` : six fonctions pour les six méthodes.
2. `quanti.sup` et `quali.sup` traduisent la décision **active** ou **illustrative**.
3. `dimdesc` vérifie l'interprétation des axes ; elle ne la remplace pas.
4. `HCPC(..., consol = TRUE)` enchaîne CAH et k-means en une instruction.
5. R sort les **contributions** et $\cos^2$ d'emblée ; Python demande de les recalculer.
6. Le logiciel **ne choisit pas les variables actives** et **ne dit pas si les classes existent**.

La fin du parcours

Cinq méthodes, une seule diagonalisation, un seul fichier de trois cents clients.

Vous avez décrit leurs opinions, croisé leurs profils, formé leurs segments, construit une règle de ciblage et positionné huit enseignes. **Aucun de ces résultats n'était visible dans le tableau de départ** — et c'est précisément l'objet de la discipline.