Analyse en composantes principales avec python

1. Mise en œuvre pratique du programme (exemple « Mini_Sem »)
2. Mise en œuvre pratique du programme (exemple « Sem300 »)
3. Contenu du fichier source téléchargeable : acomp_dtmF.py

Les fichiers d'exemples sont des fichiers de données sémiométriques (notes de 1 à 7 données à des mots par des individus selon que ces mots sont désagréables ou agréables). Dans les exemples en vraies grandeurs, il y a 210 mots et parfois plusieurs milliers d'individus.

Les ouvrages relatifs à la sémiométrie sont en libre chargement sur ce site (www.dtmvic.com).
http://www.dtmvic.com/Semiometrie.html (« La sémiométrie », Dunod, version française de 2003)
http://www.dtmvic.com/doc/Semio_2014_Cover_354x244.pdf (« The semiometric challenge » , L2C, couverture de la version anglaise de 2014)
http://www.dtmvic.com/doc/Semio_2014_format_169x244.pdf (version anglaise de 2014)

Les deux fichiers d'exemple (Mini_sem et Semio_300) comportent chacun trois volets :

Pour l'exemple **Mini_Sem** (modèle réduit, pédagogique) :

1) Individus et variables actifs (Mini_Sem.txt) (7 mots notés par 12 individus).

2) Variables supplémentaires (Mini_Sem_vsup.txt) (3 mots supplémentaires notés par les mêmes 12 individus, + le genre (sexe) du répondant, variable nominale)

3) Individus supplémentaires (Mini_Sem_isup.txt). (4 nouveaux individus ayant noté les mêmes mots actifs).

C'est avec cet exemple « Modèle réduit » qu'il convient d'aborder le programme python (dont le listage complet figure aussi ci-dessous en section 3). Le programme (code python) figure dans le fichier : acomp_dtmF.py.

L'exemple **Sem300,** plus réaliste (en tout 300 individus et 70 mots à noter) est traité en section 2.

1) Individus actifs et variables actives (Sem300.txt) (63 mots notés par 296 individus).

2) Variables supplémentaires (Sem300_vsup.txt) (7 mots supplémentaires notés par les mêmes 296 individus (variables numériques), + 6 variables nominales : classes d'âge, niveau d'éducation, sexe croisé avec l'âge, avec le niveau d'éducation, âge croisé avec éducation, et genre (sexe) du répondant)

3) Individus supplémentaires (Sem300_isup.txt). (4 nouveaux individus ayant noté les mêmes mots actifs).

1. Mise en œuvre pratique du programme (exemple Mini_Sem).

Il convient de télécharger les trois fichiers de données précédents ainsi que fichier code acomp_dtmF.py dans un dossier de travail. Le fichier acomp_dtmF.py figure en annexe de cette note, mais constitue aussi un fichier séparé téléchargeable comme les données.

Contenu des trois fichiers de données téléchargeables (exemple réduit Mini_Sem):

Mini_Sem.txt

```
arbre,cadeau,danger,morale,orage,politesse,sensuel
R01, 7,  4,  2,  2 , 3 , 1 , 6
R02, 6 , 3 , 1 , 2 , 4 , 1 , 7
R03, 4 , 5 , 3 , 4 , 3 , 4 , 3
R04, 5 , 5 , 1 , 7 , 2 , 7 , 1
R05, 4 , 5 , 2 , 7 , 1 , 6 , 2
R06, 5 , 7 , 1 , 5 , 2 , 6 , 5
R07, 4 , 2 , 1 , 3 , 5 , 3 , 6
R08, 4 , 1 , 5 , 4 , 5 , 4 , 7
R09, 6 , 6 , 2 , 4 , 7 , 5 , 5
R10, 6 , 6 , 3 , 5 , 3 , 6 , 6
R11, 7 , 7 , 6 , 7 , 7 , 6 , 7
R12, 2 , 2 , 1 , 2 , 1 , 3 , 2
```

Mini_Sem_vsup.txt

```
fleur,gateau,risque, genre
R01, 7,  4,  1, Fem
R02, 7 , 4 , 1, Fem
R03, 4 , 5 , 3, Fem
R04, 5 , 6 , 2, Mal
R05, 4 , 5 , 2, Fem
R06, 5 , 6 , 1, Fem
R07, 4 , 2 , 1, Fem
R08, 4 , 1 , 4, Mal
R09, 7 , 6 , 2, Fem
R10, 7 , 6 , 2, Fem
R11, 4 , 6 , 6, Mal
R12, 2 , 2 , 2, Mal
```

Mini_sem_isup.txt

```
arbre,cadeau,danger,morale,orage,politesse,sensuel
R13, 6,  3,  2,  3 , 3 , 1 , 7
R14, 7 , 3 , 1 , 2 , 4 , 1 , 7
R15, 5 , 4 , 2 , 4 , 3 , 4 , 2
R16, 7,  6 , 1 , 7 , 1 , 7 , 1
```

1. Instructions préliminaires

Une fois le fichier code acomp_dtmF.py copié dans votre dossier de travail (appelé ici « mydirectory ») avec les trois fichiers de données précédents, il suffit de taper une à une dans l'interface python (comme IDLE par exemple) les 4 instructions suivantes pour donner accès au programme et aux données :

Toutes les instructions qui suivent figurent en commentaires dans le fichier acomp_dtmF.py, et peuvent donc être simplement copiées à partir de ce ficbier.

```
import os
os.chdir("c:/mydirectory")
import acomp_dtmF
from acomp_dtmF import *
```

Il suffit de copier ces lignes directement à partir du fichier `acomp_dtmF.py` que l'on aura ouvert dans un éditeur de texte libre (tel que Notepad ++, qui permet des éditions claires en python).

Ici, le dossier "mydirectory" est directement dans la racine "c:/". (A adapter pour chaque utilisateur, qui remplacera « mydirectory » par le chemin menant à son dossier de travail).

L'importation ci-dessus de acomp_dtmF.py entraîne automatiquement le chargement des bibliothèques pandas, numpy et matplotlib.

2. Calculs de base de l'ACP : fonction ACP_base()

Pour l'exemple "Mini_Sem" , on écrira dans l'interface python les 3 noms d'accès *sur une seule ligne* :

```
lane, lane_sup, lane_var_sup =
    'Mini_Sem.txt', 'Mini_Sem_isup.txt', Mini_Sem_vsup.txt'
```

Pour la lecture des données et les calculs de base de l'ACP, on écrira :

```
X, c_indiv, c_var, vecp, moy, ecinv = ACP_base (lane)
```

On obtient les impressions et graphiques suivants (figure 1) sur l'interface.

```
The coordinates are in the created file; ACP_file_Mini_Sem.txt

Eigenvalues
[2.76445231 2.50395915 0.94918253 0.35160848 0.20423643 0.18497933
 0.04158177]

Coordinates of variables

       ident   c_var_1   c_var_2   c_var_3
0      arbre -0.628178  0.492948  0.536678
1     cadeau -0.781022 -0.282650  0.473224
2     danger -0.624935  0.370061 -0.587579
3     morale -0.774409 -0.547625 -0.175524
4      orage -0.477985  0.718025 -0.176605
5  politesse -0.698772 -0.640137 -0.172968
6    sensuel -0.229877  0.904929  0.007299

 complete coord. and coord. of indiv. are in the created file:
ACP_file_Mini_Sem.txt
```

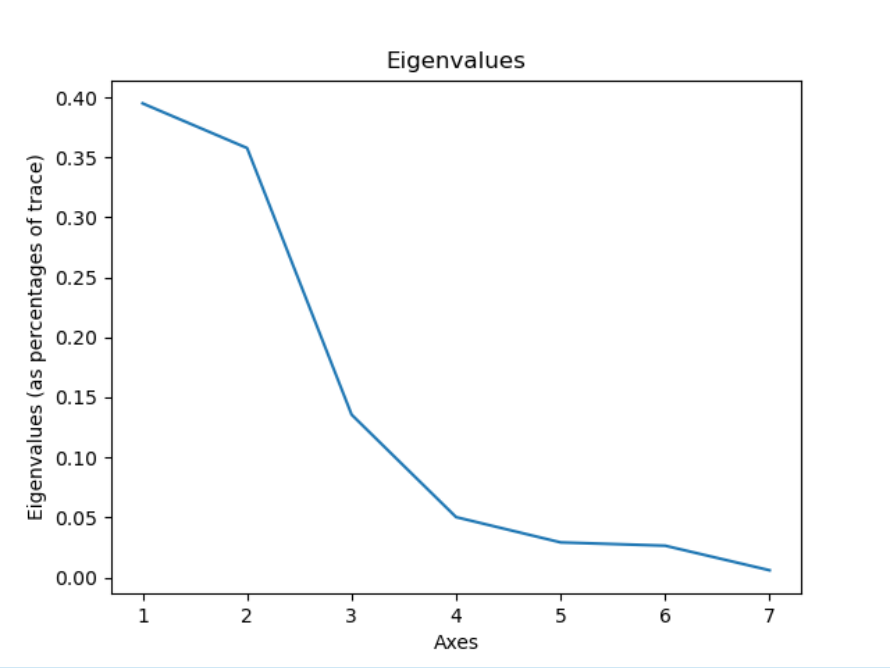

Figure 1. Série des valeurs propres

3) Premières visualisations

Selection d'une paire d'axes pour les visualisations.
Pour la sémiométrie, l'axe 1 (facteur de taille, pour lequel toutes les variables sont simultanément positives ou négatives) n'est pas pris en compte.
On retiendra ici les visualisations sur les axes 2 et 3.

ax_h, ax_v = 2, 3

L'appel de la fonction grafact( ) produit le plan factoriels éléments actifs (indiv. + variables)

grafact (X, c_indiv, c_var, ax_h, ax_v)

