```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.preprocessing import StandardScaler
        from numpy import linalg as LA
        from fanalysis.mca import MCA
```

```
In [2]: D = pd.read_excel("Chiens.xlsx",sheet_name="Feuil1",index_col=0)
        print(D.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8 entries, Beauceron to Labrador
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Taille     8 non-null      object 
 1   Velocite   8 non-null      object 
 2   Affection  8 non-null      object 
 3   Cote       8 non-null      float64
 4   Fonction   8 non-null      object 
dtypes: float64(1), object(4)
memory usage: 384.0+ bytes
None
```

Extraction des variables actives

```
In [3]: DActives = D[['Taille','Velocite','Affection']]
        print(DActives)
        #nombre de variables
        p = DActives.shape[1]
        #nombre d'observations
        n = DActives.shape[0]
```

```
             Taille Velocite Affection
Chien                                 
Beauceron    Taille++  Veloc++    Affec+
Basset         Taill-    Velo-     Affe-
Berger All   Taille++  Veloc++    Affec+
Boxer         Taille+   Veloc+    Affec+
Bull-Dog       Taill-    Velo-    Affec+
Bull-Mastif  Taille++    Velo-     Affe-
Caniche        Taill-   Veloc+    Affec+
Labrador      Taille+   Veloc+    Affec+
```

```
In [4]: display(DActives)
```

Chien	Taille	Velocite	Affection
Beauceron	Taille++	Veloc++	Affec+
Basset	Taill-	Velo-	Affe-
Berger All	Taille++	Veloc++	Affec+
Boxer	Taille+	Veloc+	Affec+
Bull-Dog	Taill-	Velo-	Affec+
Bull-Mastif	Taille++	Velo-	Affe-
Caniche	Taill-	Veloc+	Affec+
Labrador	Taille+	Veloc+	Affec+

Tableau disjonctif complet (TDC)

```
In [5]: X = pd.get_dummies(DActives,prefix='',prefix_sep='')
        print(X)
```

```
             Taill-  Taille+  Taille++  Velo-  Veloc+  Veloc++  Affe-  Affec+
Chien                                                                         
Beauceron         0        0         1      0       0        1      0       1
Basset            1        0         0      1       0        0      1       0
Berger All        0        0         1      0       0        1      0       1
Boxer             0        1         0      0       1        0      0       1
Bull-Dog          1        0         0      1       0        0      0       1
Bull-Mastif       0        0         1      1       0        0      1       0
Caniche           1        0         0      0       1        0      0       1
Labrador          0        1         0      0       1        0      0       1
```

```
In [6]: #nombre total de modalités
        M = X.shape[1]
```

Détermination du profil profil-moyen

```
In [7]: somme_col = np.sum(X.values,axis=0)
        print('Le profil colonne:',somme_col/(n*p))
```

```
Le profil colonne: [0.125      0.08333333 0.125      0.125      0.125      0.08333333
 0.08333333 0.25      ]
```

Le profil-moyen (Diapo 17) est alors : c=(0.125 0.083 0.125 0.125 0.125 0.083 0.083 0.25)

Analyse ACM

```
In [8]: acm = MCA(row_labels=DActives.index,var_labels=DActives.columns)
        acm.fit(DActives.values)
        None
```

Nombre max de facteurs

Le nombre maximum de facteurs principaux est égal à : m-p (Diapo 31)

- m étant le nombre total de modalités
- p le nombre de varaibles.

```
In [9]: Fmax = M-p
        print('Le nombre maximum de facteurs principaux est égal à', Fmax)
```

```
Le nombre maximum de facteurs principaux est égal à 5
```

Valeurs propres

```
In [10]: print(pd.DataFrame(np.transpose(acm.eig_),columns=['Val.P','%','Cumul %'],index=range(1,Fmax+1)))
```

```
      Val.P          %     Cumul %
1  0.708031  42.481875   42.481875
2  0.591489  35.489362   77.971237
3  0.261992  15.719509   93.690746
4  0.069747   4.184791   97.875538
5  0.035408   2.124462  100.000000
```

La matrice précédente montre que la valeur de l'inertie cumulée est de 77.97% si on considère les deux premiers facteurs principaux

