Exploratory Principal Components Analysis in Python.

1. Practical implementation of the program with the small example “Mini_sem”.
2. Practical implementation of the program with the example "Sem300"
3. Python code: Content of the source file: **`acomp_dtm.py`**

The two example files are excerpt from sample surveys files based on “semiometric questionnaires”. The numerical variables consist of scores from 1 to 7 given to a list of words given by individuals (respondents) depending on whether these words are unpleasant or pleasant. In full-size examples, there are 210 words and sometimes several thousand individuals. The nominal variables (or categorical variables) describe the various characteristics of the respondents.

The works relating to semiometry are free to download on this site (www.dtmvic.com).
http://www.dtmvic.com/Semiometrie.html (“La sémiométrie”, Dunod, French version of 2003)
http://www.dtmvic.com/doc/Semio_2014_Cover_354x244.pdf (“The semiometric challenge”, L2C, cover of the English version of 2014)
http://www.dtmvic.com/doc/Semio_2014_format_169x244.pdf (2014 English version)

The two example data files (Mini_sem and Sem300) consist of three parts:

For the **Mini_Sem** example (reduced model, for a first elementary contact with the program):

1) Active individuals and variables (Mini_Sem.txt) (7 words noted by 12 individuals).

2) Additional variables (Mini_Sem_vsup.txt) (3 additional words noted by the same 12 individuals, + the gender of the respondent, nominal variable)

3) Additional individuals (Mini_Sem_isup.txt). (4 new individuals having noted the same active words).

It is with this "Reduced model" example that the python program should be approached (the complete listing of which is also given below in section 3). The program (python code) is in the file: acomp_dtm.py.

The more realistic Sem300 example (in all: 300 individuals and 70 words to score) is discussed in section 2.

1) Active individuals and active variables (Sem300.txt) (63 words noted by 296 individuals).

2) Additional variables (Sem300_vsup.txt) (7 additional words noted by the same 296 individuals (numerical variables), + 6 nominal variables: age groups, education level, sex crossed with age, with level d 'education, age crossed with education, and gender (sex) of the respondent)

3) Additional individuals (Sem300_isup.txt). (4 new individuals having scored the same active words).

1. Practical implementation of the program (Mini_Sem example) .

You should download the three previous data files as well as acomp_dtm.py code file in a working folder. The acomp_dtm.py file is included in section 3 of this note but is also a separate downloadable file like the data.

Content of the three downloadable data files (Mini_Sem "mini example"):

List of active words: *arbre (tree), cadeau (gift), danger (hazard), morale (moral), orage (storm), politesse (politeness), sensuel (sensual)*

List of supplementary (or: illustrative) words and categories*: fleur (flower), gateau (cake), risque (risk), genre (gender)*

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. Preliminary instructions

Once the acomp_dtm.py code file has been copied into your working folder (called here "mydirectory") with the three previous data files, all you have to do is type the 4 instructions one by one in the python interface (like IDLE for example) to access the program and data:

You just have to copy the following four lines directly from the acomp_dtm.py file that you will have opened in a free text editor (such as Notepad ++, which allows clear edits in python).

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

All the instructions that follow are copied as *comment lines* in the acomp_dtm.py file and can therefore simply be copied from this file.

Here, the "mydirectory" folder is directly in the "c: /" root. (To be adapted for each user, who will replace "mydirectory" by the path leading to his working folder).

The above import of acomp_dtm.py automatically loads the *pandas*, *numpy* and *matplotlib* libraries.

2. Basic PCA calculations: ACP_base () function

For the "Mini_Sem" example, we will write in the python interface the 3 access names on a single line:

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

To read the data and perform basic PCA calculations, type:

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

We obtain the following impressions and graphics (Figure 1) on the 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
```

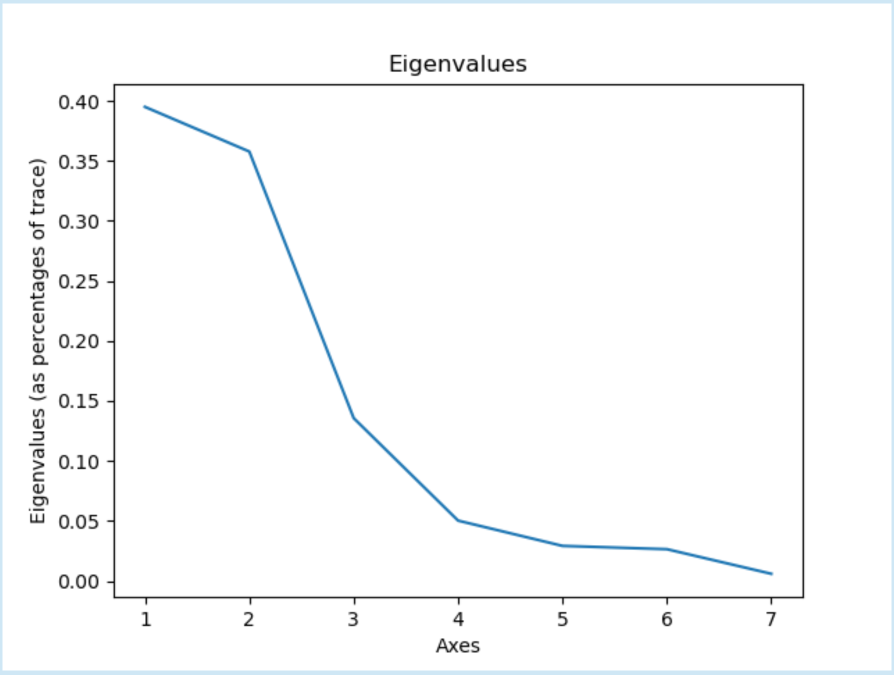

Figure 1. Series of eigenvalues

3) First visualizations

Selection of a pair of axes for visualizations.
For semiometry, axis 1 (size factor, for which most the variables are simultaneously positive or negative) is not taken into account. We will retain here the visualizations on axes 2 and 3.