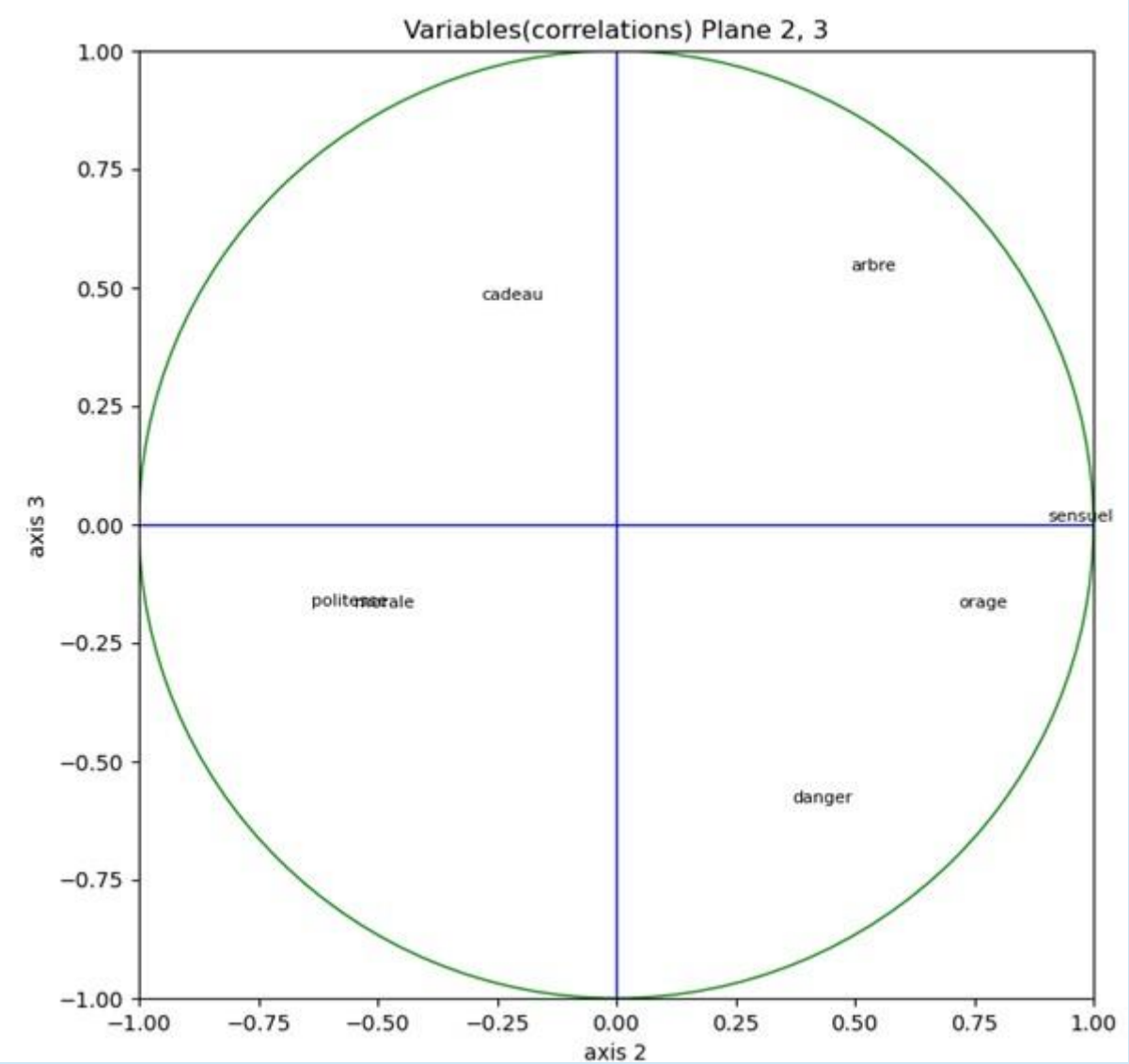

Figure 2. Position des variables dans le plan 2, 3

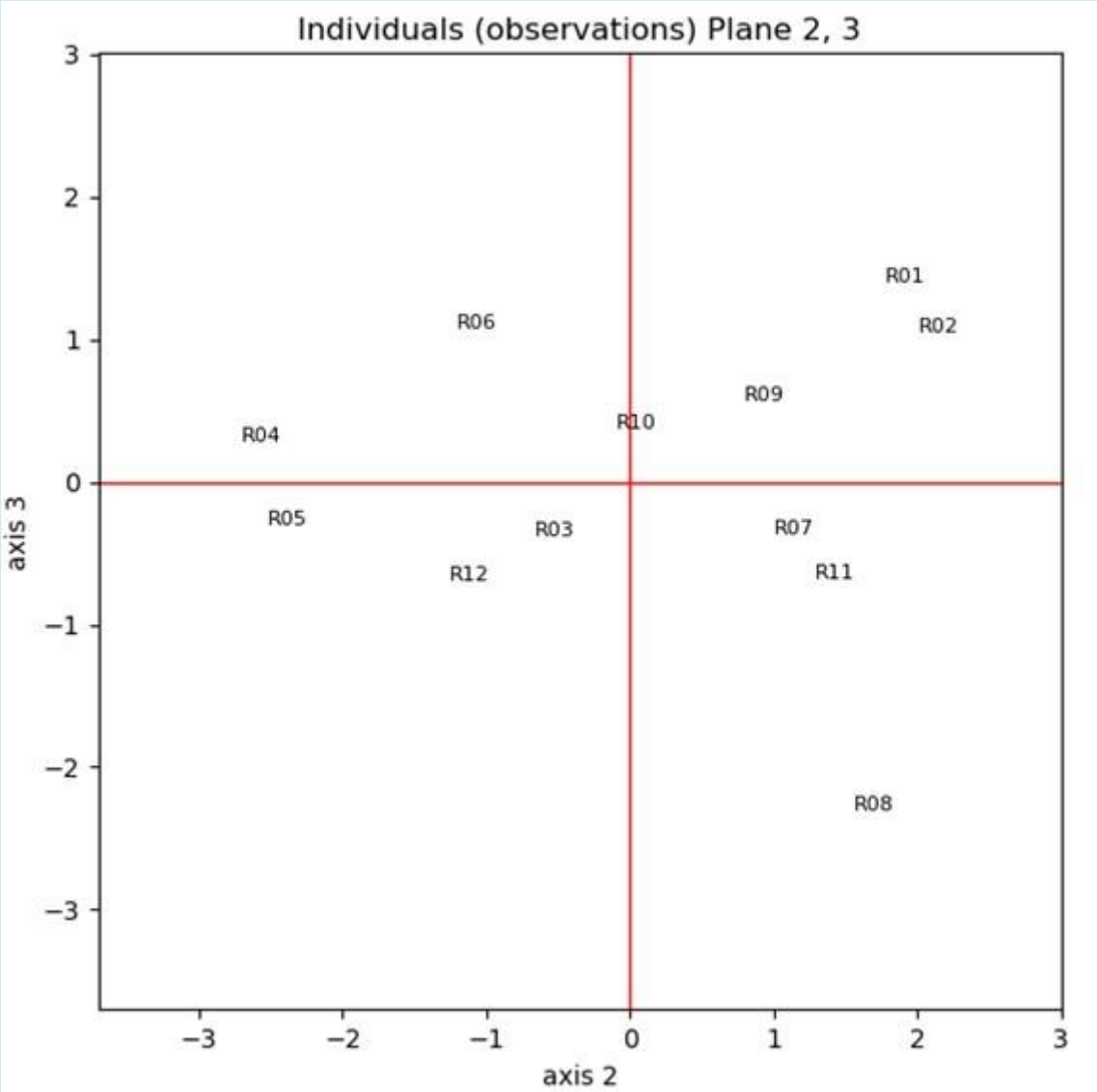

Figure 3. Position des individus dans le plan 2, 3

4) Individus supplémentaires

Tous les individus ou observations (lignes du tableau Mini_Sem_isup) sont représentés par l'appel de la fonction ACP_isup ( ).

ACP_isup(X, c_indiv, lane_sup,moy, ecinv, vecp, ax_h, ax_v)

Ils sont repérables sur le plan (2, 3) de la figure 4 par une police légèrement plus grande et une couleur bleue.

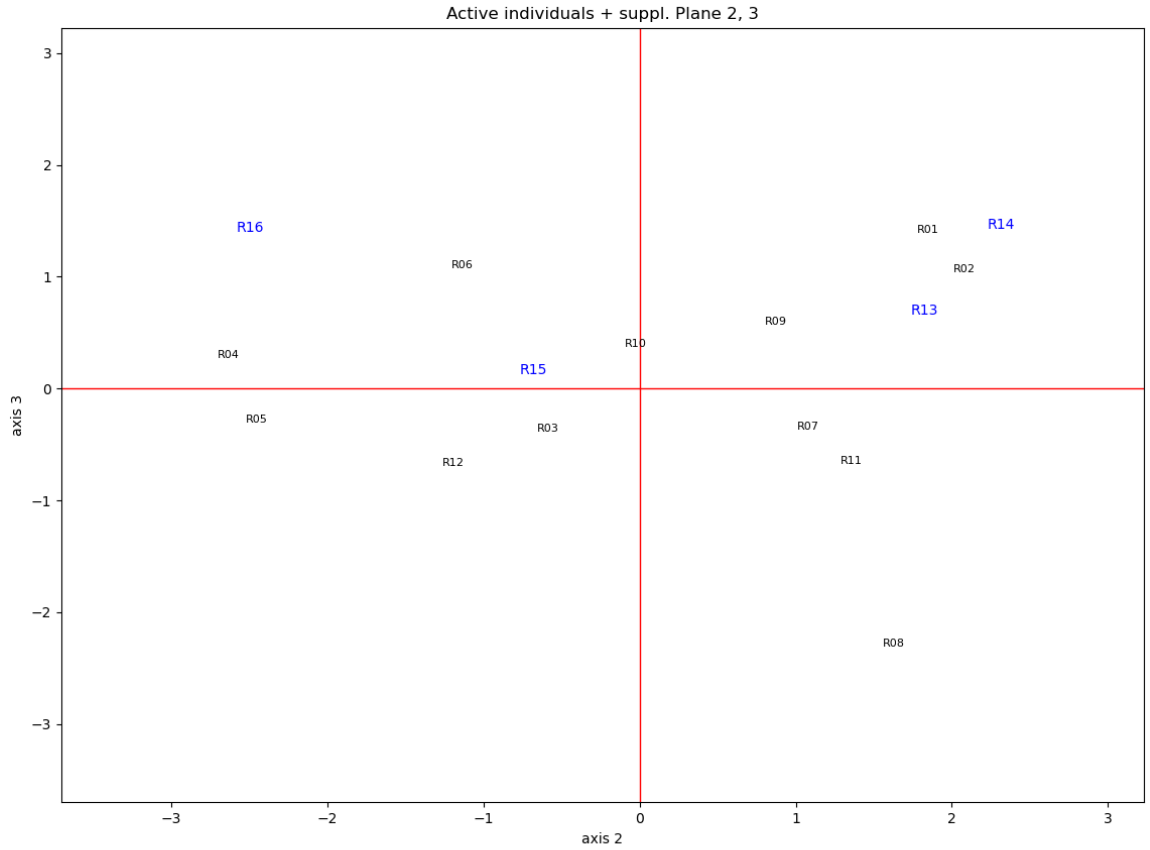

Figure 4. Position des individus supplémentaires (R13 à R16) parmi les individus actifs dans le plan (2, 3)

