# Análisis exploratorio de componentes principales en Python



Los dos archivos de ejemplo son extractos de archivos de encuestas de muestra basados en “cuestionarios semiométricos”. Las variables numéricas consisten en puntuaciones del 1 al 7 otorgadas a una lista de palabras dada por los individuos (encuestados) dependiendo de si estas palabras son desagradables o agradables. En ejemplos de tamaño completo, hay 210 palabras y, a veces, varios miles de personas. Las variables nominales (o variables categóricas) describen las diversas características de los encuestados.

Los trabajos relacionados con la semiometría se pueden descargar gratuitamente en este sitio (www.dtmvic.com).
http://www.dtmvic.com/Semiometrie.html (“La sémiométrie”, Dunod, versión francesa de 2003)
http://www.dtmvic.com/doc/Semio_2014_Cover_354x244.pdf (“The semiometric challenge”, L2C, portada de la versión en inglés de 2014)
http://www.dtmvic.com/doc/Semio_2014_format_169x244.pdf (versión en inglés de 2014)

Los dos archivos de datos de ejemplo (Mini_sem y Sem300) constan de tres partes:

Para el ejemplo Mini_Sem (modelo reducido, para un primer contacto elemental con el programa):

1) Individuos activos y variables (Mini_Sem.txt) (7 palabras notadas por 12 individuos).

2) Variables adicionales (Mini_Sem_vsup.txt) (3 palabras adicionales notadas por las mismas 12 personas, + el género del encuestado, variable nominal)

3) Personas adicionales (Mini_Sem_isup.txt). (4 nuevos individuos habiendo notado las mismas palabras activas).

Es con este ejemplo de "modelo reducido" que debe abordarse el programa Python (cuya lista completa también se da a continuación en la sección 3). El programa (código Python) está en el archivo: acomp_dtm.py.

El ejemplo más realista de Sem300 (en total: 300 personas y 70 palabras para puntuar) se analiza en la sección 2.

1) Individuos activos y variables activas (Sem300.txt) (63 palabras señaladas por 296 individuos).

2) Variables adicionales (Sem300_vsup.txt) (7 palabras adicionales notadas por los mismos 296 individuos (variables numéricas), + 6 variables nominales: grupos de edad, nivel educativo, sexo cruzado con edad, con nivel d 'educativo, edad cruzada con educación y género (sexo) del encuestado)

3) Personas adicionales (Sem300_isup.txt). (4 nuevos individuos que han notado las mismas palabras activas).

# 1. Implementación práctica del programa (ejemplo Mini_Sem)

Debe descargar los tres archivos de datos anteriores, así como el archivo de código acomp_dtm.py en una carpeta de trabajo. El archivo acomp_dtm.py se incluye en el apéndice de esta nota, pero también es un archivo descargable separado como los datos.

**Contenido de los tres archivos de datos descargables (ejemplo reducido de Mini_Sem):**
Activas palabras: *arbre (árbol), cadeau (regalo), danger (peligro), morale (moral), orage (tormenta), politesse (cortesía), sensuel (sensual)*
Variables adicionales :palabras y categoria *: fleur (flor), gateau (torta), risque (riesgo), genre (sexo)*

*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. Instrucciones preliminares

Una vez que el archivo de código acomp_dtm.py se ha copiado en su carpeta de trabajo (llamado aquí "mydirectory") con los tres archivos de datos anteriores, todo lo que tiene que hacer es escribir las 4 instrucciones una por una en la interfaz de Python (como IDLE por ejemplo) para acceder al programa y los datos.

Solo tiene que copiar las siguientes cuatro líneas directamente del archivo acomp_dtm.py que habrá abierto en un editor de texto gratuito (como Notepad ++, que permite ediciones claras en Python).

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

**Todas las instrucciones que siguen están comentadas en el archivo acomp_dtm.py y, por lo tanto, pueden simplemente copiarse de este archivo.**

Aquí, la carpeta "mydirectory" está directamente en la raíz "c: /". (Se adaptará a cada usuario, que sustituirá “mydirectory” por la ruta que lleva a su carpeta de trabajo).

La importación anterior de acomp_dtm.py carga automáticamente las bibliotecas *pandas*, *numpy* y *matplotlib*.

## 2. Cálculos básicos de ACP: función ACP_base ()

Para el ejemplo "Mini_Sem", escribiremos en la interfaz de Python los 3 nombres de acceso en una sola línea:

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

Para leer los datos y realizar cálculos básicos de ACP, escriba:

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

Obtenemos las siguientes impresiones y gráficos (Figura 1) en la interfaz.

```
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
```

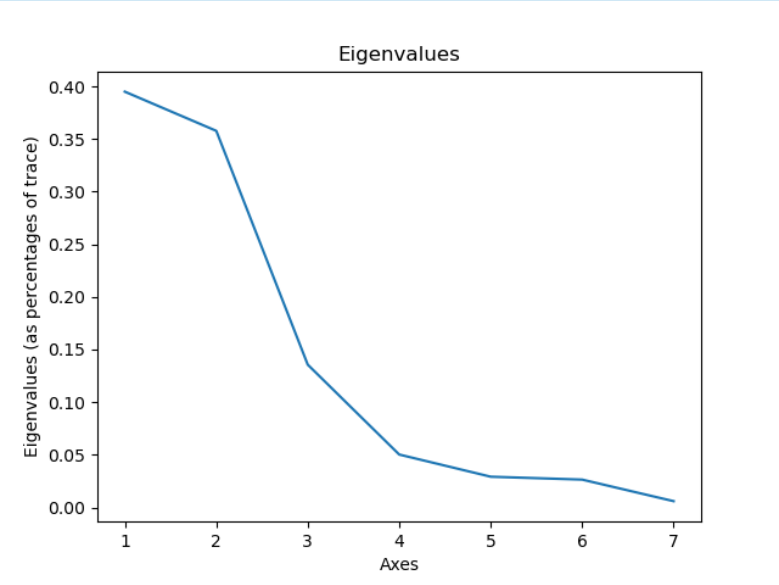


**Figura 1. Serie de valores propios**

## 3) Primeras visualizaciones

Selección de un par de ejes para visualizaciones.
Para la semiometría, no se tiene en cuenta el eje 1 (factor de tamaño, para el que todas las variables son simultáneamente positivas o negativas). Retendremos aquí las visualizaciones en los ejes 2 y 3

**ax_h, ax_v = 2, 3**

Llamar a la función grafact () produce los planos factoriales de elementos activos (variables, individuos)