```
ax_h, ax_v = 2, 3
```

Calling the grafact () function produces the factorial plan of active elements (individual and variables).

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

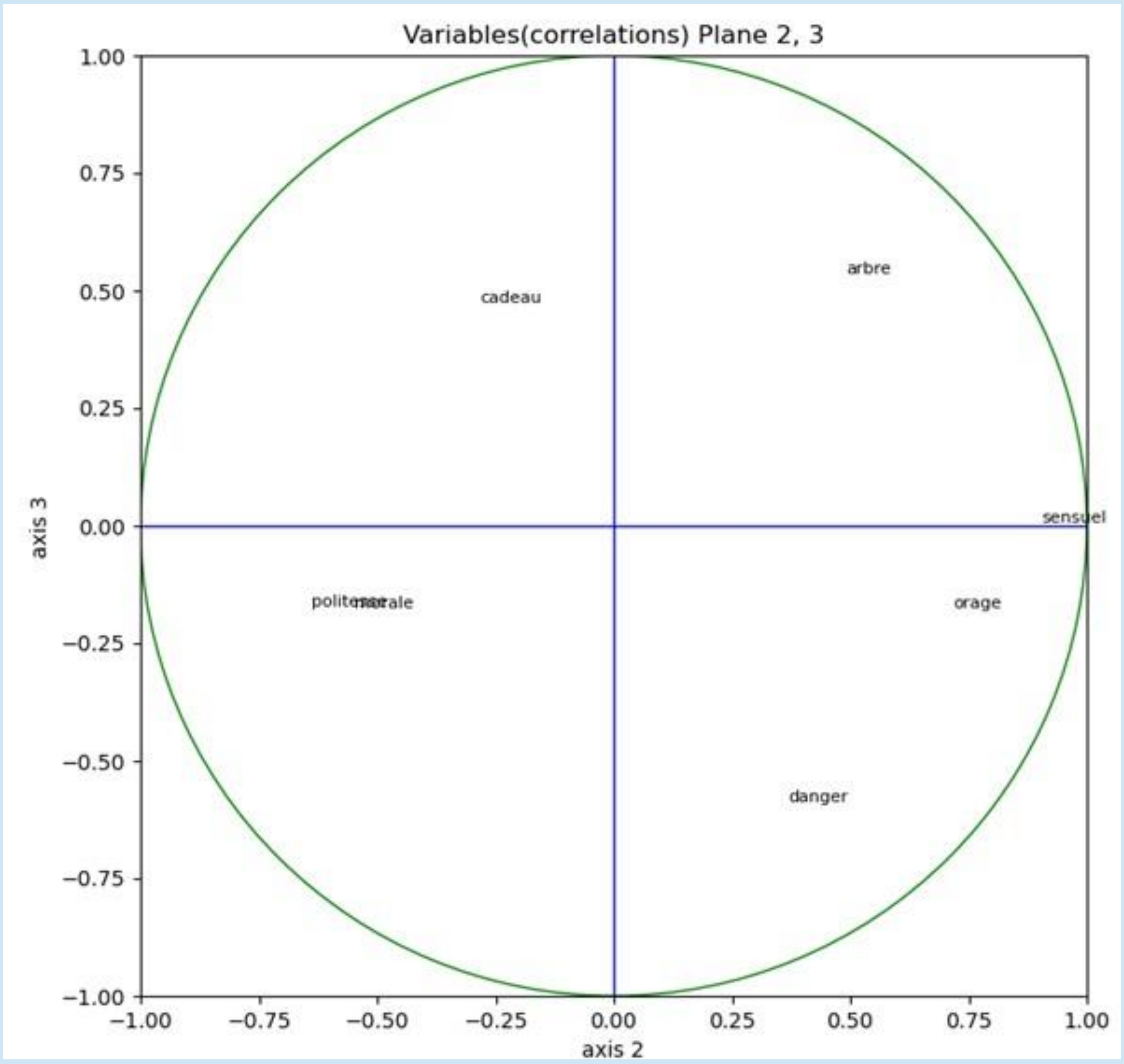

Figure 2. Position of variables in the plane 2, 3

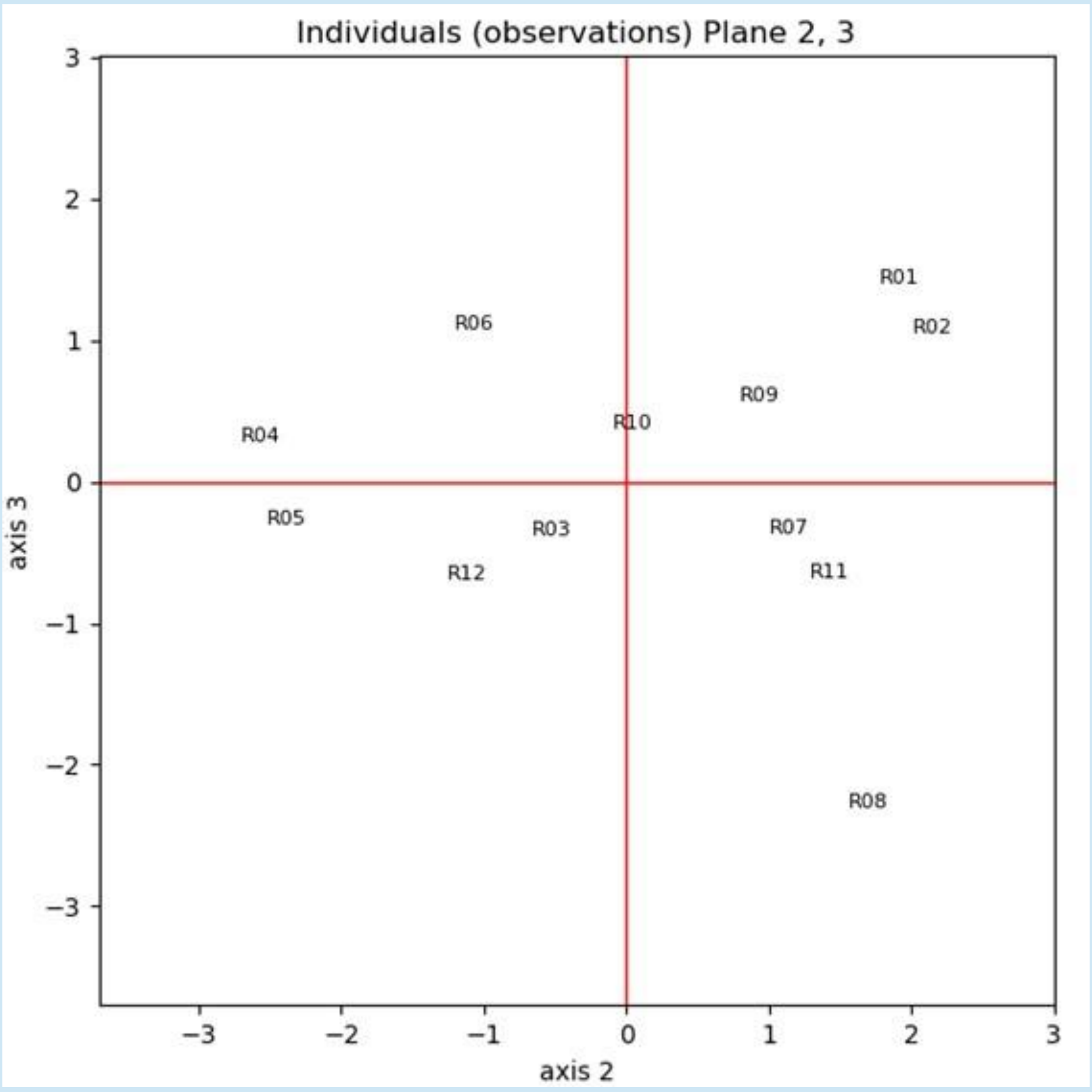

Figure 3. Position of individuals in the plane 2, 3

4) Additional individuals (or supplementary individuals)

All the individuals or observations (rows of the Mini_Sem_isup table) are represented by the call to the ACP_isup () function.

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

They can be identified on the plane (2, 3) of Fig. 4 by a slightly larger font and a blue color.

