Correspondence analysis with python

1. Practical implementation of the program (example "Words_Educ")

2. Content of the downloadable source file: "ac_dtm.py"

The table that we will try to describe with the help of correspondence analysis is a contingency table crossing, as rows, 14 words used in the answers to an open-ended question, and in column, 5 levels of education declared by each of the respondents.

The open question reads: *What are the reasons that, in your opinion, may make a woman or a couple hesitate to have a child?* (The survey was partially funded by the National Family Allowance Fund…). It was asked in 1981 to 2,000 people representing the population of residents in France aged 18 and over, in the CREDOC survey on the Living Conditions and Aspirations of the French. Four words and three additional categories (or illustrative) intervene *a posteriori* to enrich the interpretations.

This small introductory example is commented on in detail in Chapter 3 of the book "Exploring Textual Data" (Lebart, Salem and Berry, Kluwer Academic Publishers, 1998). An adapted excerpt of chapter 3 containing the example can be downloaded from: http://www.dtmvic.com/doc/EDT_chap3.pdf.
The whole book itself is not yet in the public domain.

Table 1
Cross-tabulation of words with Educational level. Raw frequencies (file: `Words_Educ_act.txt`)

Table 1 Cross-tabulation of words with Educational level. Raw frequencies

Words	No degree	Elem. Sch.	Trade Sch.	High Sch.	College	Total
Money	51	64	32	29	17	193
Future	53	90	78	75	22	318
Unemployment	71	111	50	40	11	283
Decision	1	7	5	5	4	22
Difficult	7	11	4	3	2	27
Economic	7	13	12	11	11	54
Selfishness	21	37	14	26	9	107
Occupation	12	35	19	6	7	79
Finances	10	7	7	3	1	28
War	4	7	7	6	2	26
Housing	8	22	7	10	5	52
Fear	25	45	38	38	13	159
Health	18	27	20	19	9	93
Work	35	61	29	14	12	151
Total	323	537	322	285	125	1592

The table is read as follows: the word *Money* for example, was used *51* times by persons belonging to the category "no degree". The row totals represent the number of occurrences of each word whereas the column totals represent the total number of words (within the list) used by the various categories of respondents.

Table 2

Four additional (or illustrative) lines ("Words_Educ_isup.txt" file)

Words	No degree	Elem. Sch.	Trade Sch.	High Sch.	Coll ege	Total
Comfort	*2*	*4*	*3*	*1*	*4*	*14*
Disagreement	*2*	*8*	*2*	*5*	*2*	*19*
World	*1*	*5*	*4*	*6*	*3*	*19*
Live	*3*	*3*	*1*	*3*	*4*	*14*

Table 3

Three supplementary (illustrative) columns (file : "Educ_vsup.txt")

Word	Age-30	Age-50	Age+50
Money	*59*	*66*	*70*
Future	*115*	*117*	*86*
Unemployment	*79*	*88*	*177*
Decision	*9*	*8*	*5*
Difficult	*2*	*17*	*18*
Economic	*18*	*19*	*17*
Selfishness	*14*	*34*	*61*
Occupation	*21*	*30*	*28*
Finances	*8*	*12*	*8*
War	*7*	*6*	*13*
Housing	*10*	*27*	*17*
Fear	*48*	*59*	*52*
Health	*13*	*29*	*53*
Work	*30*	*63*	*58*
Total	*433*	*575*	*663*

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 2). The program (python code) is in the file: ."**ac_dtm.py**".

1. Practical implementation of the program (example « Words_Educ »).

You should download the three previous data files (**Words_Educ_act.txt, Words_Educ_isup.txt, Words_Educ_vsup.txt**) as well as code file: : **ac_dtm.py** in a working folder. The contents of the **ac_dtm.py** python code file is published in section 2 of this note, but also constitute a separate downloadable file like the data.

1. 1. Preliminary instructions

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

All of the instructions that follow are commented out in the ac_dtm.py file, so they can simply be copied from there.

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

It suffices, remember, to copy these lines directly from the **ac_dtm.py** file that we will have opened in a free text editor (such as Notepad ++, which allows clear edits in python).

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 her/his working folder).

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

1. 2. Basic computation of CA : function AC_base( )

We will write in the python interface the 3 access names *on a single line:*

```
lane, lane_sup, lane_var_sup = "Words_Educ_act.txt",
"Words_Educ_isup.txt", "Words_Educ_vsup.txt"
```

To read the data and perform basic CA calculations, write:

```
X, psi, psi_u, phi, phi_u = AC_base(lane)
```

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

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

 Eigenvalues

[3.54015389e-02 1.31147352e-02 7.30111400e-03 6.24391345e-03
 2.04504338e-33]