5) Variables numériques supplémentaires

Pour les variables, il faut distinguer entre variables numériques et variables nominales.
Les 3 variables numériques sont en tête. On écrira : vsup_lim = 3

vsup_lim = 3 #(Mini_Sem)

Les 3 premières variables supplémentaires numériques sont prises en compte par la fonction ACP_vsup( ).

df_vsup = ACP_vsup(X, c_indiv, c_var, lane_var_sup,ax_h, ax_v, vsup_lim)

On obtient une impression (sommaire) des coordonnées des variables supplémentaires sur les axes.

```
[[-0.24064513  0.40250551  0.69716752 -0.00209384  0.13542786 -0.16249853
   0.30224307]
 [-0.71201974 -0.33518815  0.56650136  0.02435166 -0.15573326  0.05657702
   0.01056208]
 [-0.60975898  0.1493028  -0.65328061 -0.15017663 -0.30155271 -0.00774477
  -0.05920361]]
```

La figure 5 représente les positions des variables supplémentaires.

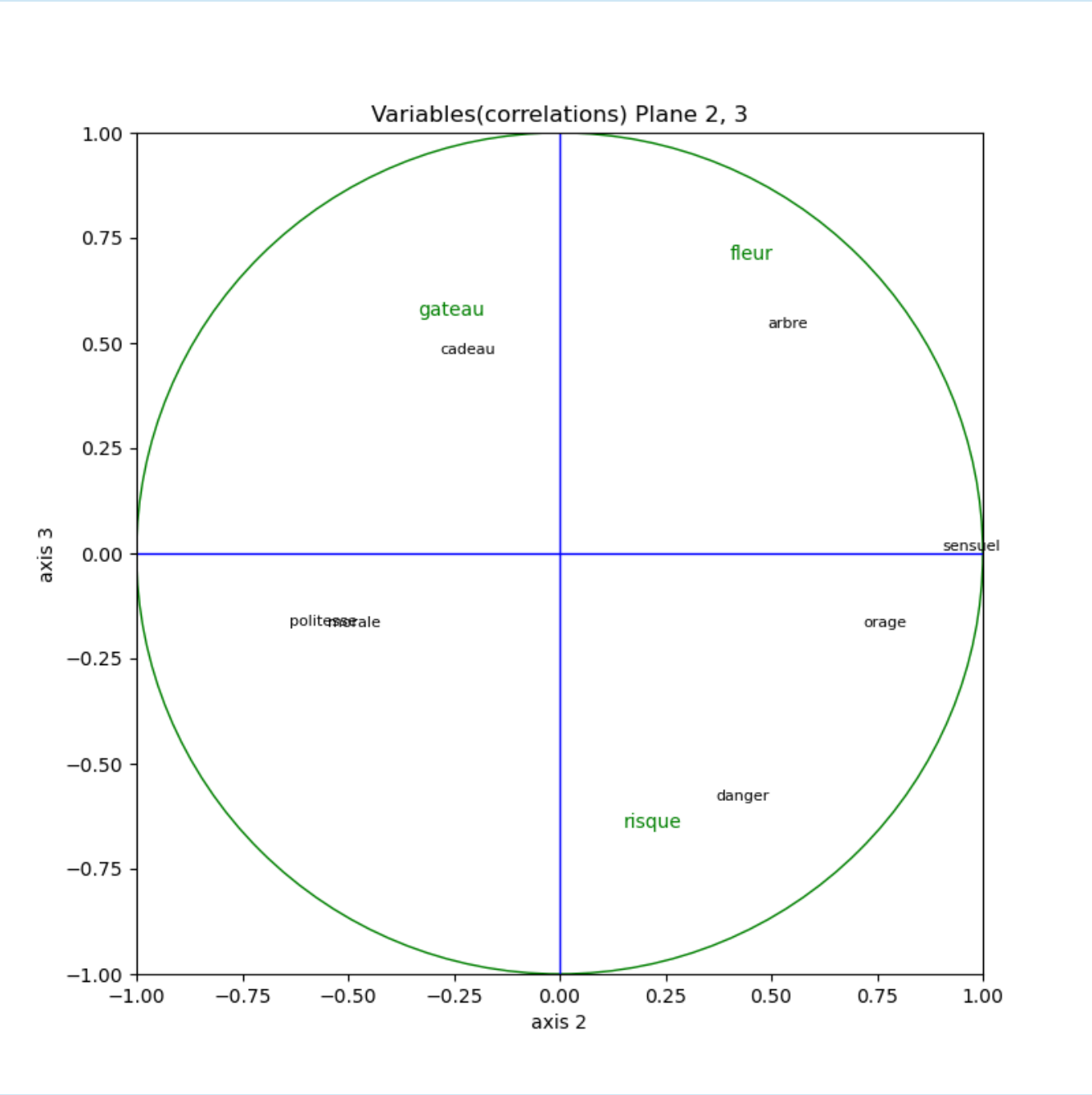

Figure 5. Position des variables supplémentaires (risque, gâteau, fleur) parmi les variables actives dans le plan 2, 3

6) Variable nominale supplémentaire

Elles sont choisies une par une car elles donnent lieu à une visualisation qui comprend les individus

vsup_nom = 4 #(Mini_Sem)

La var nominale vsup_nom est sélectionnée. La représentation se fait par la fonction ACP_nom( ).

ACP_nom(X, c_indiv, df_vsup, ax_h, ax_v, vsup_nom)

On obtient finalement la figure 6.

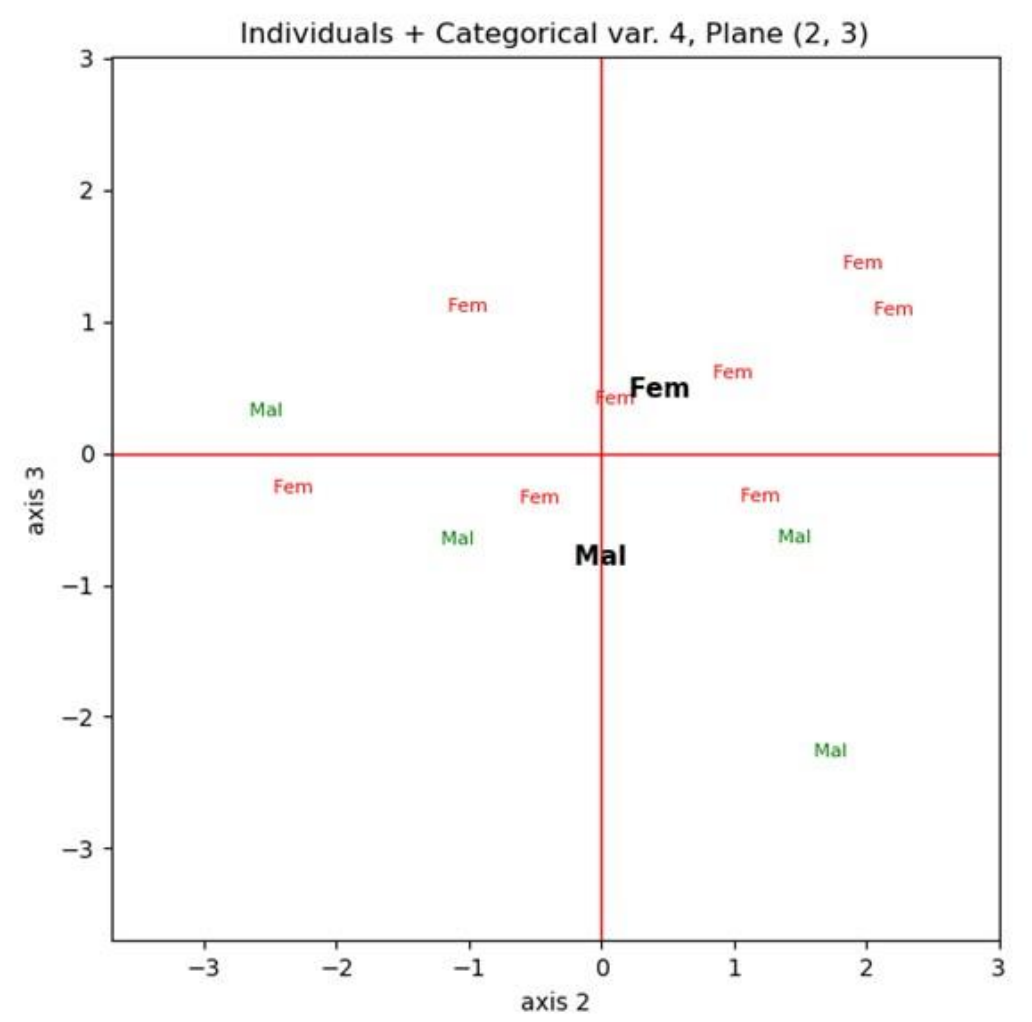

Figure 6. Position des catégories de la variable nominale supplémentaire (Female et Male) parmi les individus actifs (colorés et identifiés par leur catégorie) dans le plan 2, 3.
Les catégories supplémentaire Female et Male (noir, gras) sont au points moyens des individus concernés.