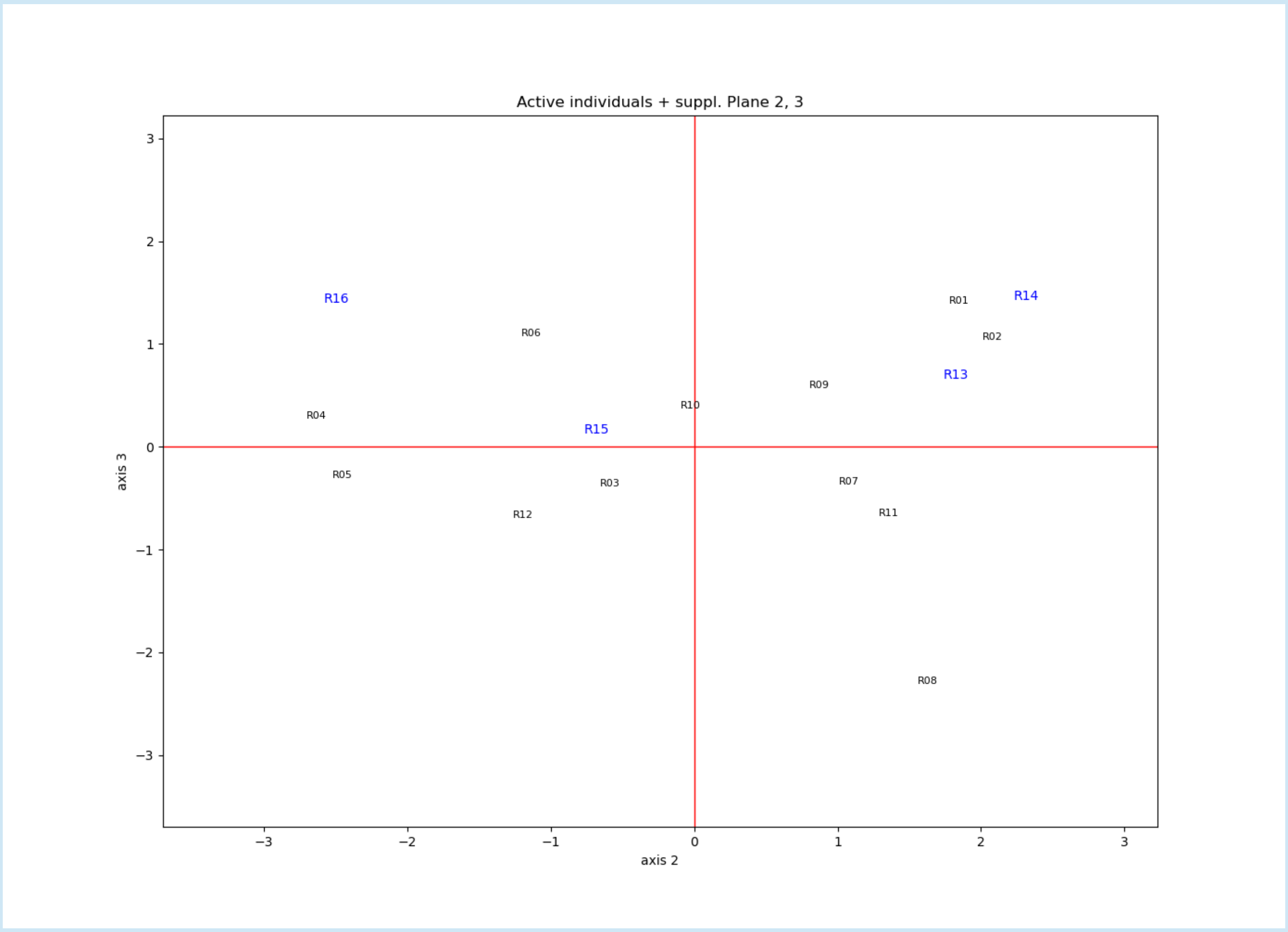

Figure 4. Position of additional individuals (R13 to R16) among active individuals in the plane (2, 3)

5) Supplementary numerical variables

For variables, it is necessary to distinguish between numerical variables and nominal (categorical) variables. The 3 numerical variables are the first columns... We will write:

vsup_lim = 3 #(Mini_Sem)

The first 3 additional numerical variables are taken into account by the ACP_vsup () function.

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

We obtain a brief and terse print of the coordinates of the additional variables on the 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]]
```