Coordinates of columns

           ident   c_var_1   c_var_2   c_var_3
0      No_degree -0.209318  0.080727 -0.072630
1    Elem_School -0.138577 -0.056047  0.018139
2   Trade_School  0.108758  0.028483  0.147013
3    High_School  0.274039  0.121344 -0.076653
4        College  0.231233 -0.317858 -0.094188

 complete coord. and coord. of rows. are in the created file:
AC_file_Words_Educ_act.txt
```

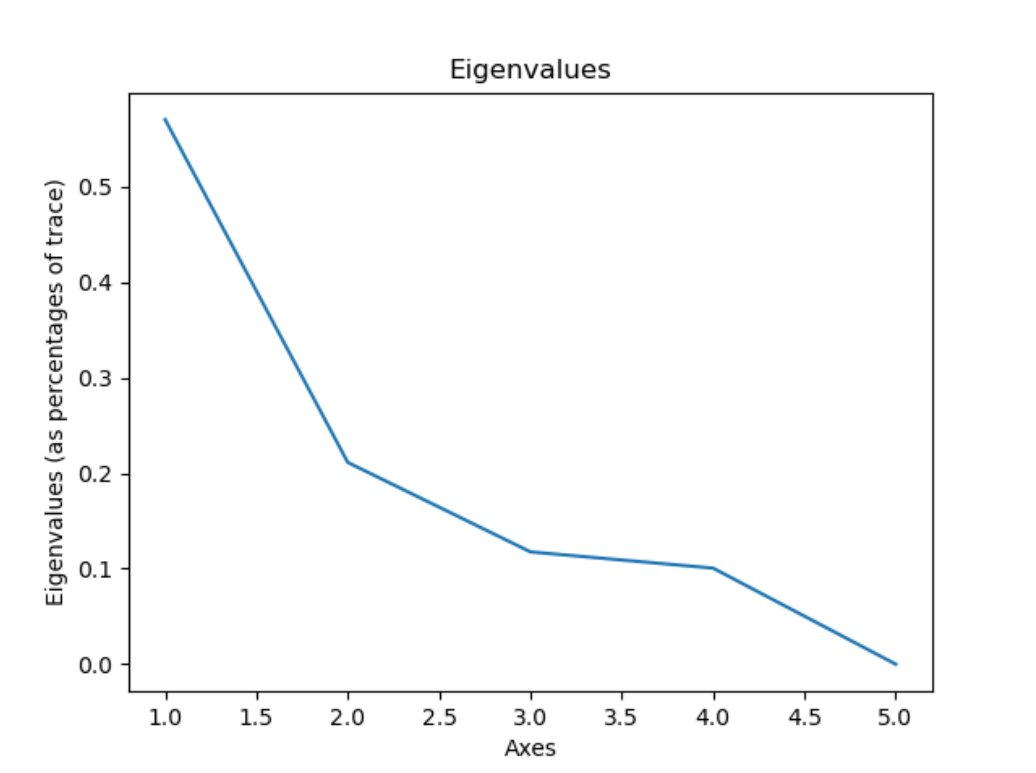

Figure 1. Series of eigenvalues

1. 3 First visualizations: active elements

Selection of a pair of axes for visualizations.
We will retain here the visualizations on axes 1 and 2.

```
ax_h, ax_v = 1, 2
```

Calling the graf_act () function produces the factorial plans of the active elements (rows/individual + columns/variables)