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

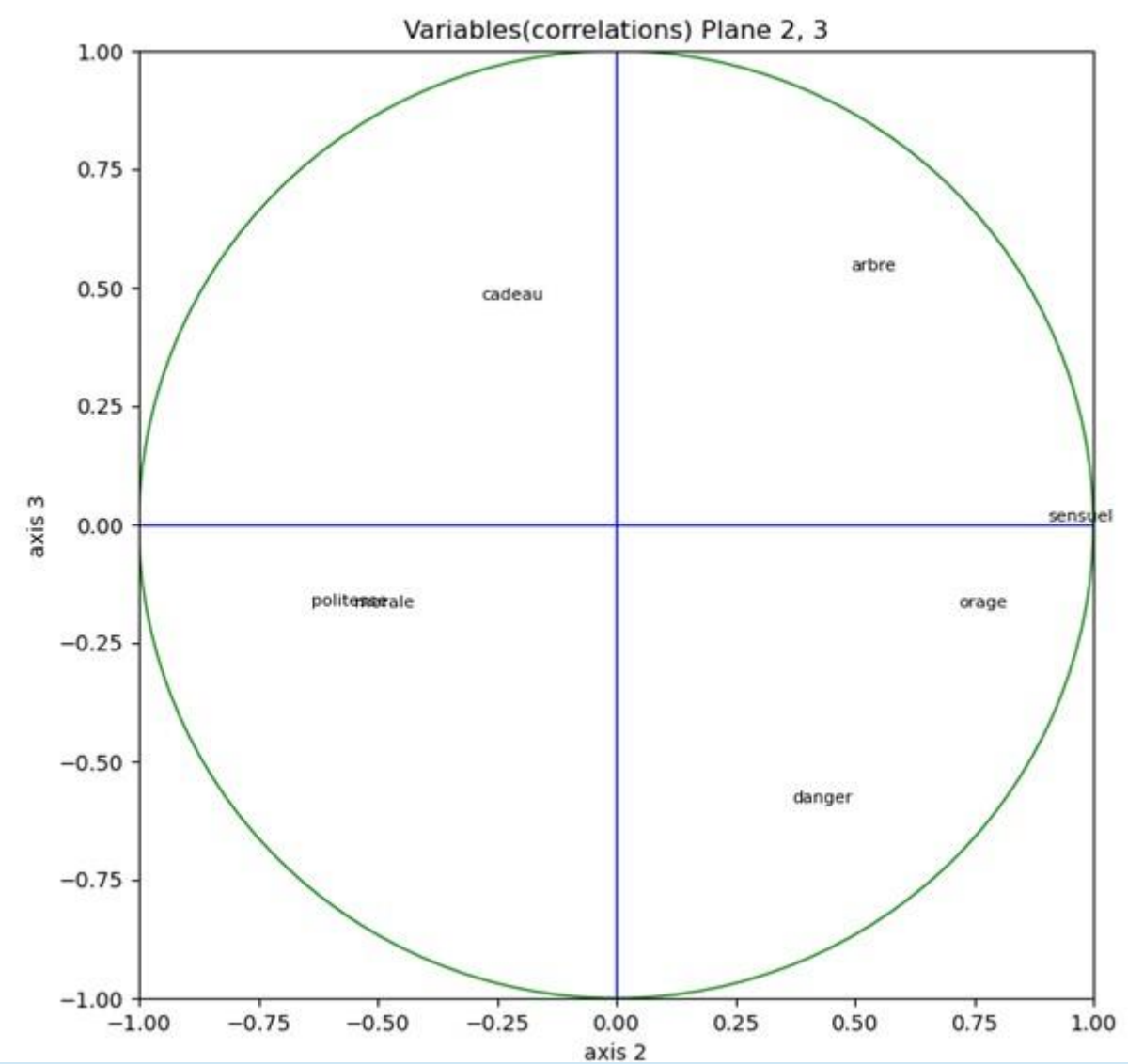


**Figura 2. Posición de las variables en el plano 2, 3**

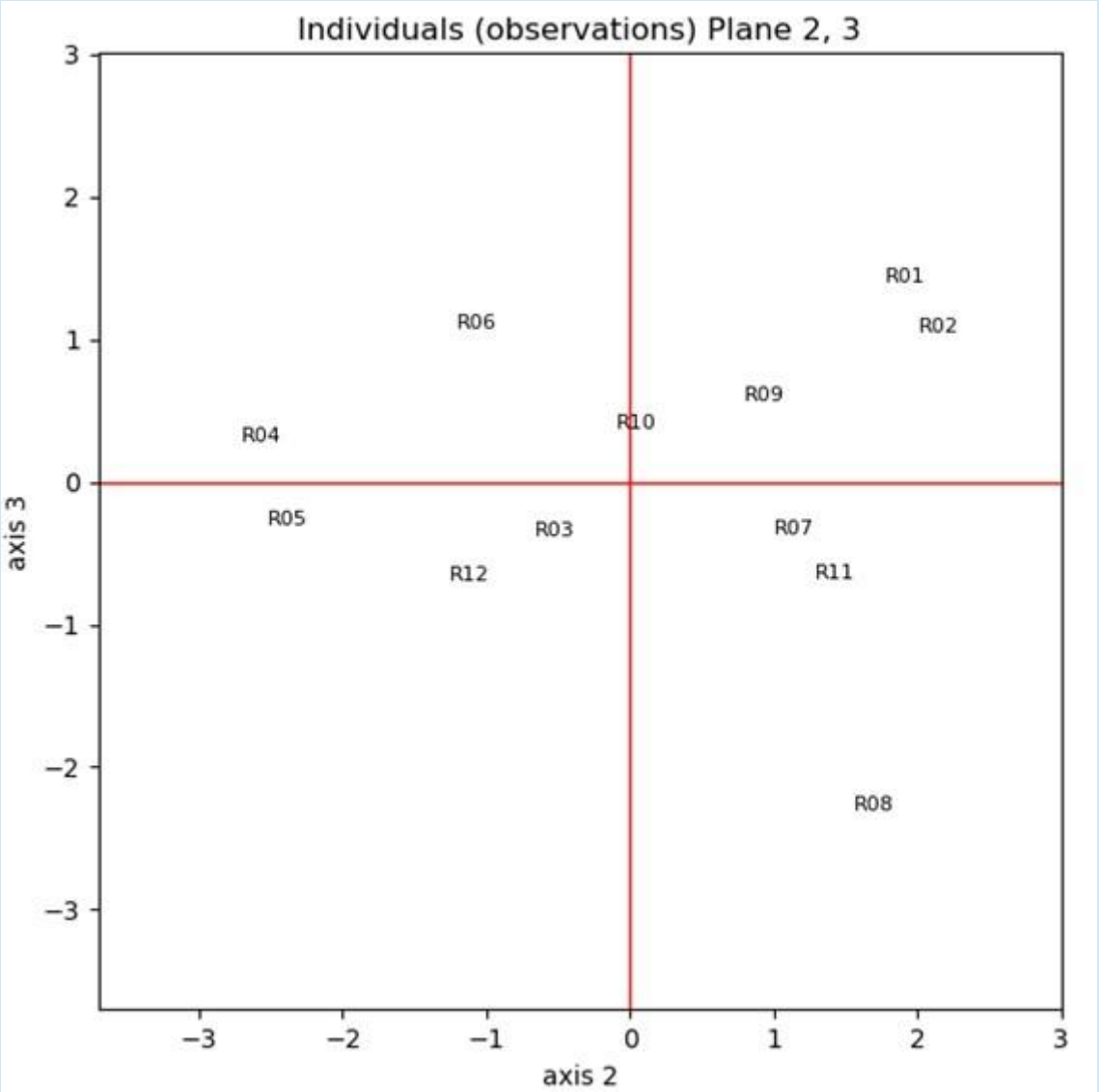


**Figura 3. Posición de los individuos en el plano 2, 3**

## 4) Individuos adicionales (suplementarios, ilustrativos)

Todos los individuos u observaciones (filas de la tabla Mini_Sem_isup) están representados por la llamada a la función ACP_isup ().

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

Se pueden identificar en el plano (2, 3) de la figura 4 por una fuente un poco más grande y un color azul.