Figure 5 shows the plot of the additional variables.

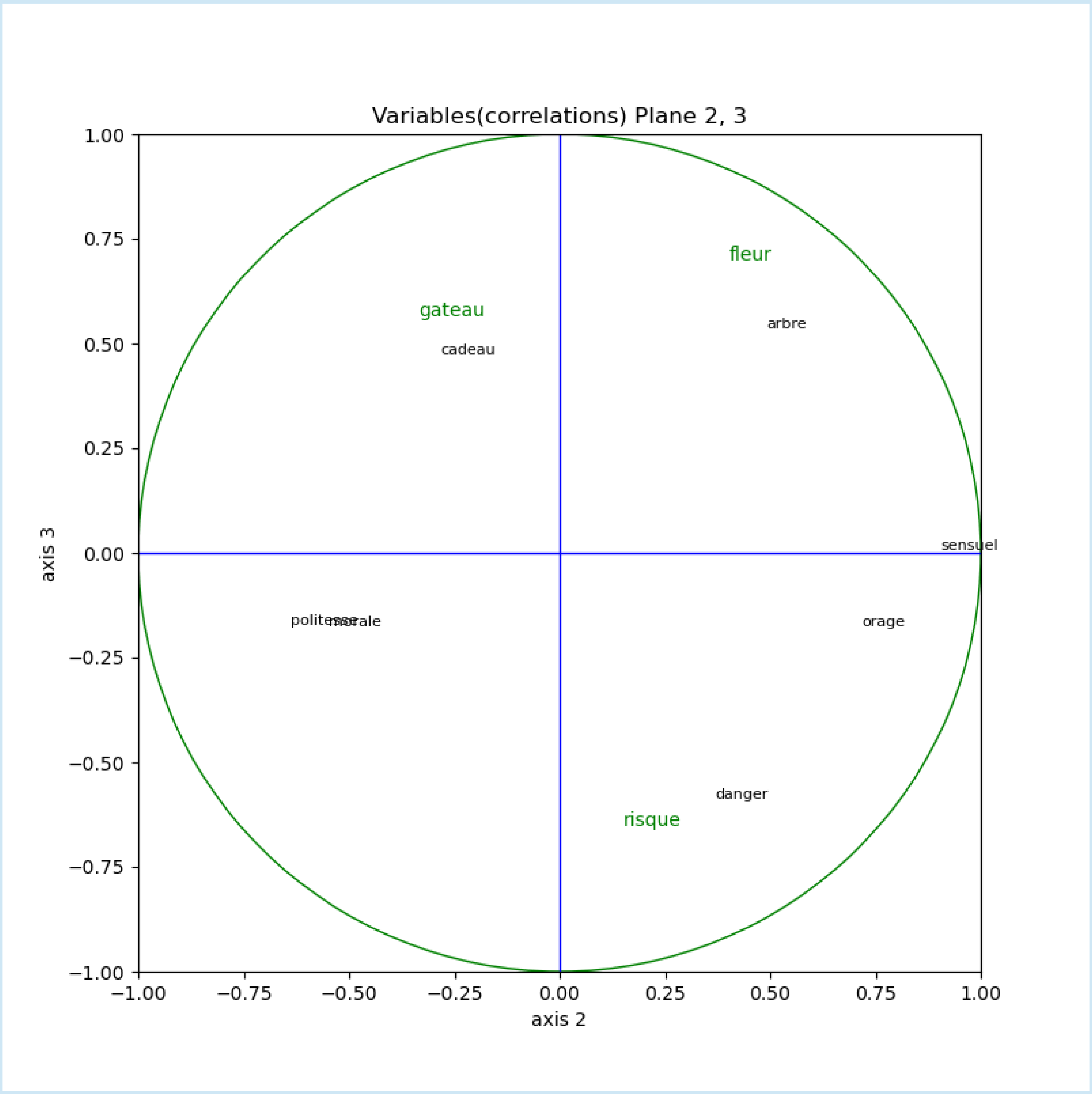

Figure 5. Position of the additional variables (risque, gâteau, fleur) (*risk*, *cake*, *flower*) among the active variables in plane (2, 3)

<u>6) Additional nominal (or categorical) variable</u>

They are chosen one by one because they give rise to a visualization that includes individuals.
The nominal variable vsup_nom is selected.

vsup_nom = 4 #(Mini_Sem)

The representation is done by the function ACP_nom ().

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

We finally get figure 6

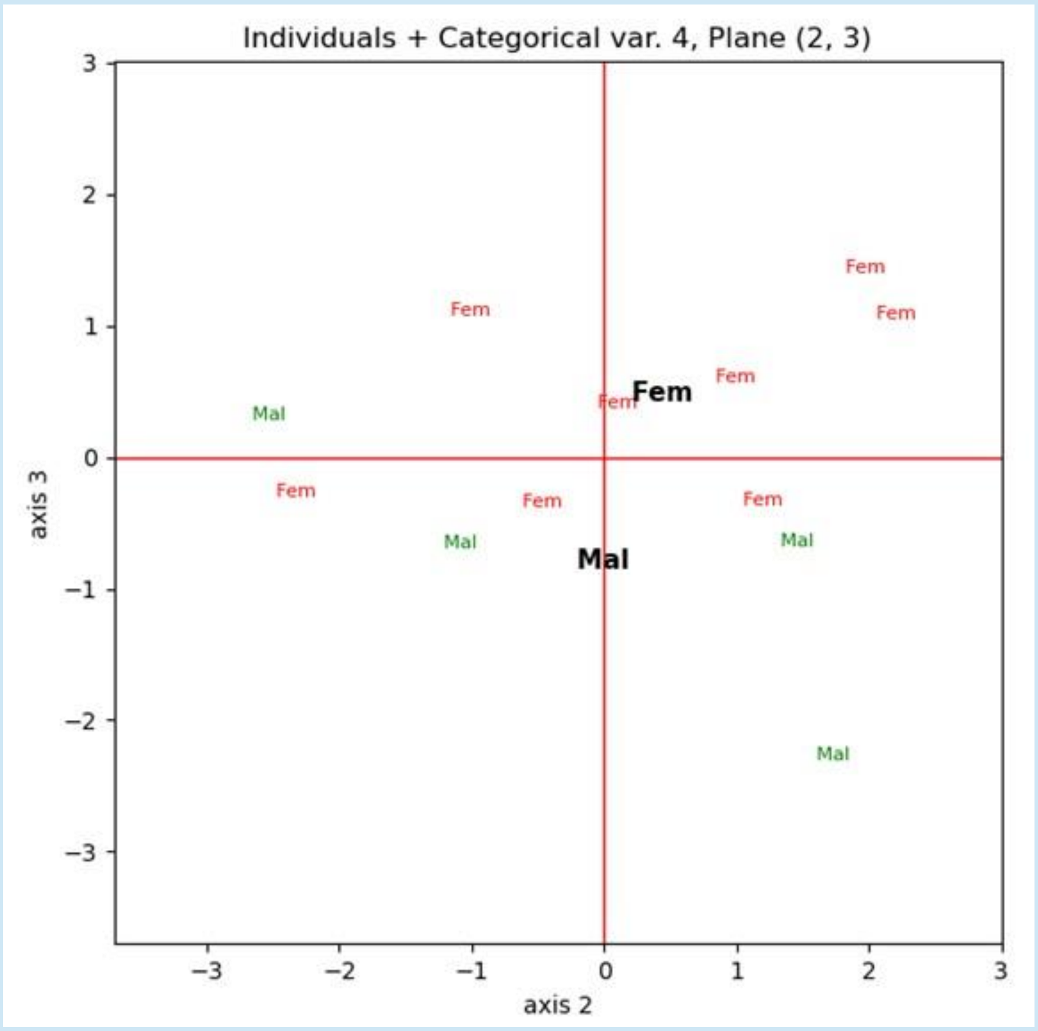

Figure 6. Position of the categories of the Supplementary nominal variable (Female and Male) among the active individuals (colored and identified by their category) in plane 2, 3.
The Supplementary categories Female and Male (black, bold) are at the average points of the individuals concerned.