```
graf_act (X, psi, phi, ax_h, ax_v )
```

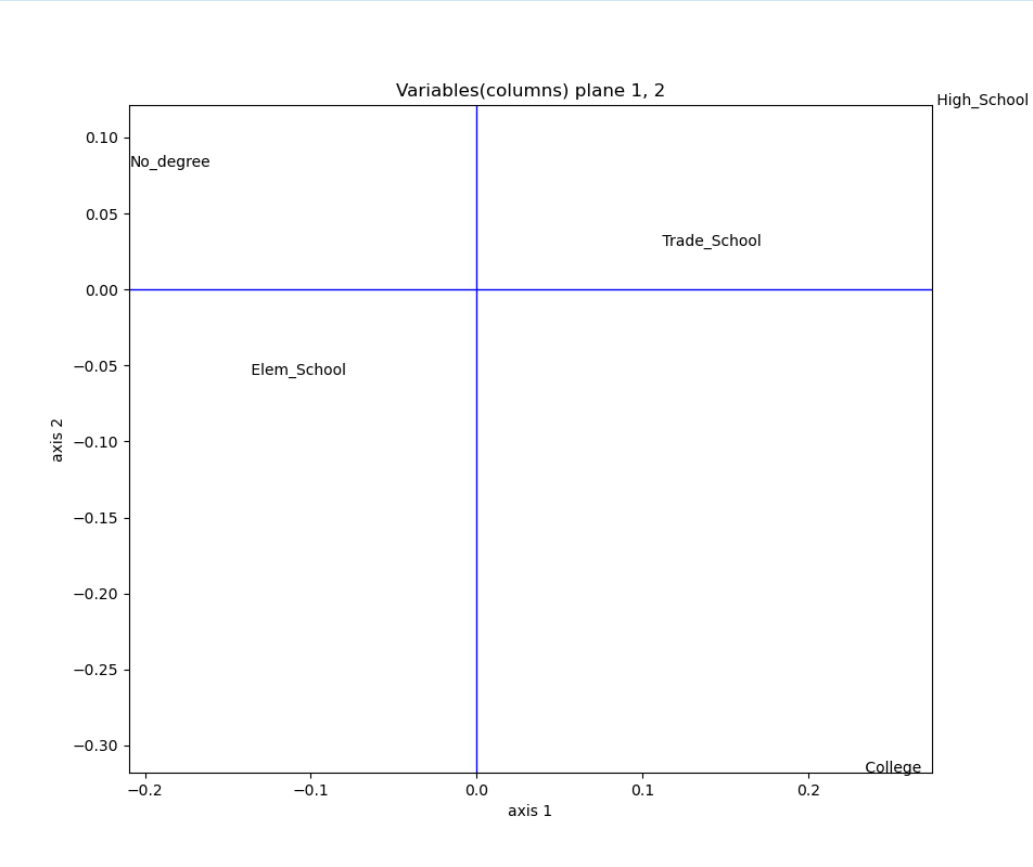

Figure 2. Position of columns (variables) in plane 1, 2

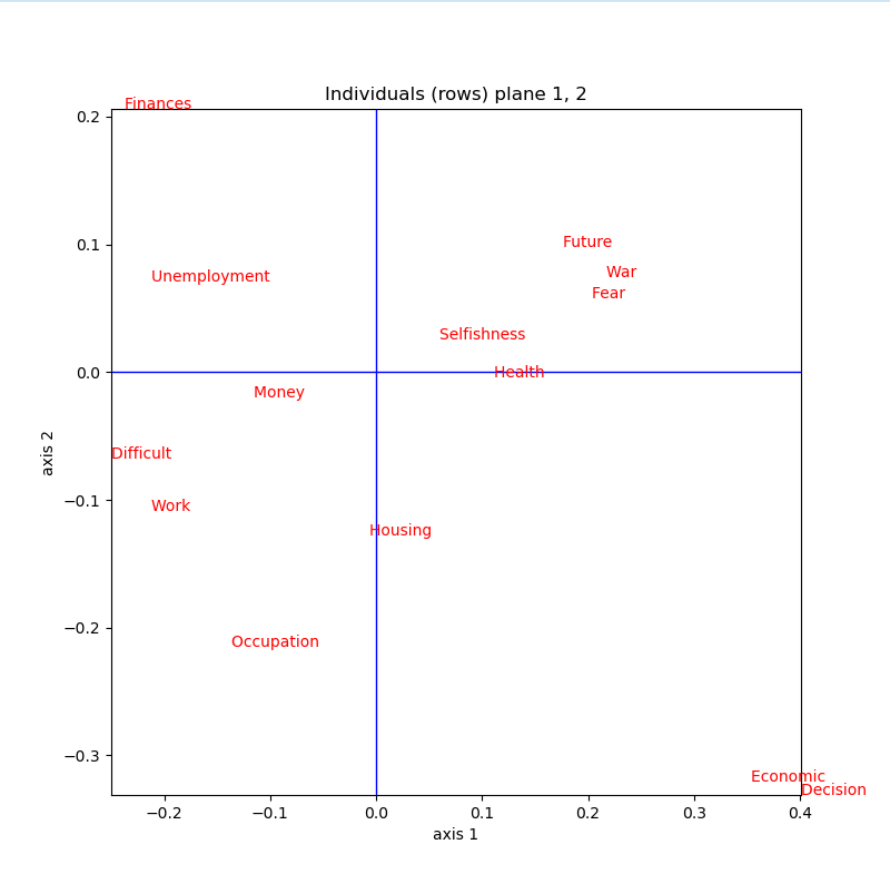

Figure 3. Position of rows (individuals) in the plane 1, 2

1. 4. Simultaneous representation of active rows and columns

Because of the possibilities of simultaneous representation of rows and columns in CA [and taking into account the specific interpretation rules] we may wish to merge figures 2 and 3, using the function:

```
graf_simult (X, psi, phi, ax_h, ax_v)
```

Remember that we cannot interpret the distance between one row-point and one column point. On the other hand, we can interpret the position of a point-line in relation to all the points-columns, and the position of a point-column in relation to all the points-lines.

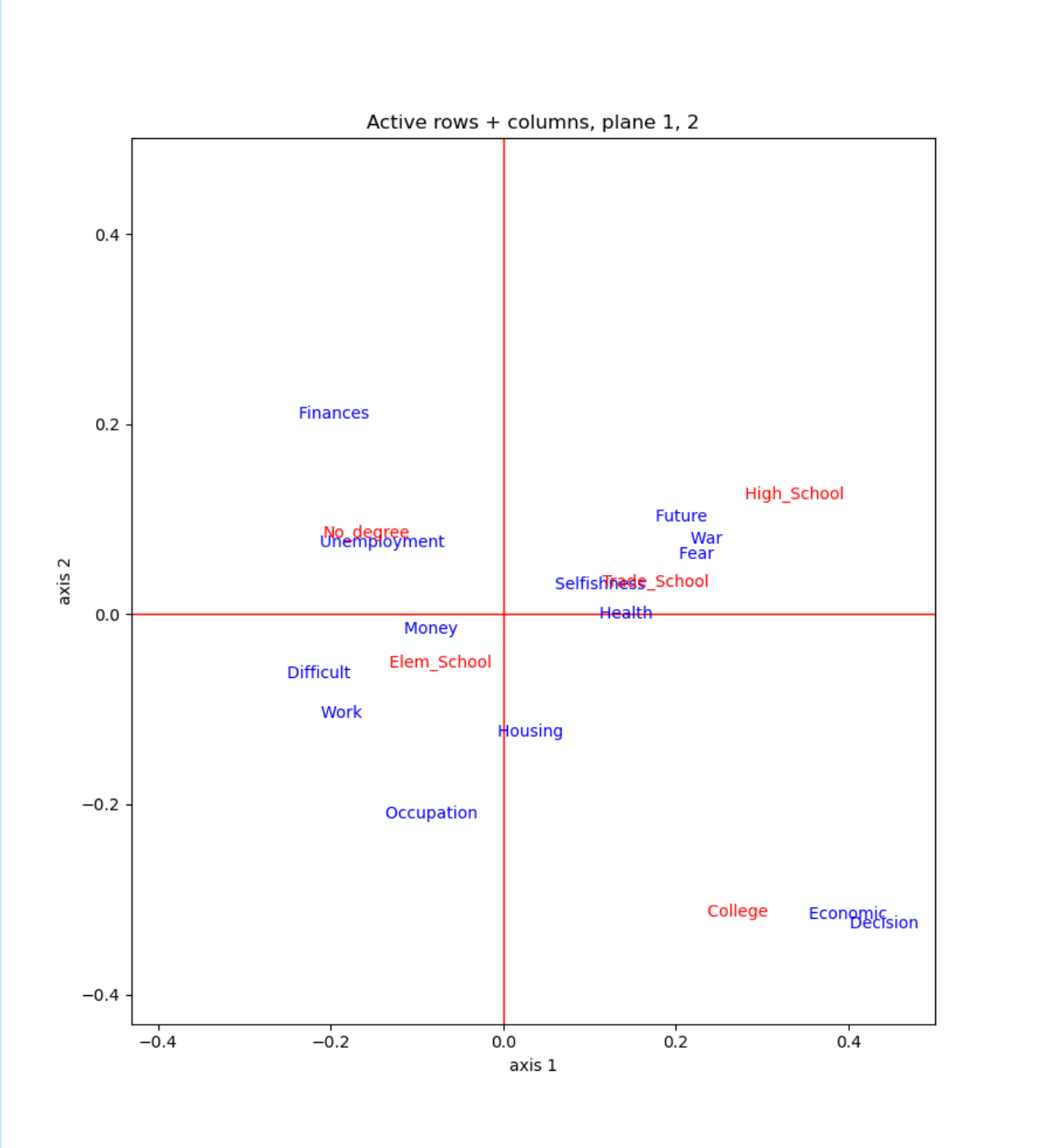

Figure 4. Simultaneous representation of rows (level of education) and columns (words) in the plane (1, 2)

1. 5. Supplementary rows

All the individuals or observations (rows of the table : **Words_Educ_isup.txt**) are represented by the call to the AC_isup () function **AC_isup ( ).**

```
AC_isup( X, psi,  phi, phi_u, lane_sup, ax_h, ax_v)
```

They can be identified on the plane (1, 2) of figure. 5 by a slightly larger font and a green color..

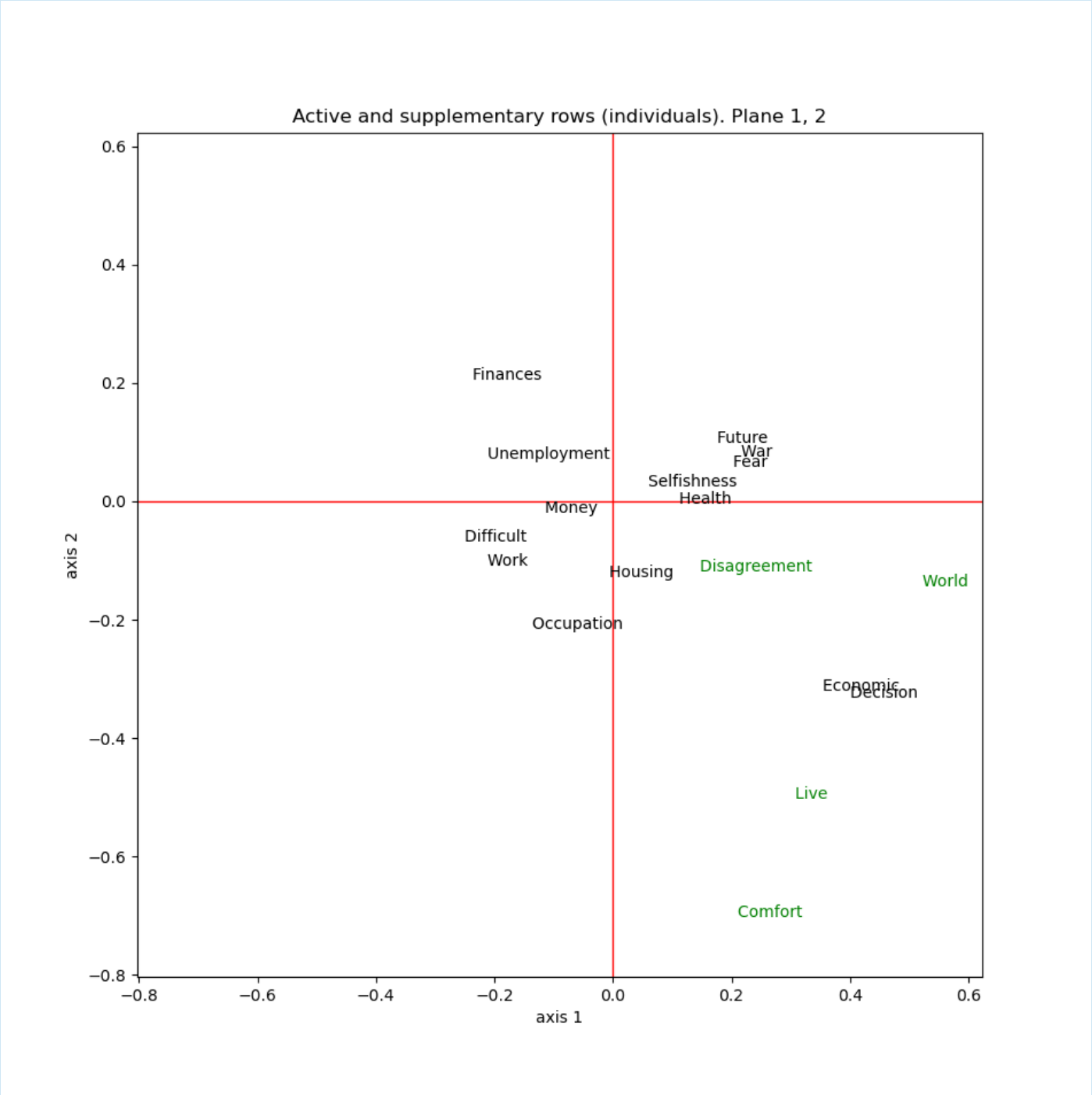

Figure 5. Representation of the 4 additional rows in the plane (1, 2)

1. 6. Supplementary columns

Additional columns (variables) are taken into account by the AC_vsup () function