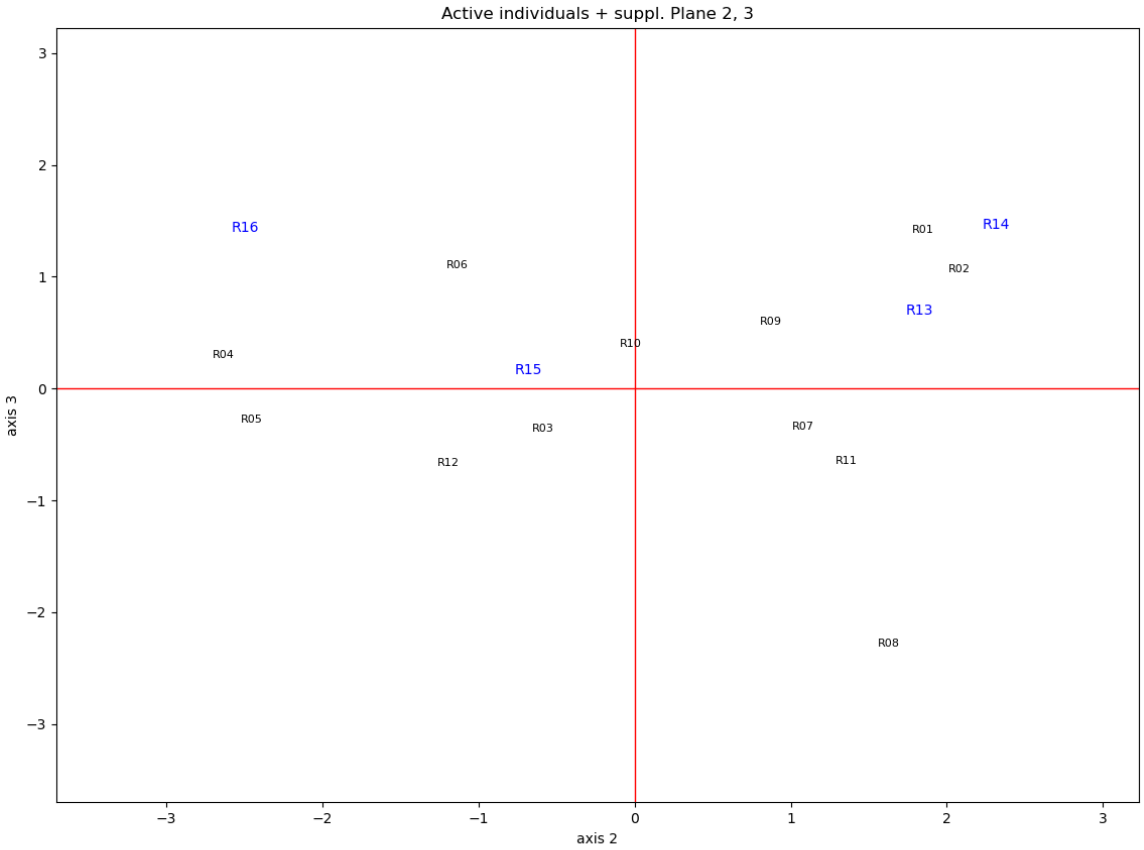


**Figura 4. Posición delos individuos adicionales (R13 a R16) entre los individuos activos en el plano (2, 3)**

## 5) Variables numéricas adicionales

Para las variables, es necesario distinguir entre variables numéricas y variables nominales.
Las 3 variables numéricas están las primeras en el fichero. Escribiremos:

**vsup_lim = 3 #(Mini_Sem)**

Las primeras 3 variables numéricas adicionales son tomadas en cuenta por la función ACP_vsup ().

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

Obtenemos una impresión (resumen) de las coordenadas de las variables adicionales en los ejes.

```
[[-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 Figura 5 muestra las posiciones de las variables adicionales

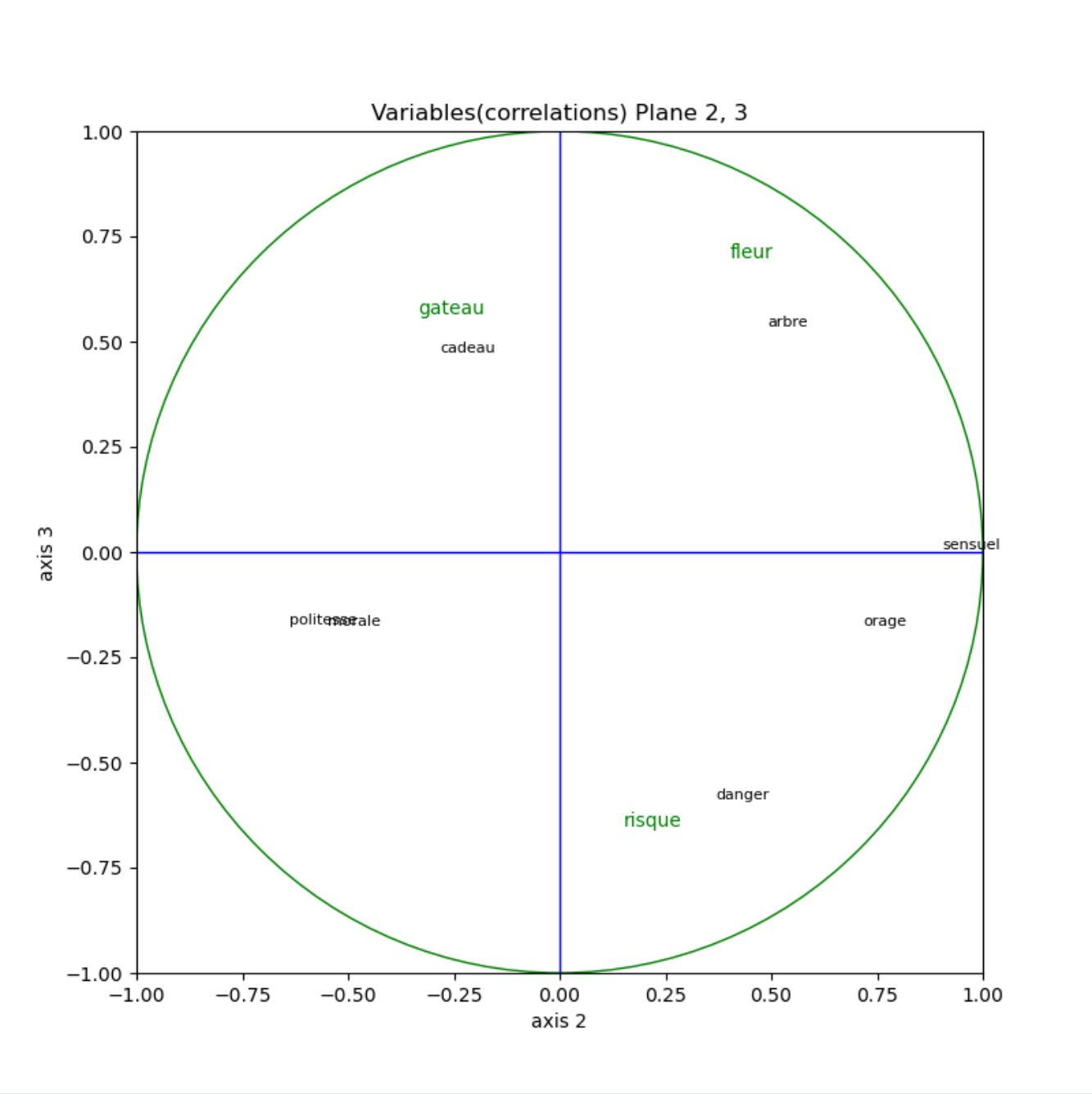


**Figura 5. Posición de las variables adicionales (riesgo, torta, flor) entre las variables activas en el plano (2, 3)**

## 6) Variable nominal adicional (… suplementaria, ilustrativa)

Se eligen uno a uno porque dan lugar a una visualización que incluye a los individuos
Se selecciona la var nominal « vsup_nom ».

.

**vsup_nom = 4**

La representación se realiza mediante la función ACP_nom ().

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

Finalmente se obtiene la figura 6.

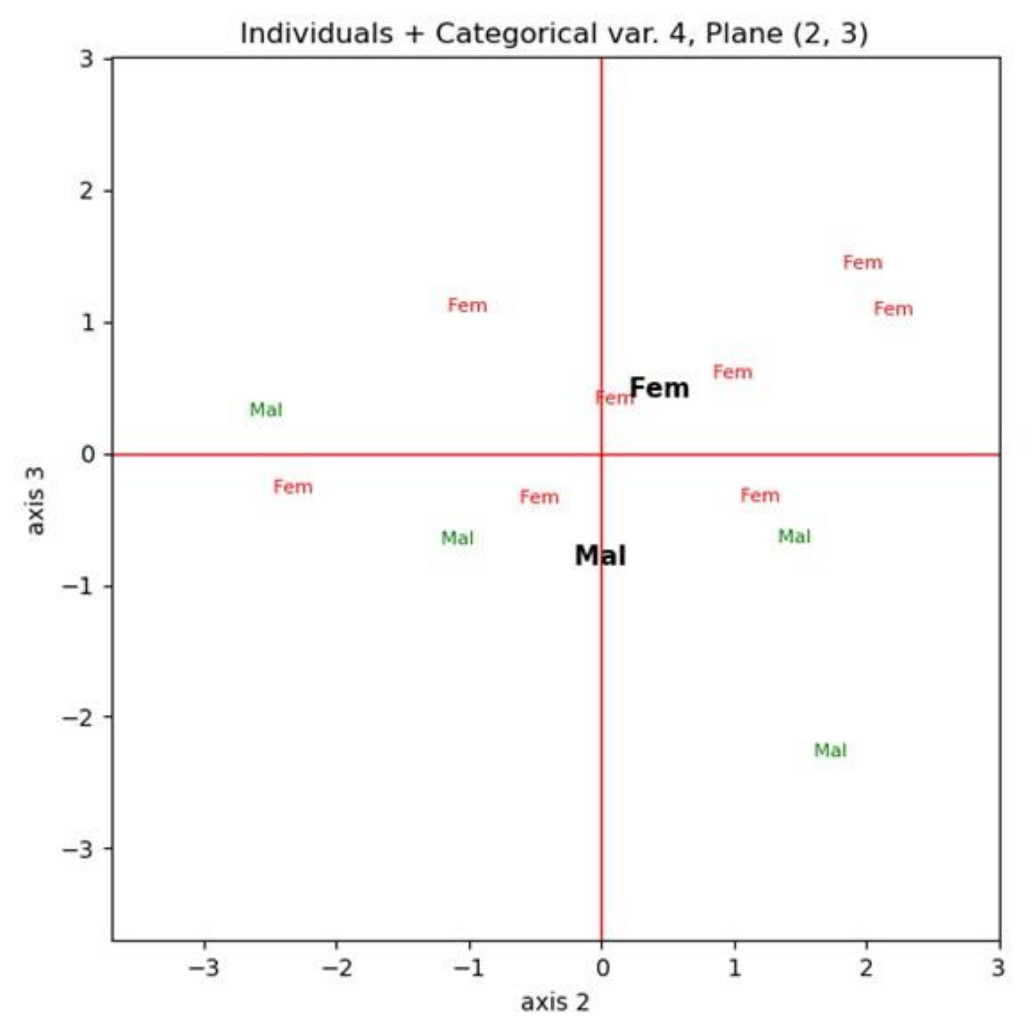


**Figura 6. Posición de las categorías de la variable nominal adicional (Femenino y Masculino) entre los individuos activos (coloreados e identificados por su categoría) en el plano (2, 3).**
**Las categorías adicionales Femenino y Masculino (negras) se encuentran en los puntos promedio de los individuos involucrados.**

Nota:

Estas 5 llamadas a funciones podrían haber sido reemplazadas por la única llamada de la función sintética ACP () que aparece al final del código (después de definir la secuencia de parámetros entre paréntesis de la función):

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

… Pero es posible que desee representar otros planos, otras variables nominales adicionales o seleccionar variables numéricas adicionales ... la llamada por funciones separadas es más flexible y brinda más información sobre el programa.

# 2. Implementación práctica del programa con el ejemplo "Sem300"

El segundo ejemplo, más realista, "Semio_300", se implementa de manera similar.
Las instrucciones para cambiar se encuentran en los comentarios del archivo *acomp_dtm.py*.
Deben descargarse los tres archivos de datos *Sem300.txt*, *Sem300_isup.txt* y *Sem300_vsup*, así como el archivo de código *acomp_dtm.py* en una carpeta de trabajo. El archivo *acomp_dtm.py* se incluye en la sección 3 de esta nota, pero también es un archivo descargable separado como los datos.

## 1. Instrucciones preliminares

Una vez que el archivo de código acomp_dtm.py se ha copiado en su carpeta de trabajo (aquí llamado "mydirectory2") con los tres archivos de datos, todo lo que tiene que hacer es escribir uno por uno en la interfaz de Python (como IDLE por ejemplo) las siguientes 4 instrucciones para proporcionar acceso al programa y a los datos:

**Todas las instrucciones que siguen están comentadas en el archivo *acomp_dtm.py* y, por lo tanto, pueden simplemente copiarse de este archivo.**

**Las instrucciones en verde son idénticas a las utilizadas para el ejemplo educativo "Mini_Sem" presentado anteriormente. Solo las 3 instrucciones en azul son diferentes.**

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

Solo tiene que copiar estas líneas directamente del archivo *acomp_dtm.py* que habrá abierto en un editor de texto gratuito (como Notepad ++, que permite ediciones claras en Python).

Aquí, la carpeta "mydirectory2" está directamente en la raíz "c: /". (Se adaptará para cada usuario, que sustituirá “mydirectory2” por la ruta que lleva a su carpeta de trabajo).
La importación anterior de acomp_dtm.py carga automáticamente las bibliotecas *pandas*, *numpy* y *matplotlib*.

## 2. Cálculos básicos de ACP: función ACP_base ()

Para el ejemplo de "Semio300", escribiremos en la interfaz de Python los 3 nuevos nombres de ficheros en una sola línea:

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

Para la lectura de datos y los cálculos básicos del PCA escribiremos también:

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

Las siguientes impresiones (abreviadas) y gráficos (figura A.1) se obtienen en la interfaz.