Note:

These 5 function calls could have been replaced by the single call of the synthetic function ACP () appearing at the end of the code: (after defining the sequence of parameters in the parenthesis of the function):

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

…but you may want to represent other principal axes, other additional nominal variables, or select additional numerical variables ... the call by separate functions, heavier, is more flexible and gives more information about the structure of the program.

2. Practical implementation of the program with the example "Sem300"

The second, more realistic example, “Semio_300”, is implemented in a similar fashion.
The instructions to change are in the comments of the acomp_dtm.py file.
The three data files Sem300.txt, Sem300_isup.txt and Sem300_vsup should be downloaded, as well as the acomp_dtm.py code file in a working folder. The acomp_dtm.py file is included in section 3 of this note but is also a separate downloadable file like the data.

1. Preliminary instructions

Once the acomp_dtm.py code file has been copied into your working folder (here called "mydirectory2") with the three data files, all you have to do is type one by one in the python interface (like IDLE for example) the following 4 instructions to provide access to the program and data:

All of the instructions that follow are commented out in the acomp_dtm.py file, and can therefore simply be copied from this file.

The instructions in green are identical to those used for the "Mini_Sem" pedagogical example presented above. Only the 3 instructions in blue are different.

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

You just have to copy these lines directly from the acomp_dtm.py file that you will have opened in a free text editor (such as Notepad ++, which allows clear edits in python).

Here, the "mydirectory2" folder is directly in the "c: /" root (to be adapted for each user, who will replace “mydirectory2” by the path leading to her/his working folder).
The above import of acomp_dtm.py automatically loads the *pandas*, *numpy* and *matplotlib* libraries.

2. Basic PCA computations: “ACP_base ()” function

For the "Semio300" example, we will write in the python interface the 3 new access names on a single line:

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

For the data reading and the basic computations of the PCA, we will write again:

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

The following impressions (abbreviated) and graphics (figure A.1) are obtained on the 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
```

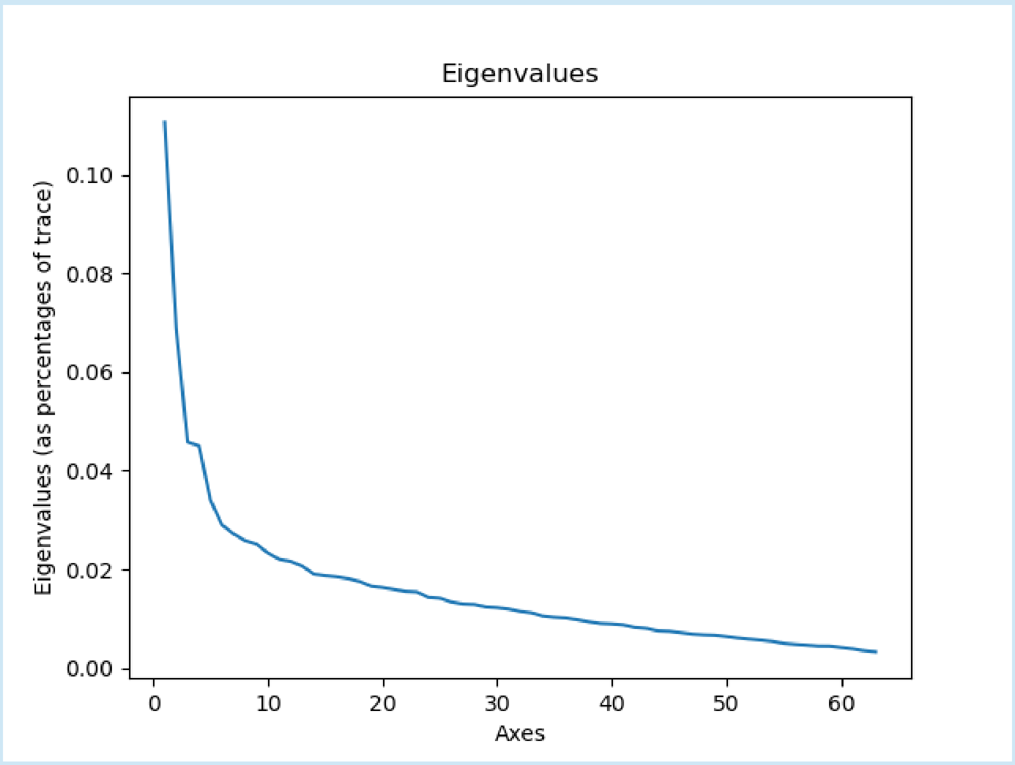

Figure A.1. Series of the 63 eigenvalues

3) First visualizations

Selection of a pair of axes for visualizations. For semiometry, axis 1 (size factor) is not taken into account. We will retain axes 2 and 3.

ax_h, ax_v = 2, 3

Calling the grafact () function produces the factorial plan of active elements (individual + variables)

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

The IDLE interface allows you to zoom in from a central rectangle inside the correlation circle for more readability.

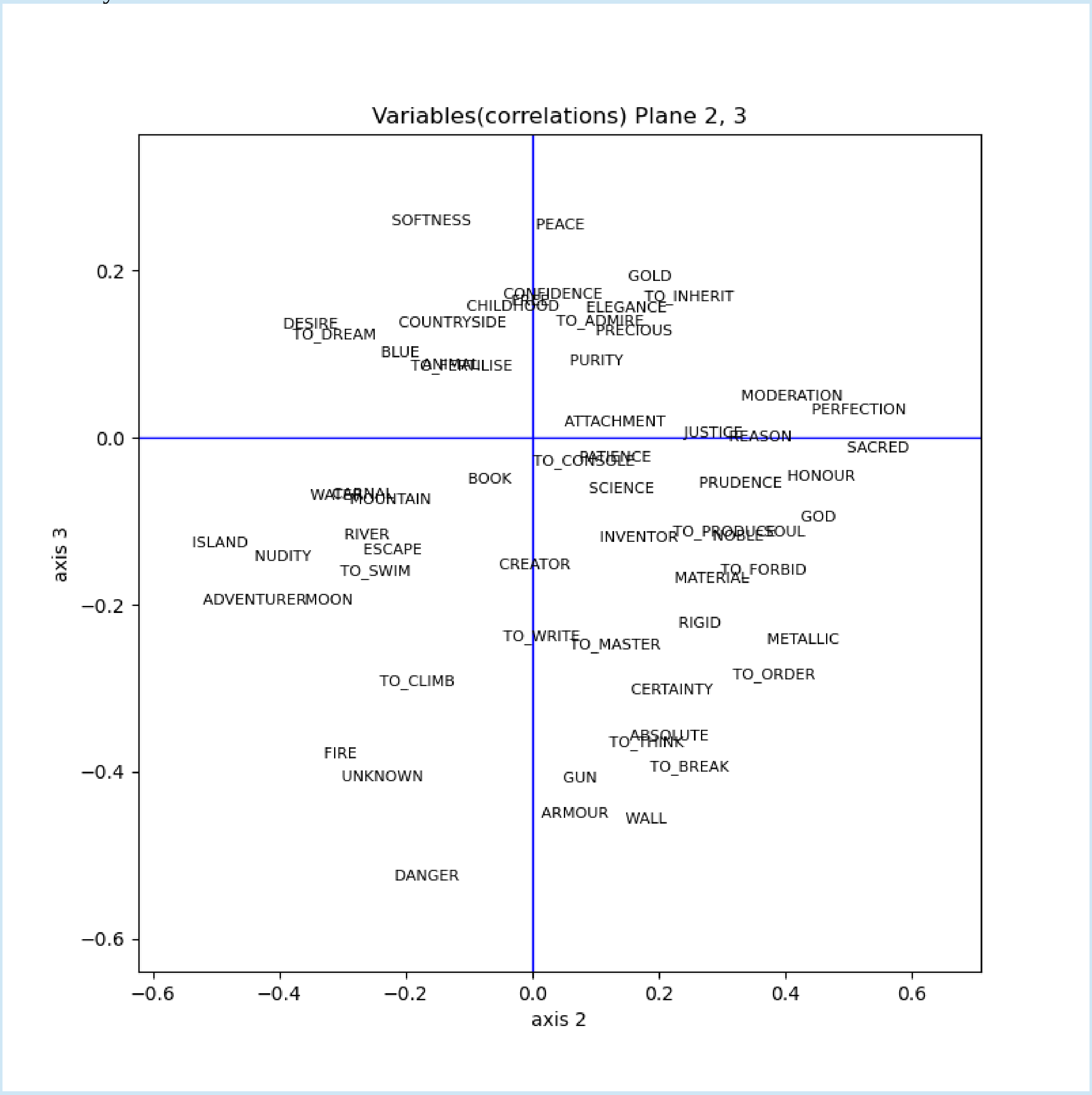

Figure A.2. Position of the 63 active variables in plane 2, 3 (zoom)

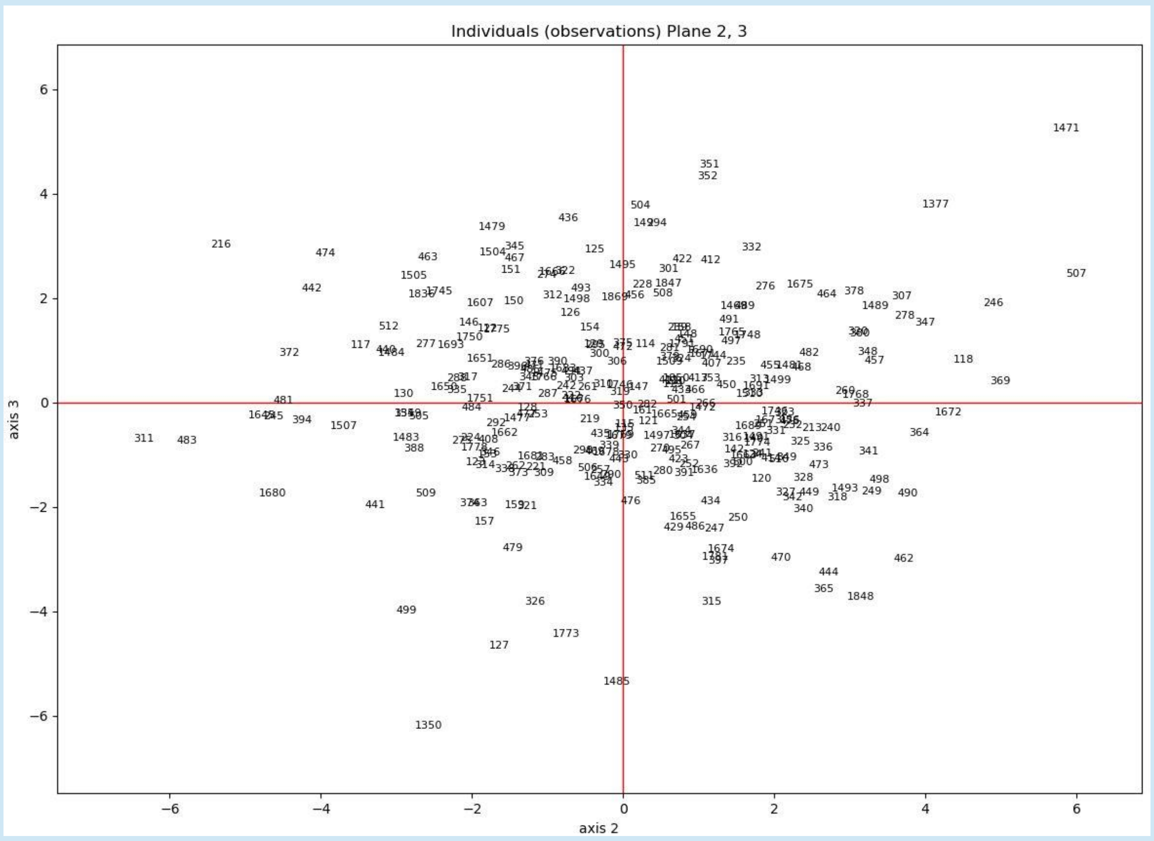

Figure A.3. Position of the 296 individuals in the plane (2, 3)

4) Additional (supplementary, illustrative) individuals

All the individuals (rows) of the *Mini_Sem_isup* array are represented by the ACP_isup () function.

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

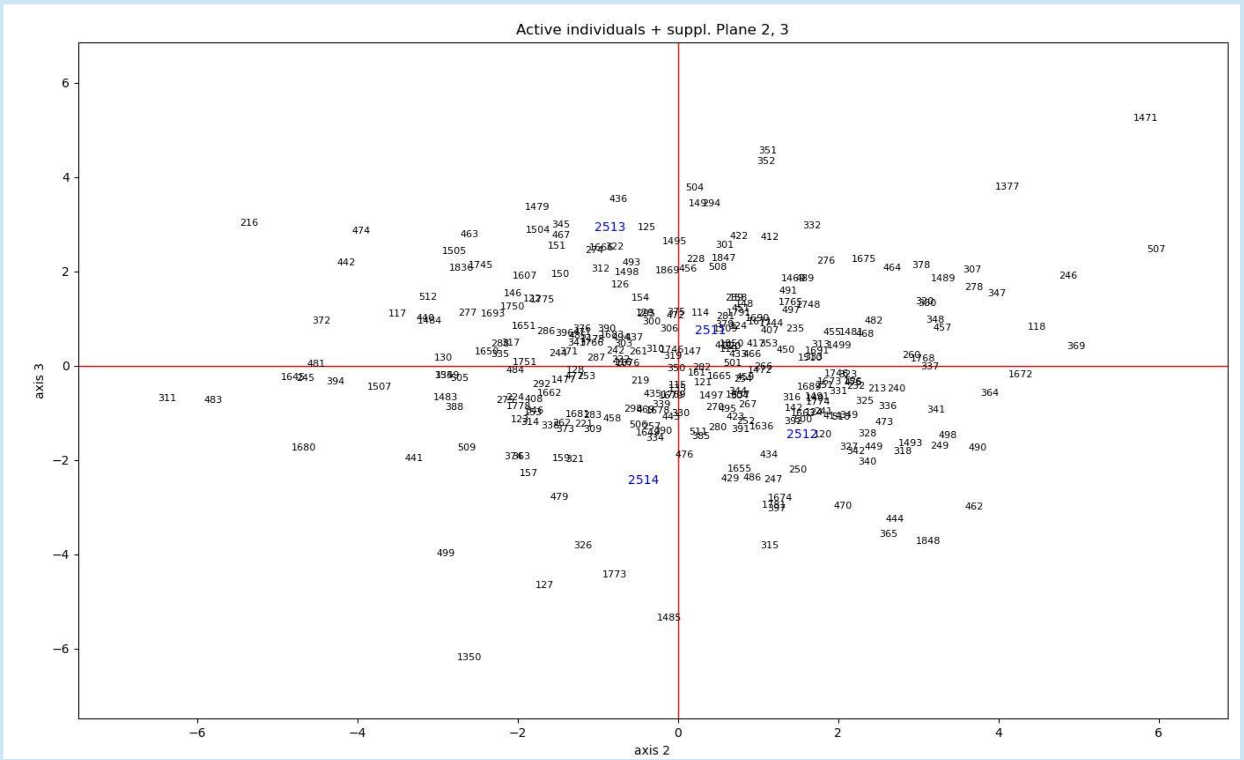

Figure A.4. Position of the 4 additional individuals (identifiers 2511 to 2514 in blue) among the active individuals in the same plane (2, 3). We could zoom in using the magnifying glass provided by the IDLE interface.