```
AC_vsup(X, psi, psi_u, phi, lane_var_sup, ax_h, ax_v)
```

We obtain an impression (summary) of the coordinates of the additional columns on the axes..

```
[[ 0.10541339  0.05969594  0.10322613  0.06977996]
 [-0.01706444 -0.04907657  0.01568923 -0.01306117]
 [-0.1770681   0.04813788 -0.10077299 -0.08517528]]
```

Figure 5 shows the positions of the additional columns.

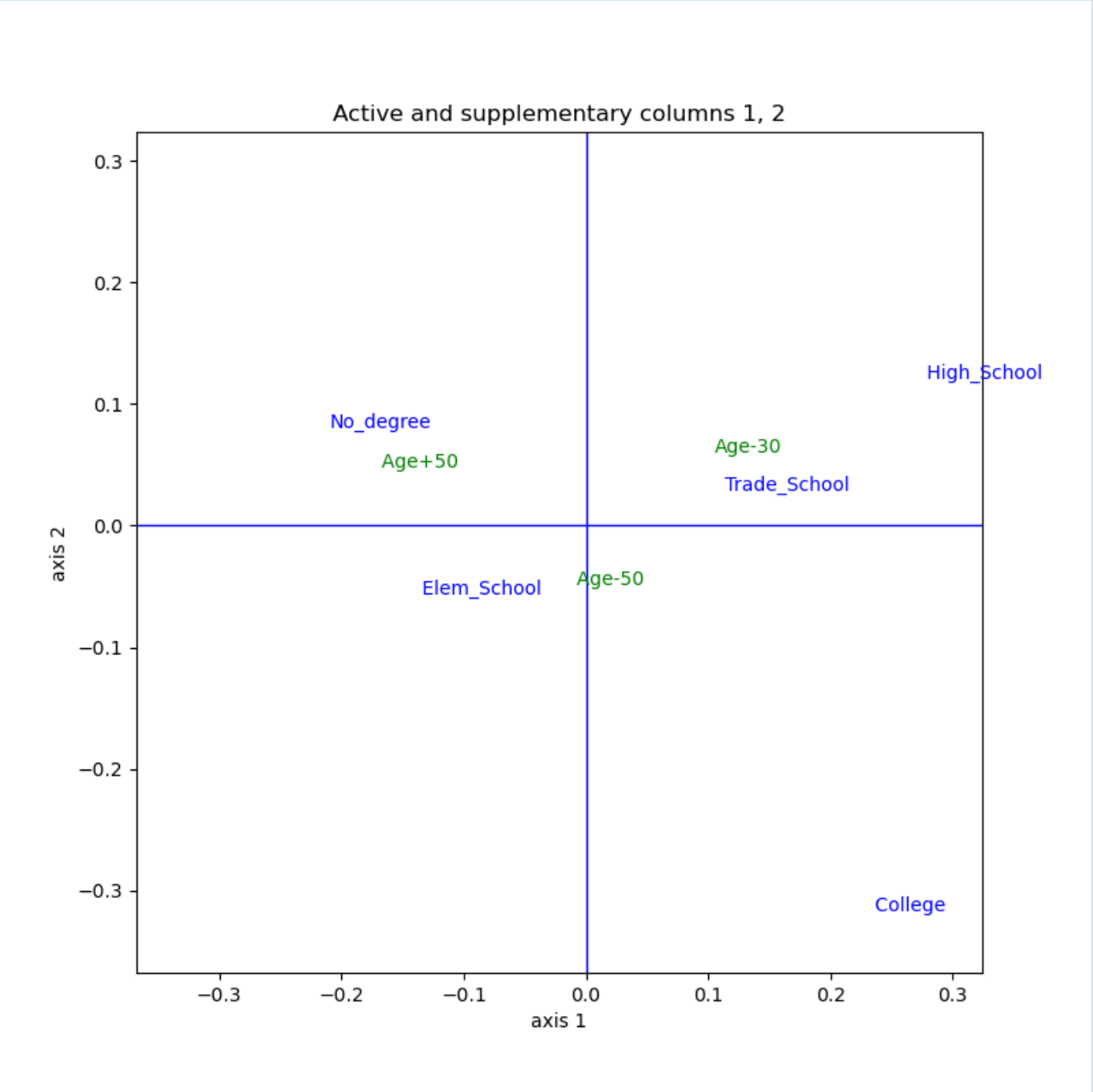

Figure 5. Position of additional columns (`Age-30, Age-50, Age+50`) among the active columns in plane 1, 2

Note :

These 5 function calls could have been replaced by the single call of the synthetic function AC () appearing at the end of the code:

```
AC (lane, lane_sup, lane_var_sup, ax_h, ax_v)
```

It is of course necessary to define beforehand the five parameters which appear in the above parenthesis.

2. Content of the downloadable source file: `ac_dtm.py`

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

```
#***------ ac_dtm.py  -------------------
#***  Here the source file: "acomp_dtm.py" and the data 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 ac_dtm
# from ac_dtm import *
#***-----------------------------------------------------
#*** Import statements for: numpy, pandas, matplotlib
#*** There is no need to copy these three lines: Automatic execution
#*** through the previous importation of: ac_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)

#*** Example "Words_Educ" : 3 paths for example: Words_Educ
#*** (lane : active elements, lane_sup: suppl. rows, lane_var_sup: suppl.
columns)
# lane, lane_sup, lane_var_sup = "Words_Educ_act.txt", "Words_Educ_isup.txt",
"Words_Educ_vsup.txt"
#***-----------------------------------------------------------
#*** CHOICE OF PARAMETERS
#***-----------------------------------------------------------
#*** SELECTION of a pair of axes for visualizations
#*** (ax_h: horisontal, ax_v: vertical)
#***-----------------------------------------------------------
# ax_h, ax_v = 1, 2
#
#*** CA EXECUTION
#***-----------------------------------------------------------
#*** 1) AC_base() : read basic data, computations
# X, psi, psi_u, phi, phi_u = AC_base(lane)
#
#*** 2) graf_act() : factorial plane of active elements
# graf_act (X, psi, phi, ax_h, ax_v )
#
#*** 3) Simultaneous visualization of rows and columns
# graf_simult (X, psi, phi, ax_h, ax_v)
#
#*** 4) AC_isup() Supplementary rows
# AC_isup( X, psi,  phi, phi_u, lane_sup, ax_h, ax_v)
#
#*** 5) AC_vsup() Supplementary columns
# AC_vsup(X, psi, psi_u, phi, lane_var_sup, ax_h, ax_v)
#
#***-----------------------------------------------------------
#*** These 5 calls can be replaced by the single call of the function AC():
# AC (lane, lane_sup, lane_var_sup, ax_h, ax_v)
#*** --------
#***  However we may want to represent other visualization planes,
#***  to adapt some outputs, to add other supplementary rows or columns ...
#***  the call by separate functions is more transparent and 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:
#***  supplementary <=> additional <=> illustrative
#***  rows <=> individuals, observations
#***  columns <=> variables
#***-----------------------------------------------------
#
#*** Function AC_base(): computations, eigenvalues, eigenvectors...
#
def AC_base(lane):
     X, F, psi, psi_u, phi, phi_u, pcent, vs, vp = AC_calc(lane)
     n,p = X.shape
     grafac_vp(pcent, p)
```

```
#---
      ch ="AC_file_" + lane    # name for the created result 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': phi[:,0], 'c_var_2':
phi[:,1],'c_var_3': phi[:,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': phi[:,0], 'c_var_2':
phi[:,1],'c_var_3': phi[:,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_ind_1': psi[:,0], 'c_ind_2':
psi[:,1],'c_ind_3': psi[:,2],  })
      saa = str(aa)
      g.write(saa)
      g.close()
      pd.reset_option('display.max_rows')
      return X, psi, psi_u, phi, phi_u
#-----------------------------------------------------
#
Function AC_calc() : basic computations
def AC_calc(lane):
      X = pd.read_csv(lane)
      n,p = X.shape
      ki = np.sum(X, axis = 1)
      kj = np.sum(X, axis = 0)
      k = np.sum(kj)
      fi, fj = ki/k, kj/k
      F = X/k
#---
      S = np.zeros((n,p))
      for i in range(n):
            for j in range(p):
```