```
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
```

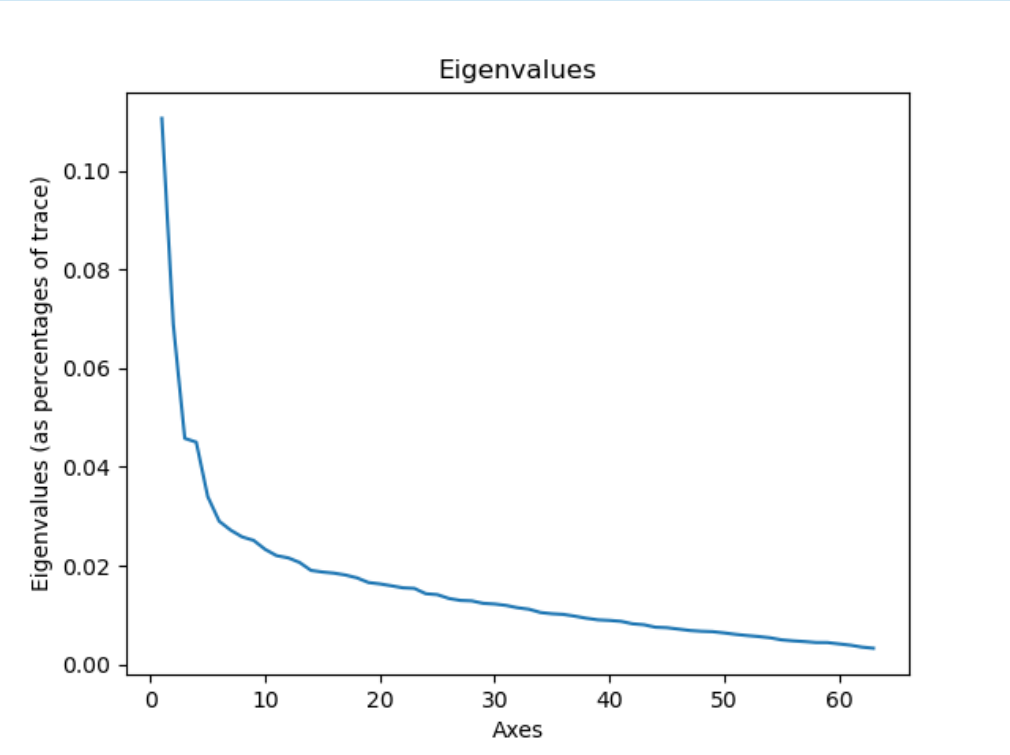


**Figura A.1. Serie de 63 valores propios**

## 3) Primeras visualizaciones

Para la semiometría, no se tiene en cuenta el eje 1 (factor de tamaño).
Conservaremos los ejes 2 y 3.

### ax_h, ax_v = 2, 3

Llamar a la función grafact () produce el plan factorial de elementos activos (individuales + variables)

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

La interfaz IDLE le permite acercarse desde un rectángulo central dentro del círculo de correlación para mayor legibilidad.

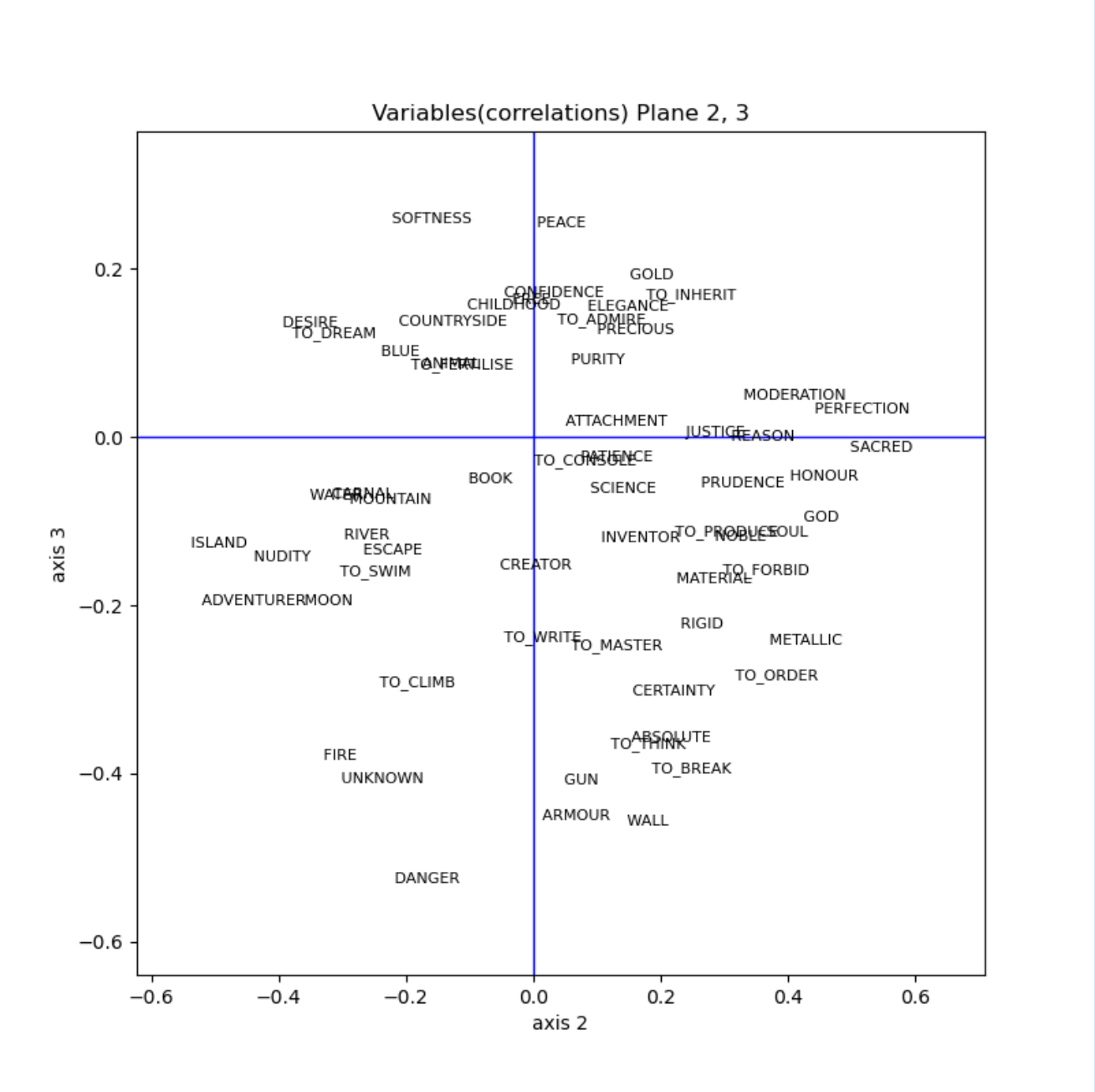


**Figura A.2. Posición de las 63 variables activas en el plano 2, 3 (zoom)**

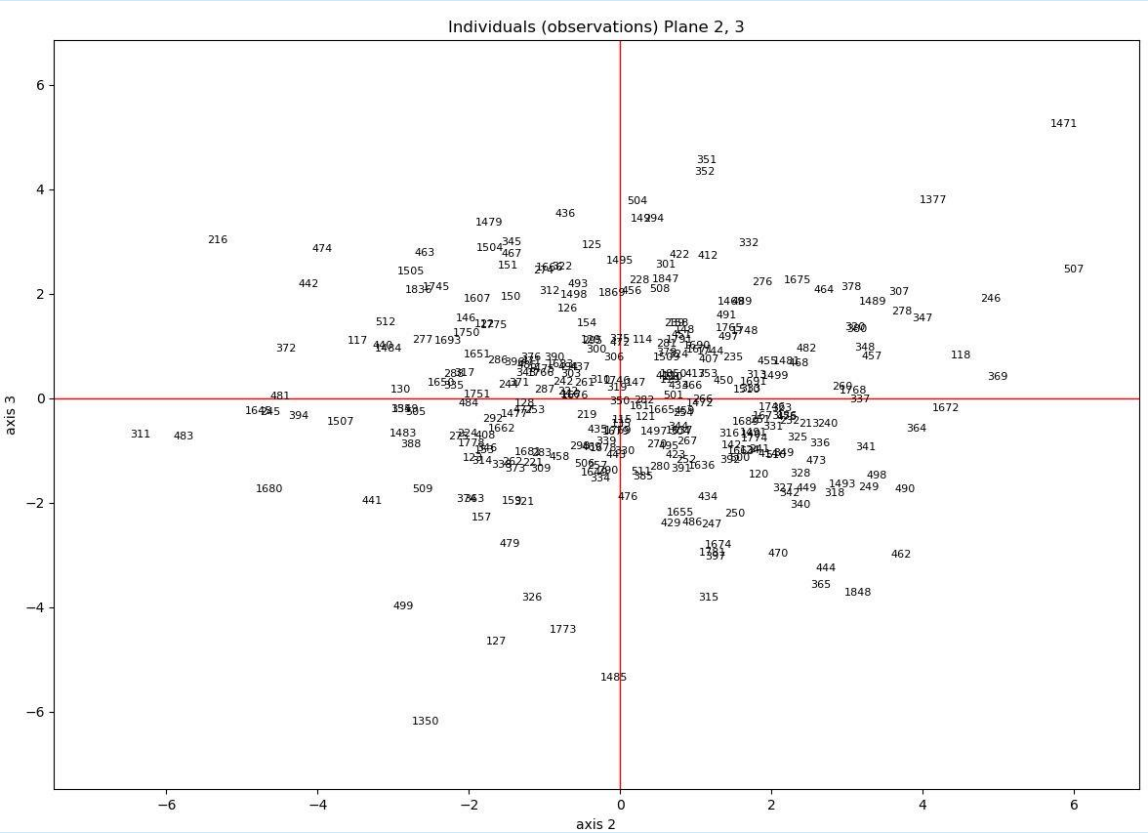


**Figura A.3. Posición de los 296 individuos en el plano (2, 3)**

## 4) Individuos adicionales

Todos los individuos (filas) están representados por la función ACP_isup ().

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

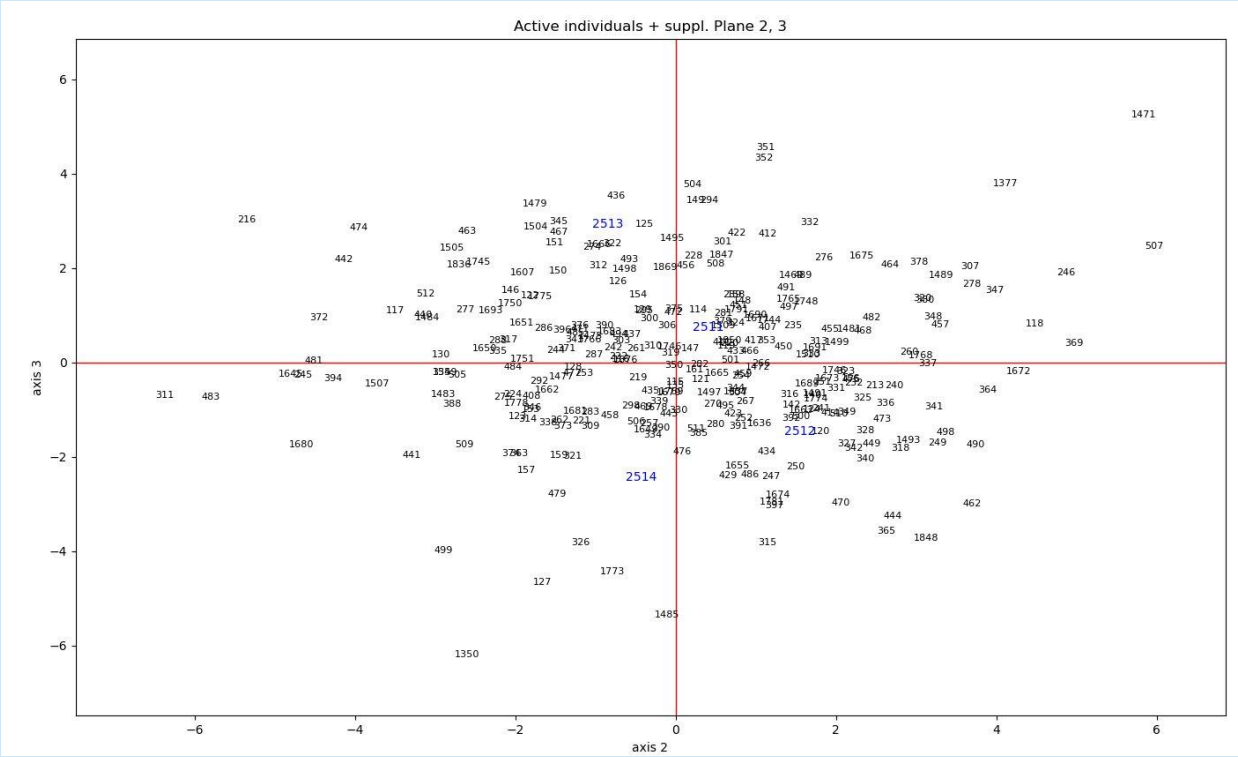


**Figura A.4. Posición de los 4 individuos adicionales (identificadores 2511 a 2514 en azul) entre los individuos activos en el mismo plano 2, 3. Podríamos acercarnos usando la lupa de la interfaz IDLE.**

## 5) Variables numéricas adicionales

Para las variables, es necesario distinguir entre variables numéricas y variables nominales.
Las 7 variables numéricas están las primeras en el fichero. Escribiremos: vsup_lim = 7
(Recuerde que para estos ejemplos, las variables numéricas son notas atribuidas a palabras, mientras que las categorías adicionales son categorías de encuestados en la misma encuesta semiométrica).

**vsup_lim = 7 # (Sem300)**

Las primeras 7 variables numéricas adicionales son tomadas en cuenta por la función ACP_vsup ().