Remarque :

Ces 5 appels de fonctions auraient pu être remplacés par l'unique appel de la fonction synthétique ACP( ) figurant à la fin du code :

```
ACP (lane, lane_sup, lane_var_sup, ax_h, ax_v, vsup_lim, vsup_nom)
```

Il faut bien sûr définir auparavant tous les paramètres qui figurent dans la parenthèse ci-dessus.
Mais on peut vouloir représenter d'autres plans factoriels, d'autres variables nominales supplémentaires, ou sélectionner les variables numériques supplémentaires... l'appel par fonctions séparées est plus souple et donne plus d'information sur le programme.

2. Mise en œuvre pratique du programme avec l'exemple « Sem300 »

Le second exemple, plus réaliste, « Semio_300 », est mis en œuvre de façon similaire.
Les instructions à modifier figurent dans les commentaires du fichier acomp_dtmF.py.

Il convient de télécharger les trois fichiers de données Sem300.txt, Sem300_isup.txt et Sem300_vsup, ainsi que fichier code acomp_dtmF.py dans un dossier de travail. Le fichier acomp_dtmF.py figure en section 3 de cette note, mais constitue aussi un fichier séparé téléchargeable comme les données.

1. Instructions préliminaires

Une fois le fichier code acomp_dtmF.py copié dans votre dossier de travail (appelé ici « mydirectory2 ») avec les trois fichiers de données, il suffit de taper une à une dans l'interface python (comme IDLE par exemple) les 4 instructions suivantes pour donner accès au programme et aux données :

Toutes les instructions qui suivent figurent en commentaires dans le fichier acomp_dtmF.py, et peuvent donc être simplement copiées à partir de ce ficbier.

Les instructions en vert sont identiques à celles utilisées pour l'exemple pédagogique « Mini_Sem » présenté plus haut. Seules les 3 instructions en bleu sont différentes.

```
import os
os.chdir("c:/mydirectory2")
import acomp_dtmF
from acomp_dtmF import *
```

Il suffit de copier ces lignes directement à partir du fichier acomp_dtmF.py que l'on aura ouvert dans un éditeur de texte libre (tel que Notepad ++, qui permet des éditions claires en python).

Ici, le dossier "mydirectory2" est directement dans la racine "c:/". (A adapter pour chaque utilisateur, qui remplacera « mydirectory2 » par le chemin menant à son dossier de travail).
L'importation ci-dessus de acomp_dtmF.py entraîne automatiquement le chargement des bibliothèques pandas, numpy et matplotlib.

2. Calculs de base de l'ACP : fonction ACP_base()

Pour l'exemple "Semio300" , on écrira dans l'interface python les 3 nouveaux noms d'accès *sur une seule ligne* :

```
lane, lane_sup, lane_var_sup  =
    'Sem300.txt', 'Sem300_isup.txt', 'Sem300_vsup.txt'
```

Pour la lecture des données et les calculs de base de l'ACP, on écrira encore :

```
X, c_indiv, c_var, vecp, moy, ecinv = ACP_base (lane)
```

On obtient les impressions (abrégées) et graphiques suivants (figure A.1) sur l'interface.

```
The coordinates are in the created file; ACP_file_Sem300.txt

 Eigenvalues

[6.96587114 4.33649481 2.88336287 2.83527565 2.14122017 1.82655785
 1.71244419 1.6266963  1.58221059 1.46803872 1.38717504 1.3599013
 1.30087134 1.19921409 1.17890621 1.16453523 1.14062562 1.10437591
. . . . . .
 0.50642673 0.47454463 0.46867903 0.45126445 0.43266203 0.42330899
 0.41813093 0.40180127 0.38334004 0.36858141 0.35585478 0.33957725
 0.31372911 0.30050549 0.29130496 0.27990392 0.27892462 0.26183643
 0.24529347 0.22105591 0.20658516]

Coordinates of variables
         ident   c_var_1   c_var_2   c_var_3
0     ABSOLUTE -0.158874  0.147754 -0.363015
1    TO_ADMIRE -0.480182  0.031286  0.135131
2         SOUL -0.334792  0.359394 -0.117727
3       ANIMAL -0.238617 -0.181069  0.082491
4       ARMOUR  0.003875  0.007344 -0.455864
..         ...       ...       ...       ...
58    TO_DREAM -0.365864 -0.386315  0.118706
59       RIGID -0.015461  0.223013 -0.227371
60    TO_BREAK  0.316239  0.179164 -0.398447
61      SACRED -0.333013  0.490422 -0.017656
62     SCIENCE -0.312540  0.081631 -0.065530

[63 rows x 4 columns]

 complete coord. and coord. of indiv. are in the created file: ACP_file_Sem300.txt
```

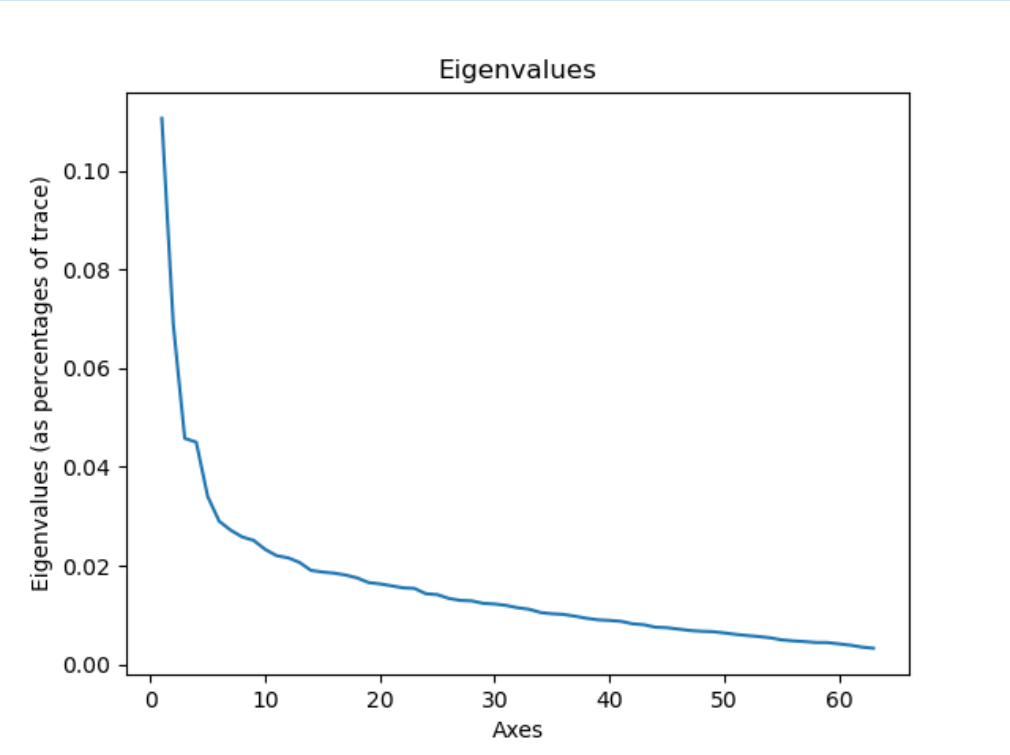

Figure A.1. Série des 63 valeurs propres

3) Premières visualisations

Selection d'une paire d'axes pour les visualisations.
Pour la sémiométrie, l'axe 1 (facteur de taille) n'est pas pris en compte.
On retiendra les axes 2 et 3.

ax_h, ax_v = 2, 3

L'appel de la fonction grafact( ) produit le plan factoriels éléments actifs (indiv. + variables)

grafact (X, c_indiv, c_var, ax_h, ax_v)

L'interface IDLE permet de procéder, pour plus de lisibilité, à un zoom d'un rectangle central à l'intérieur du cercle de corrélation.

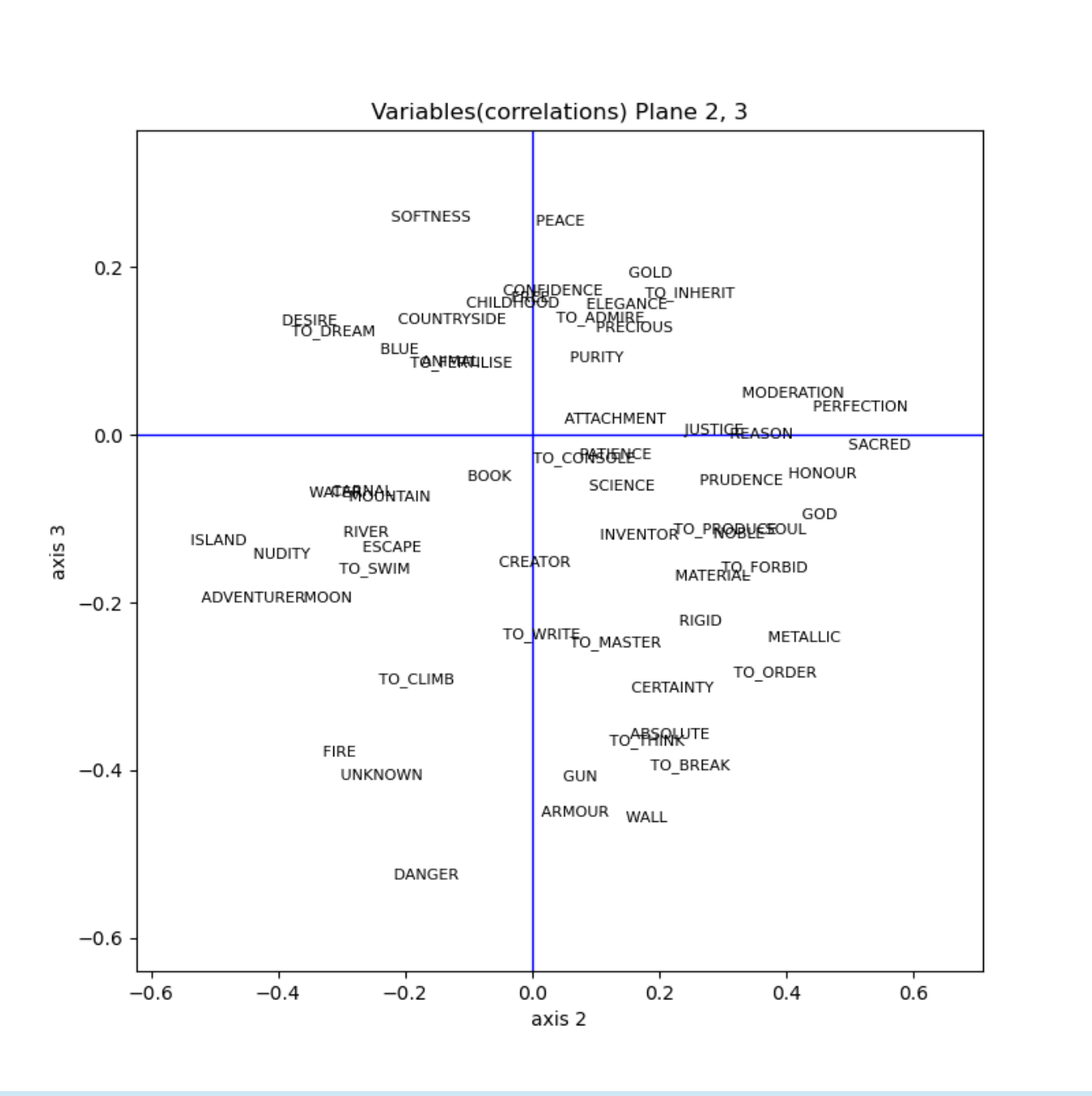

Figure A.2. Position des 63 variables actives dans le plan 2, 3 (zoom)