```
                S[i,j] = (F.iloc[i,j] -fi[i]*fj[j])/np.sqrt(fi[i]*fj[j])
#---  np.linalg.svd = singular values decomposition with numpy
     u, vs, tvh = np.linalg.svd(S, full_matrices=False)
     rang = min(n,p) -1   # # CA entails at least one zero eigenvalue
     vp = vs*vs
#
     psi=np.zeros((n,rang))
     psi_u=np.zeros((n,rang))
# coordinates psi des rows (psi_u: vector of norm 1)
     for j in range(rang):
          for i in range(n):
               psi_u [i,j] = u[i,j]/np.sqrt(fi[i])
               psi [i,j] = u[i,j]*vs[j]/np.sqrt(fi[i])
#
     tphi = np.zeros((rang,p))   # coordinates of columns
     tphi_u = np.zeros((rang,p)) # unitary vect. of columns
     for jj in range(rang):
          for j in range(p):
               tphi_u[jj,j] = tvh[jj,j]/np.sqrt(fj[j])
               tphi[jj,j] = tvh[jj,j]*vs[jj]/np.sqrt(fj[j])
#
     trace = np.sum(vp)
     pcent = vp/trace
# tphi et tphi_u are transposed into phi and phi_u
     phi = tphi.T
     phi_u = tphi_u.T
     return X, F, psi, psi_u, phi, phi_u, pcent, vs, vp
#-----------------------------------------------------
#*** Function grafac_vp() :Eigenvalues graph
def grafac_vp(pcent, p):
     plt.plot(np.arange(1, p+1), pcent)
     plt.title("Eigenvalues")
     plt.xlabel("Axes")
     plt.ylabel("Eigenvalues (as percentages of trace)")
     plt.show()
#-----------------------------------------------------
#*** Function graf_act(): Graphics (plane (ax_h, ax_v) of active elements
def graf_act ( X, psi, phi, ax_h, ax_v ):
     xx = ax_h
     yy = ax_v
     n,p = X.shape
#---
     graf_col (X, phi, p, xx, yy)
     graf_rows (X, psi,  n, xx, yy)
     return
#-----------------------------------------------------
#*** Function graf_col(): Plot (plane (ax_h, ax_v)) of columns
def graf_col ( X, phi, p, ax_h, ax_v):
     lax = 8
     xh = phi[:,ax_h - 1]
     xv = phi[:,ax_v - 1]
     fig, axes = plt.subplots(figsize = (lax,lax))
     mi_x, ma_x = min(xh), max(xh)
     axes.set_xlim (mi_x,ma_x)
     mi_y, ma_y = min(xv), max(xv)
     axes.set_ylim (mi_y,ma_y)
     FS = 10    # FS = fontsize
     for j in range(p):
```

```
          plt.annotate (X.columns[j],(phi[j,ax_h - 1], phi[j,ax_v - 1]),
fontsize = FS)
     plt.plot([mi_x,ma_x],[0,0],color = 'blue',linestyle = "-", linewidth = 1)
     plt.plot([0,0],[mi_y,ma_y],color = 'blue',linestyle = "-", linewidth = 1)
     plt.title("Variables(columns) plane "+ str(ax_h)+ ", "+ str(ax_v))
     plt.xlabel("axis " + str(ax_h))
     plt.ylabel("axis " + str(ax_v))
     plt.show()
#-------------------------------------------------------
#*** Function  graf_rows() : Plots (plane (ax_h, ax_v)) of rows
def graf_rows ( X, psi, n, ax_h, ax_v):
#--- Frame
     xh = psi[:,ax_h - 1]
     xv = psi[:,ax_v - 1]
     lax = 8
     fig, axes = plt.subplots(figsize = (lax,lax))
     FS = 8   # FS = fontsize
     mi_x, ma_x = min(xh), max(xh)
     axes.set_xlim (mi_x,ma_x)
     mi_y, ma_y = min(xv), max(xv)
     axes.set_ylim (mi_y,ma_y)
     for i in range(n):
          plt.annotate (X.index[i],(psi[i,ax_h - 1], psi[i,ax_v - 1]),
fontsize = FS, color = 'red')
     plt.plot([mi_x, ma_x],[0,0], color = 'blue', linestyle = "-",linewidth =
1)
     plt.plot([0,0],[mi_y, ma_y ], color = 'blue', linestyle = "-",linewidth =
1)
     plt.title("Individuals (rows) plane " + str(ax_h)+ ", "+  str(ax_v))
     plt.xlabel("axis " + str(ax_h))
     plt.ylabel("axis " + str(ax_v))
     plt.show()
#-------------------------------------------------------
#*** Function graf_simult() : simultaneous plot of rows and columns(plane
(ax_h, ax_v) )
def graf_simult (X, psi, phi, ax_h, ax_v):
     n,p = X.shape
#--- Frame (same scales)
     xh = psi[:,ax_h - 1]
     xv = psi[:,ax_v - 1]
     xh_var = phi[:,ax_h - 1]
     xv_var = phi[:,ax_v - 1]
#--- a to widen the frame
     lax = 8
     a = 0.1
     FS_ind = 8  # FS = fontsize
     FS_var = 10
     lmin = min(min(xv), min(xh),min(xv_var), min(xh_var)) - a
     lmax = max(max(xv), max(xh), max(xv_var), max(xh_var)) + a
#---
     fig, axes = plt.subplots(figsize = (lax,lax))
     axes.set_xlim (lmin,lmax)
     axes.set_ylim(lmin,lmax)
     for i in range(n):       # active rows
          plt.annotate (X.index[i],(psi[i,ax_h - 1], psi[i,ax_v - 1]),
fontsize = FS_ind, color = 'b')
     for j in range(p):
          plt.annotate (X.columns[j],(phi[j,ax_h - 1], phi[j,ax_v - 1]),
fontsize = FS_var, color ='r')
```

```
#--- drawing 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 rows + columns, plane " + str(ax_h)+ ", "+  str(ax_v))
     plt.xlabel("axis " + str(ax_h))
     plt.ylabel("axis " + str(ax_v))
     plt.show()
#-------------------------------------------------------