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

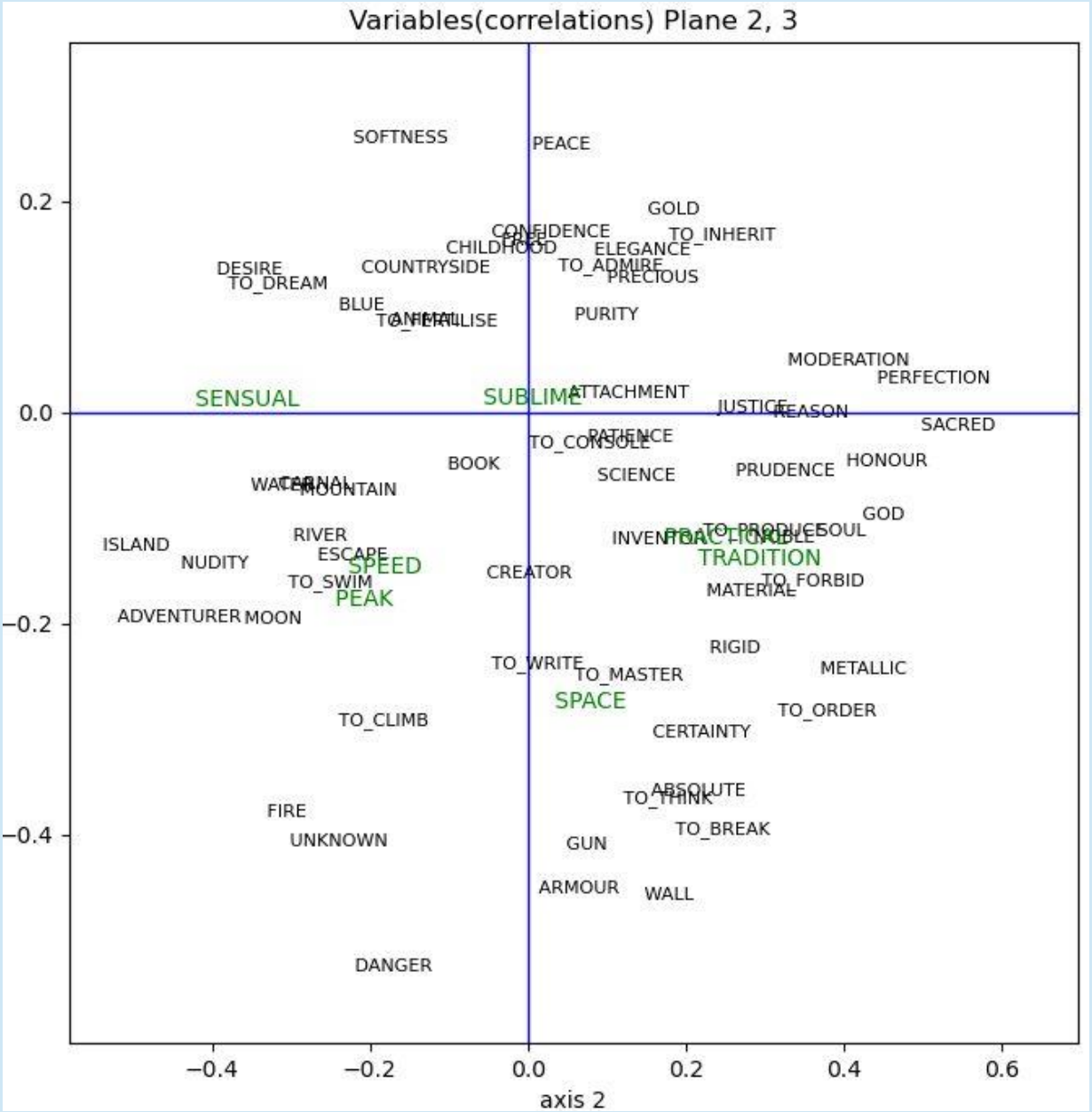


**Figura A.5. Posición de las 7 variables numéricas adicionales (`PEAK, PRACTICAL, SENSUAL, SPACE, SPEED, SUBLIME, TRADITION`) entre las variables activas del plan (2, 3) (*zoom*).**

## 6) Variable nominal adicional (suplementaria)

Se eligen uno a uno porque dan lugar a una visualización de los individuos (identificados por su categoría).

**vsup_nom = 12 # ( Sem300)**

Se selecciona la var nominal vsup_nom = 12. La variable nominal 12 incluye 9 categorías (cruce de 3 grupos de edad [–30, 30-55, +55] con 3 niveles de educación [bajo, medio, alto]. La representación siempre la realiza la función ACP_nom ().

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

Finalmente obtenemos la figura A.6 (también obtenida con el zoom de IDLE)

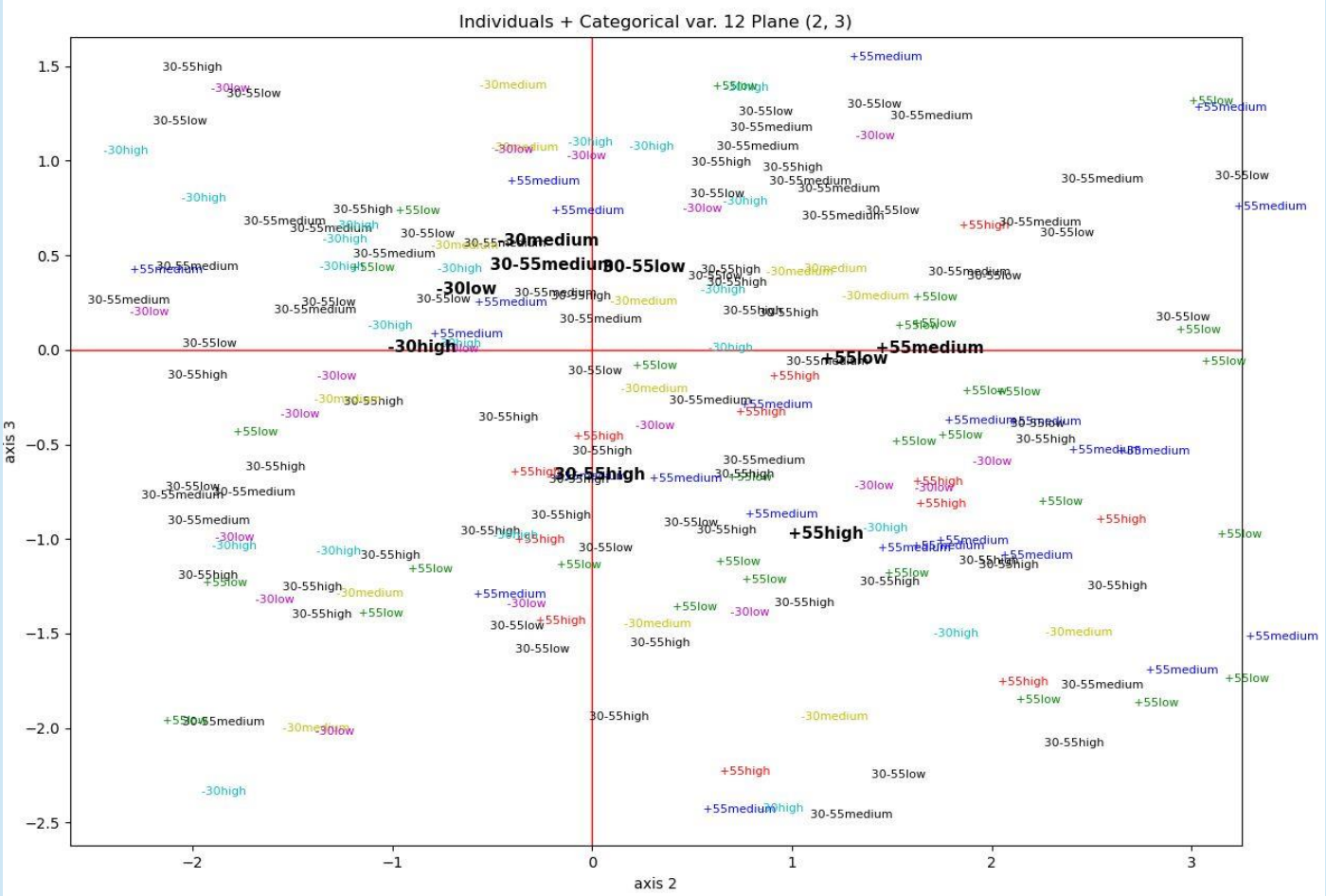


**Figura A.6. Posición de las 9 categorías de la variable nominal adicional 12 (edad, cruzada con el nivel de educación) entre los individuos activos (coloreados e identificados por su categoría) en el plano (2, 3). Las categorías adicionales (negro) son los puntos promedio de las personas involucradas.**

NOTA (*recordatorio*):

Estas 5 llamadas a funciones podrían haber sido reemplazadas por la llamada única de la función sintética ACP () que aparece al final del código:

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

Pero la llamada por funciones separadas es más flexible y brinda más información sobre el programa.

# 3. Contenido del archivo python descargable: `acomp_dtm.py`

*Este archivo contiene las instrucciones de uso del programa en forma de comentarios (instrucciones que se deben ingresar en la interfaz). Los comentarios para el usuario están en inglés.*