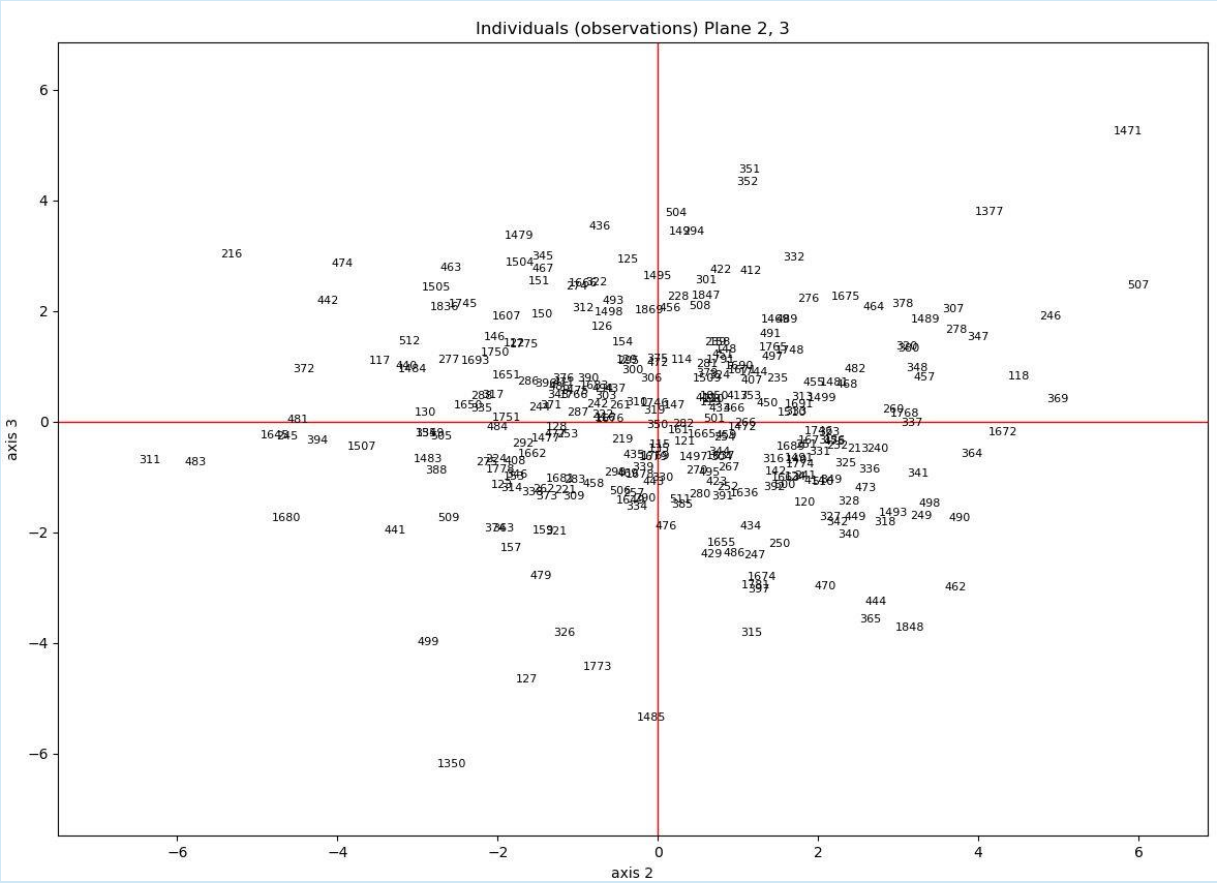

Figure A.3. Position des 296 individus dans le plan 2, 3

4) Individus supplémentaires

Tous les individus (lignes) du tableau Mini_Sem_isup sont représentés par la fonction ACP_isup( ).

ACP_isup(X, c_indiv, lane_sup,moy, ecinv, vecp, ax_h, ax_v)

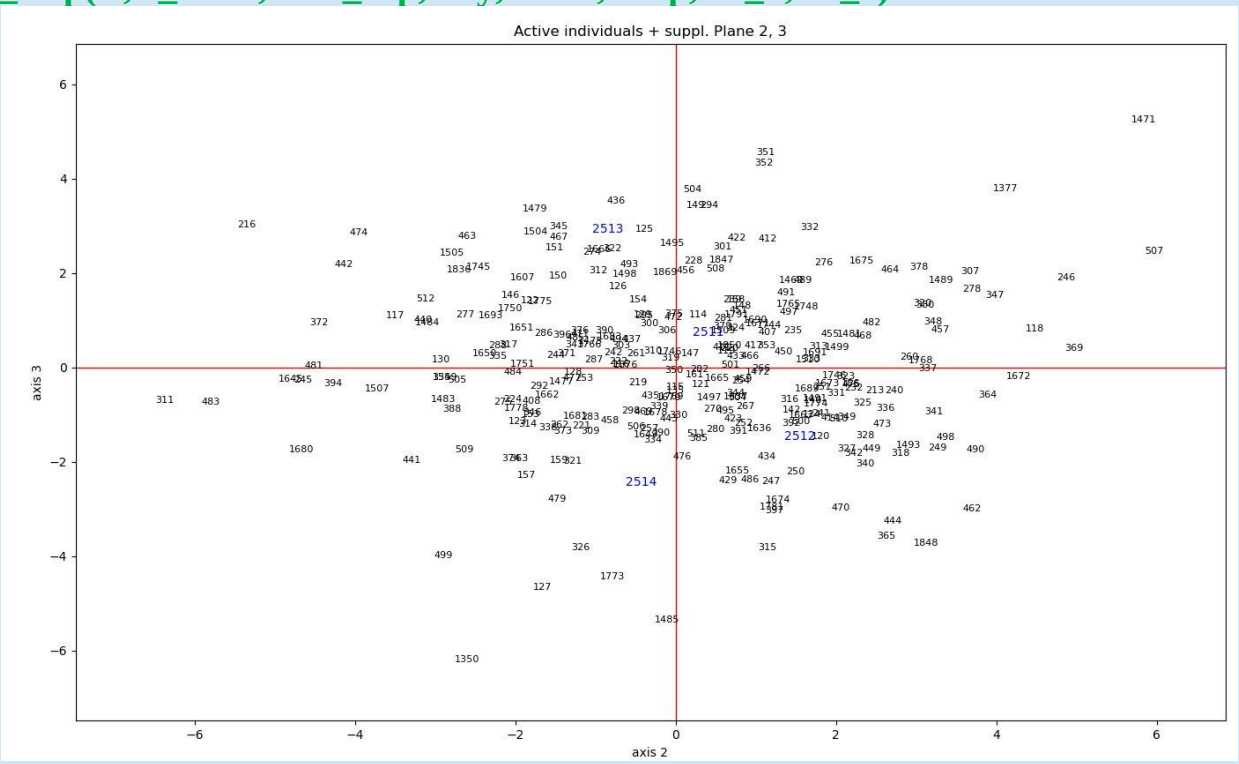

Figure A.4. Position des 4 individus supplémentaires (identificateurs de `2511` à `2514` en bleu) parmi les individus actifs dans le même plan 2, 3. On pouvait zoomer en utilisant la loupe de l'interface IDLE.

5) Variables numériques supplémentaires

Pour les variables, il faut distinguer entre variables numériques et variables nominales.
Les 7 variables numériques sont en tête. On écrira : vsup_lim = 7
(Rappelons que pour ces exemples, les variables numériques sont des notes attribuées à des mots, alors que les catégories supplémentaires sont de catégories de répondants à l'enquête sémiométrique).

vsup_lim = 7 # (Sem300)

Les 7 premières variables supplémentaires numériques sont prises en compte par la fonction ACP_vsup( ).

df_vsup = ACP_vsup(X, c_indiv, c_var, lane_var_sup,ax_h, ax_v, vsup_lim)