#*** Function AC_isup() : coordinates and plot of supplementary rows
def AC_isup( X, psi,  phi, phi_u, lane_sup, ax_h, ax_v):
#--- Data and coord. of supplementary rows
     X_isup, psi_isup = rowsup(lane_sup, phi_u)
#--- Graphical display for suppl. rows (together with active rows)
     graf_rows_sup (X, psi, psi_isup,  X_isup, ax_h, ax_v)
     return
#-------------------------------------------------------
#*** Function rowsup(): Data and coord. of supplementary rows:
def rowsup(lane_sup, phi_u):
     X_isup = pd.read_csv(lane_sup)
     q,p = X_isup.shape
     X_isup_norm = np.zeros((q,p))
     for i in range(q):
          for j in range(p):
               X_isup_norm [i,j] = X_isup.iloc[i,j] /sum(X_isup.iloc[i,:])
     psi_isup = np.dot(X_isup_norm, phi_u)
     print("\n")
     print(psi_isup)
     return X_isup, psi_isup
#-------------------------------------------------------

#*** Function graf_rows_sup() : plot of both suppl. rows and active rows
def graf_rows_sup (X, psi, psi_isup, X_isup, ax_h, ax_v):
#
     xh = psi[:,ax_h - 1]
     xv = psi[:,ax_v - 1]
     xh_sup = psi_isup[:,ax_h - 1]
     xv_sup = psi_isup[:,ax_v - 1]
#--- Frame (same units) ,"a" to widen the frame a = 0.1
     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 = 8   # size of display
#---
     n = X.shape[0]           # number of active rows
     fig, axes = plt.subplots(figsize = (lax,lax))
     axes.set_xlim (lmin,lmax)
     axes.set_ylim(lmin,lmax)
     for i in range(n):       # active rows
          plt.annotate (X.index[i],(psi[i,ax_h - 1], psi[i,ax_v - 1]),
fontsize = FS)
     nsup = X_isup.shape[0]  #number of suppl. rows
     for i in range(nsup):
          plt.annotate (X_isup.index[i],(psi_isup[i,ax_h - 1], psi_isup[i,ax_v
- 1]), color = 'g')
#--- drawing 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 and supplementary rows (individuals). Plane " +
str(ax_h)+ ", "+  str(ax_v))
      plt.xlabel("axis " + str(ax_h))
      plt.ylabel("axis " + str(ax_v))
      plt.show()
#-------------------------------------------------------
#*** Function AC_vsup(): supplementary columns
def AC_vsup(X, psi, psi_u, phi, lane_var_sup, ax_h, ax_v):
#--- Data and computation
      X_vsup, phi_vsup, q = colsup(X, psi_u, lane_var_sup)
      if (colsup == 0):
            return
#---Supplementary columns (variables) graphics
      graf_col_sup (X, phi, X_vsup, phi_vsup, ax_h, ax_v)
#-------------------------------------------------------
#*** Function colsup() : Supplementary columns, data and computation
def colsup(X, psi_u, lane_var_sup):
      n, r = psi_u.shape
      X_vsup = pd.read_csv(lane_var_sup)
      n,q = X_vsup.shape #  q = number of suppl. columns
      X_vsup_norm = np.zeros((n,q))
      for j in range(q):
            for m in range(n):
                  X_vsup_norm [m,j] = X_vsup.iloc[m,j] /sum(X_vsup.iloc[:,j])
      phi_vsup = np.dot(X_vsup_norm.T, psi_u)
      print("\n")
      print(phi_vsup)
      return X_vsup, phi_vsup, q
#-------------------------------------------------------
#*** Function graf_col_sup() : Graphics (plane (ax_h, ax_v) ) for suppl.
columns
def graf_col_sup (X, phi, X_vsup, phi_vsup,  ax_h, ax_v):
      lax = 8
      q = X_vsup .shape[1]  # number of suppl. columns
      p = X.shape[1]        # number of active columns

      fig, axes = plt.subplots(figsize = (lax,lax))

#--- Frame : same scales
      xh = phi[:,ax_h - 1]
      xv = phi[:,ax_v - 1]
      xh_sup = phi_vsup[:,ax_h - 1]
      xv_sup = phi_vsup[:,ax_v - 1]
#
      a = 0.05
      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 = 8
#---
      axes.set_xlim (lmin,lmax)
      axes.set_ylim(lmin,lmax)
#--- active columns
      for j in range(p):
            plt.annotate (X.columns[j],(phi[j,ax_h - 1], phi[j,ax_v - 1]),
fontsize = FS, color = 'b')
#--- supplementar columns
      for j in range(q):
```

```
          plt.annotate (X_vsup.columns[j],(phi_vsup[j,ax_h - 1],
phi_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("Active and supplementary columns "+ str(ax_h)+ ", "+
str(ax_v))
     plt.xlabel("axis " + str(ax_h))
     plt.ylabel("axis " + str(ax_v))
     plt.show()
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
#*** Function AC() This function combines the 5 main function calls:
#***  AC_base, graf_act, graf_simult, AC_isup, AC_vsup.#-----------------------
-----------------------------------------------
def AC (lane, lane_sup, lane_var_sup, ax_h, ax_v):
     X, psi, psi_u, phi, phi_u = AC_base(lane)
     graf_act (X, psi, phi, ax_h, ax_v )
     graf_simult (X, psi, phi, ax_h, ax_v)
     AC_isup( X, psi,  phi, phi_u, lane_sup, ax_h, ax_v)
     AC_vsup(X, psi, psi_u, phi, lane_var_sup, ax_h, ax_v)
     return
#-------------------------------------------------------------------------
```

End of python code

The code is written in the least cryptic way possible, 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.