```
#***-----------------------------------------------------
#***  Here the source file: "acomp_dtm.py" and the data: Mini_Sem.txt,
#***  Mini_Sem_vsup, Mini_Sem_isup.txt are in a "mydirectory" folder
#***  directly in the root "c: /".
#***  To be adapted for each user, which will replace "mydirectory"
#***  by the path to his working folder.

#***-----------------------------------------------------
#***  In the interface (IDLE ...),
#***  copy the following 4 lines one by one (without the "#")
#***-----------------------------------------------------
# import os
# os.chdir("c:/mydirectory")
# import acomp_dtm
# from acomp_dtm import *
#***-----------------------------------------------------
#*** Import statements for: numpy, pandas, matplotlib
#*** There is no need to copy these three lines: Automatic execution
#*** through the previous importation of: acomp_dtm.py file
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#***-------------------------------------------------------------
#***  Successively copy the instructions following the "# " symbol.
#***  Comments (# ***) obviously do not need to be copied.
#***-------------------------------------------------------------
#***  SELECTION OF PATHS (for the 3 data files)
#***  "Mini_Sem" example: 3 paths for the Mini_Sem example (mini-semiometry)
# lane, lane_sup, lane_var_sup = 'Mini_Sem.txt', 'Mini_Sem_isup.txt',
'Mini_Sem_vsup.txt'
#***
#***  Example "Sem_300": 3 paths for the example Sem_300
# lane, lane_sup, lane_var_sup  = 'Sem300.txt', 'Sem300_isup.txt',
'Sem300_vsup.txt'
#***-------------------------------------------------------------
#*** CHOICE OF PARAMETERS
#***-------------------------------------------------------------
#*** SELECTION of a pair of axes for visualizations
#*** (for semiometry, axis 1 (size factor) is not taken into account)
# ax_h, ax_v = 2, 3
#***
#*** Supplementary NUMERICAL VARIABLES
# vsup_lim = 3 #(Mini_Sem) ou vsup_lim = 7 (Sem300)
#*** vsup_lim first numerical Supplementary variables taken into account
#***
#*** Supplementary CATEGORICAL VARIABLE
# vsup_nom = 4 #(Mini_Sem) or vsup_nom = 12 (Sem300)
#*** the Supplementary nominal var vsup_nom is selected
#***-------------------------------------------------------------
#*** PCA EXECUTION
#***-------------------------------------------------------------
```