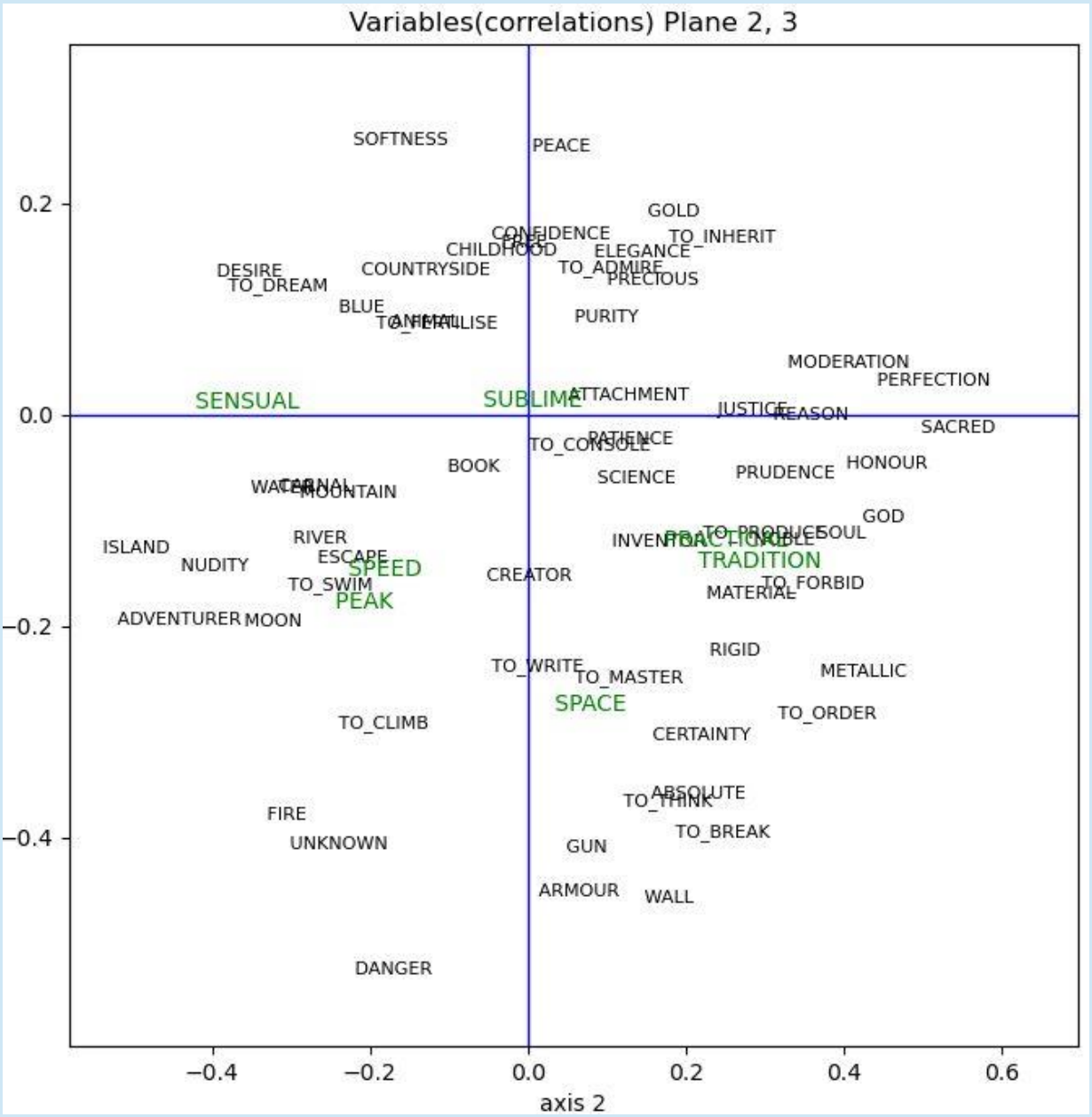

Figure A.5. Position des 7 variables numériques supplémentaires (`PEAK, PRACTICAL, SENSUAL, SPACE, SPEED, SUBLIME, TRADITION`) parmi les variables actives du plan 2, 3 (zoom).

6) Variable nominale supplémentaire

Elles sont choisies une par une car elles donnent lieu à une visualisation sur les individus

vsup_nom = 12 # (Sem300)

La var nominale vsup_nom est sélectionnée. La variable nominale 12 comprend 9 catégories (croisement de 3 classes d'âge [–30, 30-55, +55] avec 3 niveaux d'éducation [`low, medium, high`].
La représentation se fait toujours par la fonction ACP_nom( ).

ACP_nom(X, c_indiv, df_vsup, ax_h, ax_v, vsup_nom)

On obtient finalement la figure A.6 (également obtenue avec le zoom de IDLE)

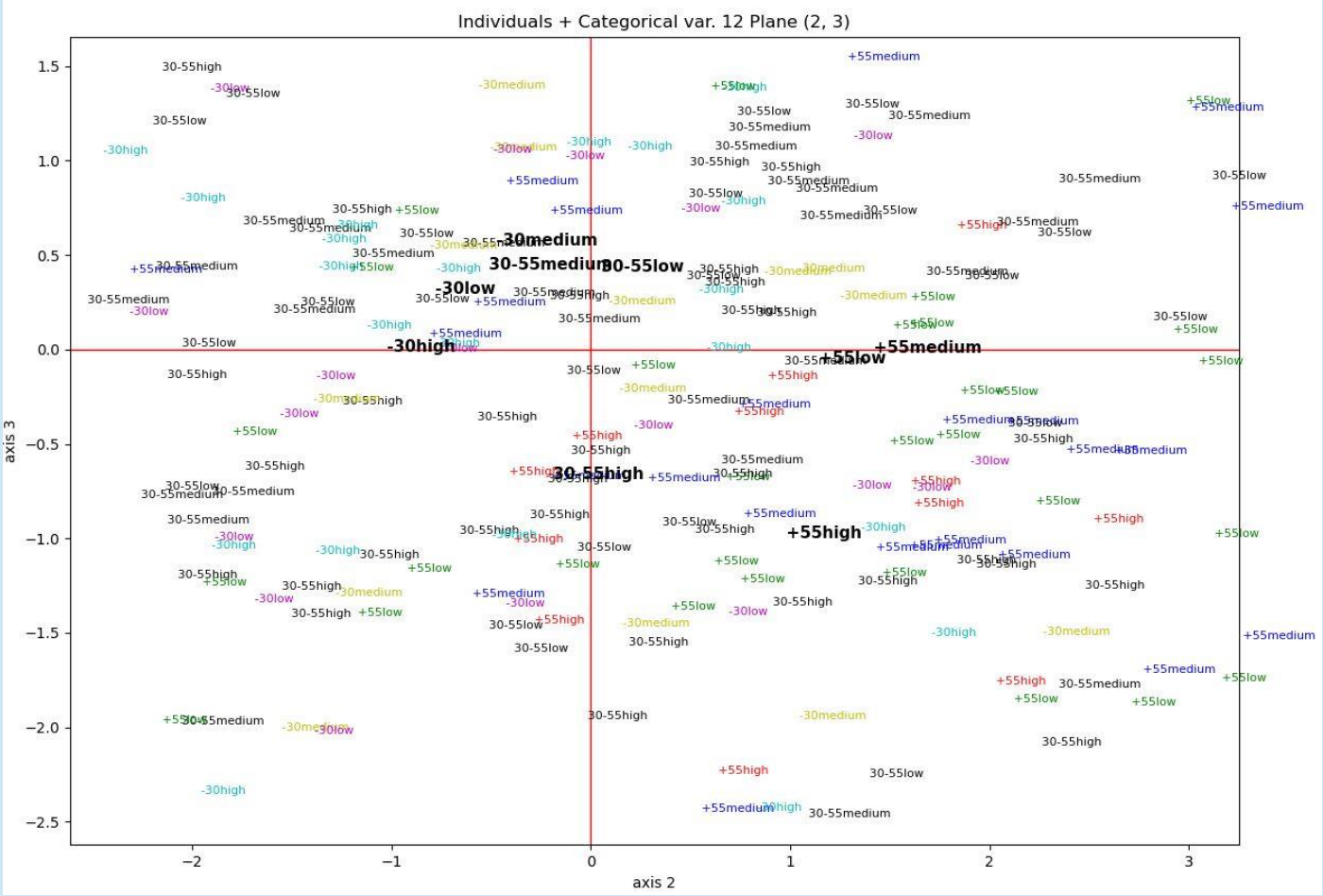

Figure A.6. Position des 9 catégories de la variable nominale supplémentaire 12 (âge, croisé avec le niveau d'éducation) parmi les individus actifs (colorés et identifiés par leur catégorie) dans le plan (2, 3). Les catégories supplémentaire (noir, gras) sont les points moyens des individus concernés.

Remarque (rappel) :

Ces 5 appels de fonctions auraient pu être remplacés par l'unique appel de la fonction synthétique ACP( ) figurant à la fin du code : **`ACP (lane, lane_sup, lane_var_sup, ax_h, ax_v, vsup_lim, vsup_nom).`** Mais on peut vouloir représenter d'autres plans factoriels, d'autres variables nominales supplémentaires, ou sélectionner les variables numériques supplémentaires... l'appel par fonctions séparées est plus souple et donne plus d'information sur le programme.