5) Supplementary numeric variables

For variables, it is necessary to distinguish between numerical variables and nominal variables.
The 7 numerical variables are printed before nominal variables. We will write: vsup_lim = 7
(Remember that for these examples, the numerical variables are scores assigned to words, while the additional categories are categories of respondents to the semiometric survey).

vsup_lim = 7 # (Sem300)

The first 7 additional numerical variables are taken into account by the ACP_vsup () function.

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

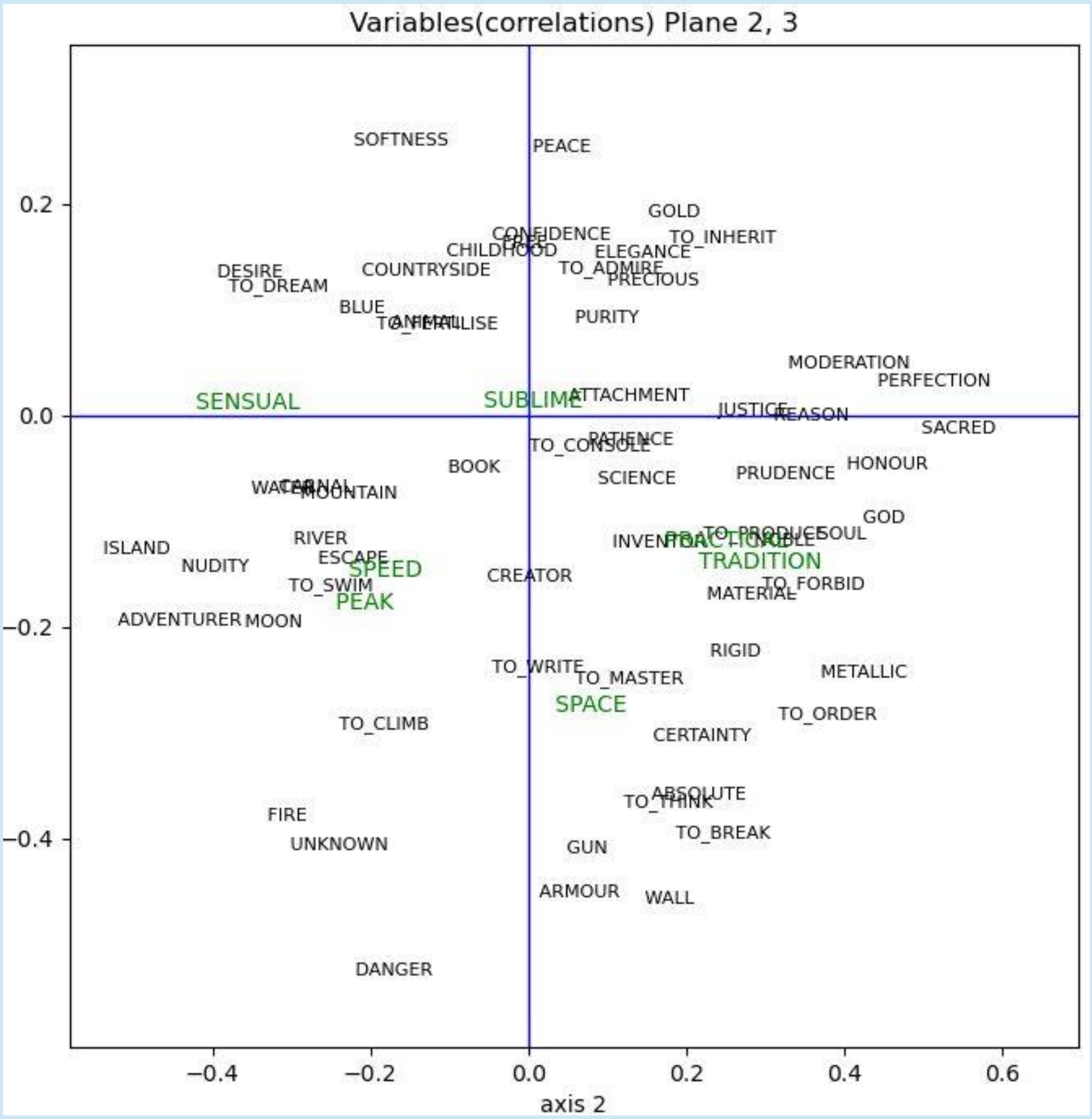

Figure A.5. Position of the 7 supplementary numerical variables () (`PEAK, PRACTICAL, SENSUAL, SPACE, SPEED, SUBLIME, TRADITION`) among the active variables of plan 2, 3 (*zoom*).

<u>6) Supplementary categorical (or: nominal) variable</u>

They are chosen one by one because they give rise to a simultaneous visualization of the individuals.

vsup_nom = 12 # (Sem300)

The nominal var vsup_nom is selected. The dummy variable 12 includes 9 categories (crossing of 3 age groups [–30, 30-55, +55] with 3 levels of education [low, medium, high].
The representation is always done by the function ACP_nom ().

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