```
#*** 1) ACP_base (): read basic data and computations
# X, c_indiv, c_var, vecp, moy, ecinv = ACP_base(lane)
#
#*** 2) grafact() : factorial plane of active elements (indiv. + Variables)
# grafact (X, c_indiv, c_var, ax_h, ax_v )
#
#*** 3) ACP_isup() Supplementary individuals
# ACP_isup(X, c_indiv, lane_sup,moy, ecinv, vecp, ax_h, ax_v)
#
#*** 4) ACP_vsup() Supplementary numerical variables (from 1 -> vsup_lim)
# ACP_vsup(X, c_indiv, c_var, lane_var_sup,ax_h, ax_v, vsup_lim )
#
#*** 5) ACP_nom() Supplementary nominal variable (vsup_nom)
# ACP_nom(X, c_indiv, ax_h, ax_v, vsup_nom, lane_var_sup )
#
#*** End of the lines to copy in the interface
#
#***------------------------------------------------------------
#*** These 5 calls can be replaced by the single call of the function
#***  ACP ():
# ACP (lane, lane_sup, lane_var_sup, ax_h, ax_v, vsup_lim, vsup_nom)
#
#***  However we may want to represent other visualization planes,
#***  use other supplementary nominal variables,
#***  or select supplementary numerical variables ...
#***  the call by separate functions is more flexible.
#***------------------------------------------------------------
#***  End of lines to copy in the interface
#***------------------------------------------------------------
#***--------------- CODES FOR ALL CALLED FUNCTIONS
#***------------------------------------------------------------
#*** Reminder: pd, np, plt are global variables for all
#***  the functions of acomp_dtm.py
#***------------------------------------------------------------
#*** Terminology: nominal <=> categorical
#***  supplementary <=> additional <=> illustrative
#***------------------------------------------------------------
#*** Function ACP_base(): computations, eigenvalues, eigenvectors...
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    # name of created file
     g = open(ch, "w+")
     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
#-----------------------------------------------------
#
#*** Function ACP_calc() : basic computations
def ACP_calc(lane):
      X = pd.read_csv(lane)
      n,p = X.shape
      moy = np.mean(X)
      ec= np.std(X)
      ecinv = 1./ec
#--- StanX : standardized data
      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 of the order of eigen-elements -----
      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])
#---end of the 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
#-----------------------------------------------------
```