3. Contenu du fichier source téléchargeable : acomp_dtmF.py

Ce fichier contient sous forme de commentaires le mode d'emploi du programme (instructions à saisir dans l'interface).

```
#*****************
#*** ici le fichier source : "acomp_dtmF.py" et les données : Mini_Sem.txt,
#*** Mini_Sem_vsup, Mini_Sem_isup.txt sont dans un dossier "mydirectory"
#*** directement dans la racine "c:/".
#*** A adapter pour chaque utilisateur, qui remplacera « mydirectory »
#*** par le chemin menant à son dossier de travail.
#-----------------------------------------------------
#*** Dans l'interface (IDLE…),
#*** copier une à une les 4 lignes suivantes (sans les « # »…)
#-----------------------------------------------------
# import os
# os.chdir("c:/mydirectory")
# import acomp_dtmF
# from acomp_dtmF import *
#-----------------------------------------------------#
#*** fonction d'importation: numpy, pandas, matplotlib
#*** Inutile de copier ces trois lignes : Exécution automatique au
#*** chargement/importation du fichier acomp_dtmF.py
#
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#-----------------------------------------------------
#*** Copier successivement les instructions qui suivent le symbole « # ».
#*** Les commentaires (#***  ) n'ont évidemment pas besoin d'être copiés.
#
#***-----------------------------------------------------------
#*** SELECTION DES CHEMINS (adresses des fichiers)
#*** Exemple "Mini_Sem" : 3 chemins pour l'exemple Mini_Sem (mini-sémiométrie)
# lane, lane_sup, lane_var_sup = 'Mini_Sem.txt', 'Mini_Sem_isup.txt',
#   'Mini_Sem_vsup.txt'
#***
#*** Exemple "Sem_300" : 3 chemins pour l'exemple Sem_300
# lane, lane_sup, lane_var_sup  = 'Sem300.txt', 'Sem300_isup.txt',
#  'Sem300_vsup.txt'
#***-----------------------------------------------------------
#*** CHOIX DES PARAMETRES
#*** SELECTION d'une paire d'axes pour les visualisations
#*** pour la sémiométrie, l'axe 1 (facteur de taille) n'est pas pris en compte
# ax_h, ax_v = 2, 3
#***
#*** VARIABLES NUMERIQUES SUPPLEMENTAIRES
# vsup_lim = 3 #(Mini_Sem) ou vsup_lim = 7 (Sem300)
#*** vsup_lim premières variables supplémentaires numériques prises en compte
#***
#*** VARIABLE NOMINALE SUPPLEMENTAIRE
# vsup_nom = 4 #(Mini_Sem) ou vsup_nom = 12 (Sem300)
#*** la var nominale supplémentaire vsup_nom est sélectionnée
#***-----------------------------------------------------------
#*** EXECUTION de l'ACP
#***-----------------------------------------------------------
#*** 1) ACP_base() : lecture données et calculs de base
```

```
# X, c_indiv, c_var, vecp, moy, ecinv = ACP_base(lane)
#
#*** 2) grafact() : plan factoriels éléments actifs (indiv. + variables)
# grafact (X, c_indiv, c_var, ax_h, ax_v )
#
#*** 3) ACP_isup() Individus supplémentaires
# ACP_isup(X, c_indiv, lane_sup,moy, ecinv, vecp, ax_h, ax_v)
#
#*** 4) ACP_vsup() Variables numériques supplémentaires (de 1 -> vsup_lim)
# ACP_vsup(X, c_indiv, c_var, lane_var_sup,ax_h, ax_v, vsup_lim )
#
#*** 5) ACP_nom() Variable nominale supplémentaire (vsup_nom)
# ACP_nom(X, c_indiv, ax_h, ax_v, vsup_nom, lane_var_sup )
#
#***-----------------------------------------------------------
#*** Ces 5 appels peuvent être remplacés par l'unique appel de la fonction
#*** ACP():
#
# ACP (lane, lane_sup, lane_var_sup, ax_h, ax_v, vsup_lim, vsup_nom)
#
#*** Mais on peut vouloir représenter d'autres plans factoriels,
#*** d'autres variables nominales supplémentaires,
#*** ou sélectionner les variables numériques supplémentaires...
#*** l'appel par fonctions séparées est plus souple.
#***-----------------------------------------------------------
#*** fin des lignes à copier dans l'interface
#-------------------------------------------------------------------------
#
#
#-------------------------------------------------------
#--------------- CODES DE TOUTES LES FONCTIONS APPELEES
#-------------------------------------------------------
#*** Rappel: pd, np, plt sont des variables globales pour toutes
#*** les fonctions de acomp_dtmF.py
#-------------------------------------------------------
#
#*** ACP: appel du calcul, éditions
def ACP_base(lane):
#---
     X, c_indiv, c_var, pourcent, vecp, vp, moy, ecinv = ACP_calc(lane)
     n,p = X.shape
     grafacp_vp(pourcent, p)
#---
     ch ="ACP_file_" + lane      # nom du fichier créé
     g = open(ch, "w+")          # ouverture de "ch"
     print ("\n The coordinates are in the created file; " + ch)
     print("\n Eigenvalues\n")
     print(vp)
     print("\n")
#---
     g.write("\n Dataset\n")
     g.write(lane)
     g.write("\n")
     g.write("\n Eigenvalues\n")
     g.write(str(vp))
     g.write("\n\n")
     pd.set_option('display.max_rows', 10)
#---
     ligne = "Coordinates of variables"
```

```
      print(ligne)
      print("\n")
      print(pd.DataFrame({'ident':X.columns, 'c_var_1': c_var[:,0], 'c_var_2':
c_var[:,1],'c_var_3': c_var[:,2],  }))
      g.write(ligne)
      g.write("\n\n")
      info = "\n complete coord. and coord. of indiv. are in the created file:
" + ch + "\n\n"
      print(info)
      pd.set_option('display.max_rows', n)
#---
      a = pd.DataFrame({'ident':X.columns, 'c_var_1': c_var[:,0], 'c_var_2':
c_var[:,1],'c_var_3': c_var[:,2],  })
      sa = str(a)
      g.write(sa)
      g.write("\n\n")
      ligne = "Coordinates of individuals or observations"
      g.write(ligne)
      g.write("\n\n")
#---
      aa = pd.DataFrame({'ident':X.index, 'c_indiv_1': c_indiv[:,0],
'c_indiv_2': c_indiv[:,1],'c_indiv_3': c_indiv[:,2],  })
      saa = str(aa)
      g.write(saa)
      g.close()
      pd.reset_option('display.max_rows')
      return X, c_indiv, c_var, vecp, moy, ecinv
#-----------------------------------------------------
#
#*** ACP : calculs de base
def ACP_calc(lane):
      X = pd.read_csv(lane)
      n,p = X.shape
      moy = np.mean(X)
      ec= np.std(X)
      ecinv = 1./ec
#--- Donnée standardisées StanX
      StanX = np.zeros((n,p))
      for i in range(n):
            for j in range(p):
                  StanX[i,j] = (X.iloc[i,j] -moy[j])*ecinv[j]
      C = np.dot(StanX.T, StanX)/n
      vp1, vecp1 = np.linalg.eigh(C)
      c_var = np.zeros((p,p))
      vecp  = np.zeros((p,p))
      vp    = np.zeros(p)
#---inversion de l'ordre des val et vec propres -----
      for j in range(p):
            jj = p-j-1
            vp[j] = vp1[jj]
            vecp[:,j] = vecp1[:,jj]
            c_var[:,j]= vecp[:,j]*np.sqrt(vp[j])
#---Fin de l'inversion
      c_indiv = np.dot(StanX, vecp)
      trace = np.sum(vp)
      pourcent = vp/trace
      return X, c_indiv, c_var, pourcent, vecp, vp, moy, ecinv
#-----------------------------------------------------
#
```

```
#*** Graphique des valeurs propres
def grafacp_vp(pourcent, p):
     plt.plot(np.arange(1, p+1), pourcent)
     plt.title("Eigenvalues")
     plt.xlabel("Axes")
     plt.ylabel("Eigenvalues (as percentages of trace)")
     plt.show()
#-----------------------------------------------------
#
#*** Graphiques (plan (ax_h, ax_v) des éléments actifs)
def grafact ( X, c_indiv, c_var, ax_h, ax_v ):
     xx = ax_h
     yy = ax_v
     n,p = X.shape
#---
     graf_var (X, c_var, p, xx, yy)
     graf_ind (X, c_indiv,  n, xx, yy)
     return
#-----------------------------------------------------
#
#*** Graphiques (plan (ax_h, ax_v) ) des variables
def graf_var ( X, c_var, p, ax_h, ax_v):
     lax = 8
     fig, axes = plt.subplots(figsize = (lax,lax))
     axes.set_xlim (-1,1)
     axes.set_ylim(-1,1)
     FS = 8    # FS = fontsize
     for j in range(p):
          plt.annotate (X.columns[j],(c_var[j,ax_h - 1], c_var[j,ax_v - 1]),
fontsize = FS)
     plt.plot([-1,1],[0,0], color = 'blue', linestyle = "-", linewidth = 1)
     plt.plot([0,0],[-1,1], color = 'blue', linestyle = "-", linewidth = 1)
     plt.title("Variables(correlations) Plane "+ str(ax_h)+ ", "+ str(ax_v))
     plt.xlabel("axis " + str(ax_h))
     plt.ylabel("axis " + str(ax_v))
     discor = plt.Circle( (0,0),1, color = 'green', fill = False)
     axes.add_artist(discor)
     plt.show()
#-----------------------------------------------------
#
#*** Graphiques (plan (ax_h, ax_v) ) des individus
def graf_ind ( X, c_indiv, n, ax_h, ax_v):
#--- cadrage (mêmes unités)
     xh = c_indiv[:,ax_h - 1]
     xv = c_indiv[:,ax_v - 1]
#--- a :  pour élargir le cadre
     a = 1
     FS = 8   # FS = fontsize
     lmin = min(min(xv), min(xh)) - a
     lmax = max(max(xv), max(xh)) + a
     lax = lmax - lmin
     fig, axes = plt.subplots(figsize = (lax,lax))
     axes.set_xlim (lmin,lmax)
     axes.set_ylim(lmin, lmax)
     for i in range(n):
          plt.annotate (X.index[i],(c_indiv[i,ax_h - 1], c_indiv[i,ax_v - 1]),
fontsize = FS)
     plt.plot([lmin,lmax],[0,0], color = 'red', linestyle = "-",linewidth = 1)
     plt.plot([0,0],[lmin,lmax], color = 'red', linestyle = "-",linewidth = 1)
```

```
     plt.title("Individuals (observations) Plane " + str(ax_h)+ ", "+
str(ax_v))
     plt.xlabel("axis " + str(ax_h))
     plt.ylabel("axis " + str(ax_v))
     plt.show()
#-------------------------------------------------------
#
#*** appels des coordonnées et graphiques des individus supplémentaires