We finally obtain figure A.6 (also obtained with the zoom of IDLE)

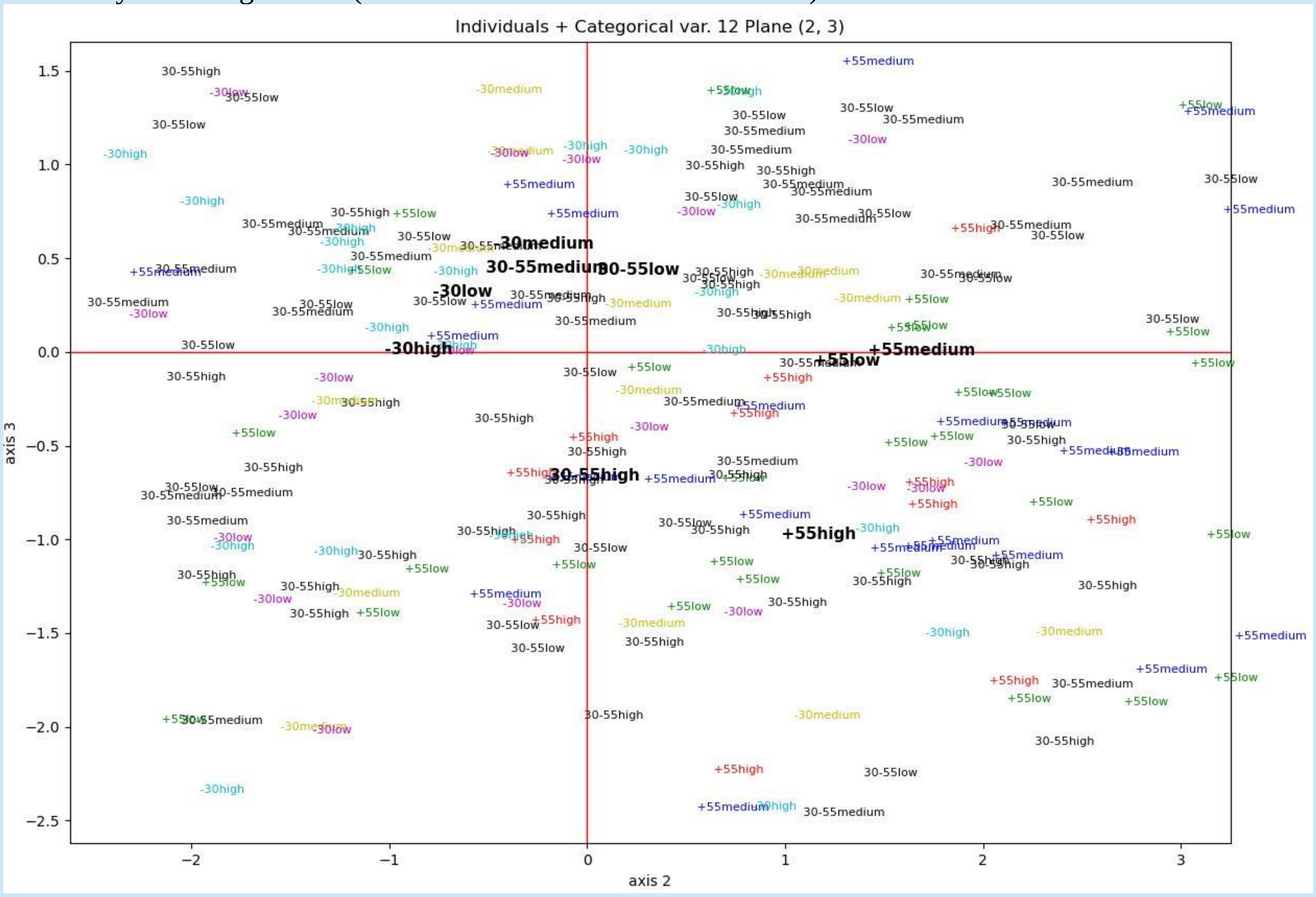

Figure A.6. Position of the 9 categories of the additional nominal variable 12 (age, crosstabulated with the level of education) among the active individuals (colored and identified by their category) in the plane (2, 3). The supplementary categories (black, bold) are the mean points of the individuals concerned.

Note (*reminder*):

These 5 function calls could have been replaced by the single call of the synthetic function ACP () appearing at the end of the code: `ACP (lane, lane_sup, lane_var_sup, ax_h, ax_v, vsup_lim, vsup_nom).` However, the call by separate functions is more flexible, and gives more information about the program.

3. Python code: Content of the source file: `acomp_dtm.py`

This file contains the instructions for use of the program in the form of comments (instructions to be entered in the interface).

```
#***-------------------------------------------------------
#***  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
#------------------------------------------------------------------------
```

End of python code

The code is written in the least cryptic way possible, in order to allow adaptations and additions, in particular concerning the output of the numerical results.
Evidently, python stylists can also make the code more compact and elegant.