```
#
#*** Function grafacp_vp(): Eigenvalue graph
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()
#-----------------------------------------------------
#
#*** Function gracfact(): Graphics (plane (ax_h, ax_v) of active elements)
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
#-----------------------------------------------------
#
#*** Function graf_var(): Plots (plane (ax_h, ax_v)) of 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()
#-----------------------------------------------------
#
#*** Function  graf_ind() : Plots (plane (ax_h, ax_v)) of individuals
def graf_ind ( X, c_indiv, n, ax_h, ax_v):
#--- Frame (same units)
     xh = c_indiv[:,ax_h - 1]
     xv = c_indiv[:,ax_v - 1]
#--- a:  to widen the frame
     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()
#-------------------------------------------------------
#
#*** Function ACP_isup() : graphics of supplementary individuals
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
#-------------------------------------------------------
#
#*** Function isup(): Coordinates of supplementary individuals
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
#-------------------------------------------------------
#
#*** Function graf_ind_sup() : Graphical display (plane (ax_h, ax_v) )
#*** of suppl. indiv.
def graf_ind_sup (X, c_indiv, c_isup, df_isup, ax_h, ax_v):
#--- Frame (same units)
      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: to widen the frame
      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]            # number of active individuals
      fig, axes = plt.subplots(figsize = (2*lax,2*lax))
      axes.set_xlim (lmin,lmax)
      axes.set_ylim(lmin,lmax)
      for i in range(n):        # active individuals
            plt.annotate (X.index[i],(c_indiv[i,ax_h - 1], c_indiv[i,ax_v - 1]),
fontsize = FS)
      nsup = df_isup.shape[0]  # number of suppl. individuals
      for i in range(nsup):    # suppl. individuals
            plt.annotate (df_isup.index[i],(c_isup[i,ax_h - 1], c_isup[i,ax_v -
1]), color = 'b')
#--- drawing the axes
      plt.plot([-lax,lax],[0,0], color = 'red', linestyle = "-",linewidth = 1)
      plt.plot([0,0],[-lax,lax], color = 'red', linestyle = "-",linewidth = 1)
      # titles and tags
```

```
      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()
#-----------------------------------------------------
#
#*** Function ACP_vsup(): Supplementary numerical variable
def ACP_vsup(X, c_indiv, c_var,  lane_var_sup,ax_h, ax_v, vsup_lim ):
#--- Call Supplementary variables data
      df_vsup, df_vs_num, c_vsup = vsup(X, c_indiv, lane_var_sup, vsup_lim)
      if (vsup == 0):
            return
#--- Call Supplementary variables graphics
      graf_var_sup (X, c_var, df_vsup, df_vs_num, c_vsup, ax_h, ax_v)
      return df_vsup
#-----------------------------------------------------
#
#*** Function vsup() : Supplementary variables data:
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 = total number of suppl. variables
      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]  # number of numer. suppl. variables
      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
#-----------------------------------------------------
#
#*** Function graf_var_sup() : Graphics (plane (ax_h, ax_v)) of
#***  supplementary variables
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]  # number of suppl. variables
      p = X.shape[1]  # number of axes (= number of active variables)
      fig, axes = plt.subplots(figsize = (lax,lax))
      axes.set_xlim (-1,1)
      axes.set_ylim(-1,1)
      FS = 8  # FS = fontsize
#--- active variables
      for j in range(p):
            plt.annotate (X.columns[j],(c_var[j,ax_h - 1], c_var[j,ax_v - 1]),
fontsize = FS)
#--- supplémentary numerical variables
      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()
#-------------------------------------------------
#
#*** Function ACP_nom() : Supplementary nominal variable data
def ACP_nom(X, c_indiv, df_vsup, ax_h, ax_v, vsup_nom ):
#--- choice of supplementary nominal variable
      df_vs_nom = df_vsup.iloc[:,vsup_nom - 1]
#--- the nominal variable is the "vsup_nom" column of df_vsup
#--- Conversion of categories into colors
      col,b, nmod, dicmod = couleur (df_vs_nom)
# Individuals / Supplementary nominal variables display
      graf_ind_mod ( X, c_indiv, df_vs_nom, vsup_nom, ax_h, ax_v, col, b, nmod,
dicmod)
      return
#-------------------------------------------------
#
#*** Function graf_ind_mod() : Graphics (plane (ax_h, ax_v)) of a
#***  supplementary nominal variable
def graf_ind_mod (X, c_indiv, df_vs_nom, vsup_nom, ax_h, ax_v, col, b, nmod,
dicmod):
#--- Frame (same units)
      h,  v = ax_h - 1,  ax_v - 1
      xh = c_indiv[:,h]
      xv = c_indiv[:,v]
#---  a : to widen the frame
      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]  # number of active individuals
#---
      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
#-------------------------------------------------
#*** Function numer(): numerical coding (consecutive integers) of categories
#*** (purpose: this digitization is intended to color the individuals
```

```
#*** on the graphs according to their categories)
def numer (df_vs_nom):
      categ = np.unique(df_vs_nom) # list of distinct categories
      nmod = len(categ)
      a = list(df_vs_nom)
      dic_mod = {}  # dictionary key = category (string), value = number
      for i in range (nmod):
            dic_mod[categ[i]] = i
      b = []
      for i in range(len(a)):
            b.append(dic_mod[a[i]]) # numer. coding of categories
      dicmod = {}
      for cle, val in list(dic_mod.items()):
            dicmod[val] = cle
      return b, nmod, dicmod
#-----------------------------------------------------
#
#***  Function couleur(): Conversion of 6 first categories into colors
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 # categories beyond 7 are black
            col.append(color7[a])
      return col,b, nmod, dicmod
#-----------------------------------------------------
#
#*** Function coord_nom() : Coordinates of Suppl. categories
#*** (as average points of the individuals concerned)
def coord_nom(d, b, nmod,n):
#--- d = numerical vector, length = n (indiv cordinates)
#--- nmod = number of categories
#--- b = numerical coding (1,2,...) of the categ. Variable
      moy = np.zeros(nmod) # sum per category
      eff = np.zeros(nmod) frequency of category
      for i in range(n):
            eff[b[i]] = eff[b[i]] + 1
            moy[b[i]] = moy[b[i]] + d[i]
      return moy/eff
#-------------------------------------------------------------------------
#*** ACP () function. This function combines the 5 main function calls:
#*** 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 del codigo de Python**

El código está escrito de la manera menos críptica posible, para permitir adaptaciones y adiciones, en particular en lo que respecta a la salida de los resultados numéricos.
Los estilistas de Python también pueden hacer que el código sea más compacto y elegante