def ACP_isup( X, c_indiv, lane_sup,moy, ecinv, vecp, ax_h, ax_v):
#--- Données individus supplémentaires
     df_isup, df_isup_norm, c_isup = isup(lane_sup, moy, ecinv, vecp)
#--- Graphiques individus supplémentaires
     graf_ind_sup (X, c_indiv, c_isup,  df_isup, ax_h, ax_v)
     return
#-------------------------------------------------------
#
#*** Coordonnées des individus supplémentaires:
def isup(lane_sup, moy, ecinv, vecp):
     df_isup = pd.read_csv(lane_sup)
     q,p = df_isup.shape
     df_isup_norm = np.zeros((q,p))
     for i in range(q):
          for j in range(p):
               df_isup_norm [i,j] = (df_isup.iloc[i,j] -moy[j])*ecinv[j]
     c_isup = np.dot(df_isup_norm, vecp)
     return df_isup, df_isup_norm, c_isup
#-------------------------------------------------------
#
#*** Graphiques (plan (ax_h, ax_v) ) des individus supplémentaires
def graf_ind_sup (X, c_indiv, c_isup, df_isup, ax_h, ax_v):
#--- cadrage (mêmes unités)
     xh = c_indiv[:,ax_h - 1]
     xv = c_indiv[:,ax_v - 1]
     xh_sup = c_isup[:,ax_h - 1]
     xv_sup = c_isup[:,ax_v - 1]
#--- a pour élargir le cadre
     a = 1
     FS = 8  # FS = fontsize
     lmin = min(min(xv), min(xh),min(xv_sup), min(xh_sup)) - a
     lmax = max(max(xv), max(xh), max(xv_sup), max(xh_sup)) + a
     lax = lmax - lmin
#---
     n = X.shape[0]           # nombre d'indiv. actifs
     fig, axes = plt.subplots(figsize = (2*lax,2*lax))
     axes.set_xlim (lmin,lmax)
     axes.set_ylim(lmin,lmax)
     for i in range(n):       # ind actifs
          plt.annotate (X.index[i],(c_indiv[i,ax_h - 1], c_indiv[i,ax_v - 1]),
fontsize = FS)
     nsup = df_isup.shape[0]  # nombre d'indiv. suppl.
     for i in range(nsup):    # ind suppl.
          plt.annotate (df_isup.index[i],(c_isup[i,ax_h - 1], c_isup[i,ax_v -
1]), color = 'b')
#--- tracé des axes
     plt.plot([-lax,lax],[0,0], color = 'red', linestyle = "-",linewidth = 1)
     plt.plot([0,0],[-lax,lax], color = 'red', linestyle = "-",linewidth = 1)
     # titres et étiquettes
     plt.title("Active individuals + suppl. Plane " + str(ax_h)+ ", "+
str(ax_v))
```

```
      plt.xlabel("axis " + str(ax_h))
      plt.ylabel("axis " + str(ax_v))
      plt.show()
#-------------------------------------------------------
#
#*** Variables numériques supplémentaires
def ACP_vsup(X, c_indiv, c_var,  lane_var_sup,ax_h, ax_v, vsup_lim ):
#--- Appel Données variables supplémentaires
      df_vsup, df_vs_num, c_vsup = vsup(X, c_indiv, lane_var_sup, vsup_lim)
      if (vsup == 0):
            return
#--- Appel Graphiques variables supplémentaires
      graf_var_sup (X, c_var, df_vsup, df_vs_num, c_vsup, ax_h, ax_v)
      return df_vsup
#-------------------------------------------------------
#
#*** Données variables supplémentaires:
def vsup(X, c_indiv, lane_var_sup, vsup_lim):
      n, p = X.shape
      df_vsup = pd.read_csv(lane_var_sup)
      qtot = df_vsup.shape[1] #  qtot = nombre total de v. sup
      if (vsup_lim > qtot):
            raise ValueError(" Error, vsup_lim too large !")
#---vsup_lim = les vsup_lim premières vsup (numériques) sont projetées
      df_vs_num = df_vsup.iloc[:,:vsup_lim].values
      q = df_vs_num.shape[1]  # nombre de vsup num
      c_vsup = np.zeros((q,p))
      for j in range(p):
            for m in range(q):
                  c_vsup[m,j] = np.corrcoef(df_vs_num[:,m], c_indiv[:,j])[0,1]
      print(c_vsup)
      return df_vsup, df_vs_num, c_vsup
#-------------------------------------------------------
#
#*** Graphiques (plan (ax_h, ax_v) ) des variables supplémentaires
def graf_var_sup (X, c_var, df_vsup, df_vs_num, c_vsup,  ax_h, ax_v):
      lax = 8
      q = df_vs_num.shape[1]  # nombre de var. sup.
      p = X.shape[1]  # nombre d'axes (donc de var. act.)
      fig, axes = plt.subplots(figsize = (lax,lax))
      axes.set_xlim (-1,1)
      axes.set_ylim(-1,1)
      FS = 8  # FS = fontsize
#--- variables actives
      for j in range(p):
            plt.annotate (X.columns[j],(c_var[j,ax_h - 1], c_var[j,ax_v - 1]),
fontsize = FS)
#--- variables num suppl.
      for j in range(q):
            plt.annotate (df_vsup.columns[j],(c_vsup[j,ax_h - 1], c_vsup[j,ax_v
- 1]), color = 'g')
#---
      plt.plot([-1,1],[0,0], color = 'blue', linestyle = "-", linewidth = 1)
      plt.plot([0,0],[-1,1], color = 'blue', linestyle = "-", linewidth = 1)
      plt.title("Variables(correlations) Plane "+ str(ax_h)+ ", "+ str(ax_v))
      plt.xlabel("axis " + str(ax_h))
      plt.ylabel("axis " + str(ax_v))
      discor = plt.Circle( (0,0),1, color = 'green', fill = False)
      axes.add_artist(discor)
```

```
      plt.show()
#-----------------------------------------------------
#
#*** Données variables nominales supplémentaires
def ACP_nom(X, c_indiv, df_vsup, ax_h, ax_v, vsup_nom ):
#--- choix de la variable nominale supplémentaire
      df_vs_nom = df_vsup.iloc[:,vsup_nom - 1]
#---la variable nominale occupe la colonne vsup_nom de df_vsup.
#---
      col,b, nmod, dicmod = couleur (df_vs_nom)
# Graphiques individus/variables nominales supplémentaires
      graf_ind_mod ( X, c_indiv, df_vs_nom, vsup_nom, ax_h, ax_v, col, b, nmod,
dicmod)
      return
#-----------------------------------------------------
#
#*** Graphiques (plan (ax_h, ax_v) ) d'une variables nominale supplémentaire
def graf_ind_mod (X, c_indiv, df_vs_nom, vsup_nom, ax_h, ax_v, col, b, nmod,
dicmod):
#--- cadrage (mêmes unités)
      h,  v = ax_h - 1,  ax_v - 1
      xh = c_indiv[:,h]
      xv = c_indiv[:,v]
#---  a   pour élargir le cadre
      FS = 8  # FS = fontsize
      a = 1
      lmin = min(min(xv), min(xh)) - a
      lmax = max(max(xv), max(xh)) + a
      lax = lmax - lmin
      fig, axes = plt.subplots(figsize = (lax,lax))
      axes.set_xlim (lmin,lmax)
      axes.set_ylim(lmin, lmax)
      n = X.shape[0]  # nombre d'indiv. actifs
#---
      a = list(df_vs_nom)
      for i in range(n):
            plt.annotate (a[i],(xh[i], xv[i]), color = col[i], fontsize = FS)
#---
      xhnom = coord_nom(xh, b, nmod,n)
      xvnom = coord_nom(xv, b, nmod,n)
      for j in range(nmod):
            plt.annotate (dicmod[j],(xhnom[j], xvnom[j]), color = 'k',
weight='bold', fontsize = FS +3)
#---
      plt.plot([lmin,lmax],[0,0], color = 'red', linestyle = "-",linewidth = 1)
      plt.plot([0,0],[lmin,lmax], color = 'red', linestyle = "-",linewidth = 1)
      plt.title("Individuals + Categorical var. " + str(vsup_nom) + ", Plane ("
+ str(ax_h)+ ", "+  str(ax_v) + ")")
      plt.xlabel("axis " + str(ax_h))
      plt.ylabel("axis " + str(ax_v))
      plt.show()
      return
#-----------------------------------------------------
#
#*** codage numérique (entiers consécutifs) des modalités
def numer (df_vs_nom):
      categ = np.unique(df_vs_nom) # liste des modalités distinctes
      nmod = len(categ)
      a = list(df_vs_nom)
```

```
      dic_mod = {}  # dictionnaire clef = categorie (string), valeur = numéro
      for i in range (nmod):
            dic_mod[categ[i]] = i
      b = []
      for i in range(len(a)):
            b.append(dic_mod[a[i]]) # modalité en codage numérique réduit
      dicmod = {}
      for cle, val in list(dic_mod.items()):
            dicmod[val] = cle
      return b, nmod, dicmod
#-----------------------------------------------------
#
#***  Conversion des modalités en couleurs
def couleur (df_vs_nom):
      b, nmod, dicmod = numer(df_vs_nom)
      color7 = ['r','g','b','c','m','y','k']  #  k = black...
      col = []
      for i in range(len(b)):
            a = b[i]
            if(a > 6):
                  a = 6 # les modalités au delà de 7 sont en noir
            col.append(color7[a])
      return col,b, nmod, dicmod
#-----------------------------------------------------
#
#*** Coord. des modalités d'une nominale supplémentaire
#*** (comme points moyens des individus concernés)
def coord_nom(d, b, nmod,n):
#--- d = numerical vector, length = n (indiv cordinates)
#--- nmod = nombre de modalités
#--- b = codage numérique (1,2,...) de la variable nominale
      moy = np.zeros(nmod) # somme par categorie
      eff = np.zeros(nmod) # effectif par catégorie
      for i in range(n):
            eff[b[i]] = eff[b[i]] + 1
            moy[b[i]] = moy[b[i]] + d[i]
      return moy/eff
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
#*** Fonction ACP() Cette fonction réunit les 5 appels des fonctions
#*** principales : ACP_base, grafact, ACP_isup, ACP_vsup, ACP_nom.
#***
def ACP (lane, lane_sup, lane_var_sup, ax_h, ax_v, vsup_lim, vsup_nom):
      X, c_indiv, c_var, vecp, moy, ecinv = ACP_base(lane)
      grafact (X, c_indiv, c_var, ax_h, ax_v )
      ACP_isup(X, c_indiv, lane_sup,moy, ecinv, vecp, ax_h, ax_v)
      ACP_vsup(X, c_indiv, c_var, lane_var_sup,ax_h, ax_v, vsup_lim )
      ACP_nom(X, c_indiv, ax_h, ax_v, vsup_nom, lane_var_sup )
      return
#-------------------------------------------------------------------------
```

Fin du code python

Le code est écrit de la façon la moins cryptique possible, de façon à permettre des adaptations et des compléments, notamment concernant la sortie des résultats numériques. Les stylistes python pourront aussi rendre le code plus compact et élégant.