Nombres de facteurs à retenir en se basant sur la règle de Kaiser

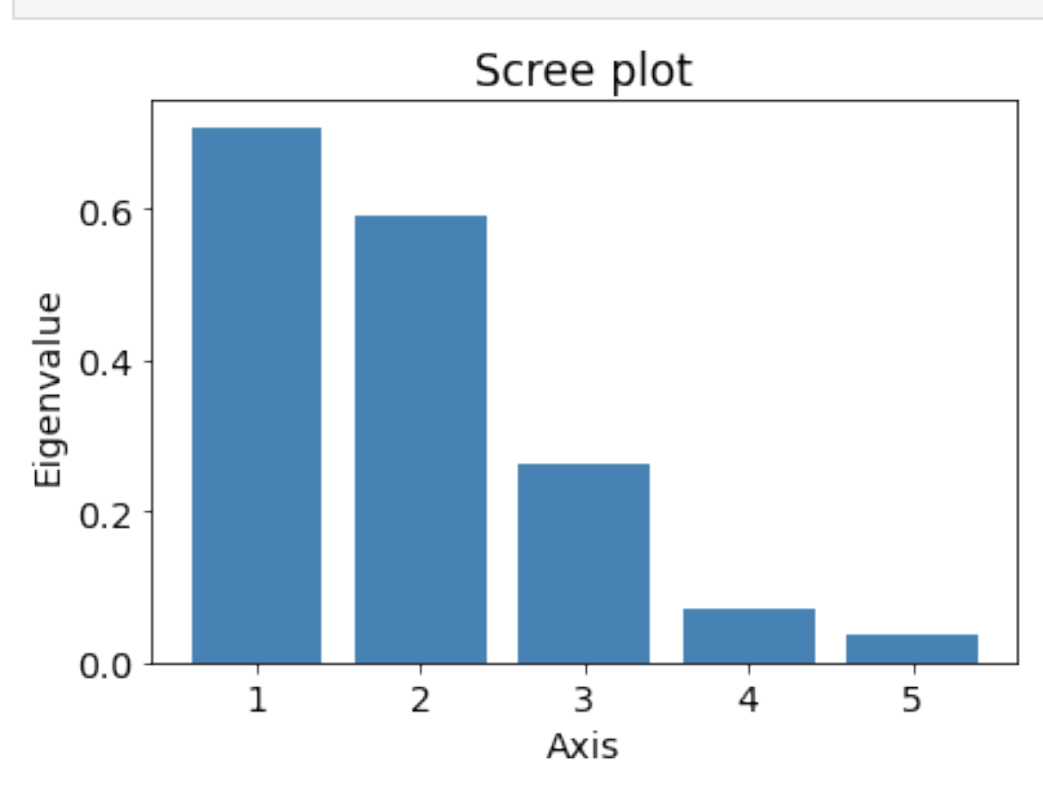
```
In [11]: Lambda_min = 1/p
         print('Lambda_min est égal à', Lambda_min)
```

```
Lambda_min est égal à 0.3333333333333333
```

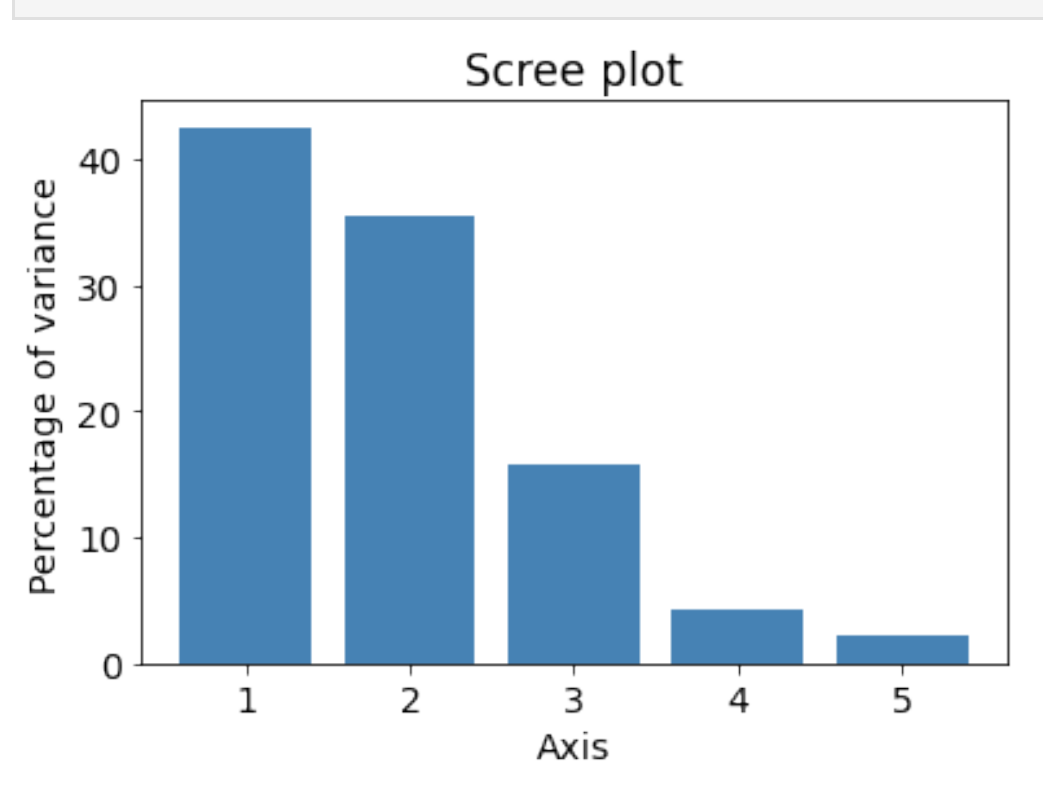
Selon la règle de Kaiser, on doit retenir les facteurs principaux dont les valeurs propres sont > à 1/p = 0.33 (Diapo 33). On retient donc les deux premiers facteurs principaux

Nombres de facteurs à retenir en se basant sur la règle du coude

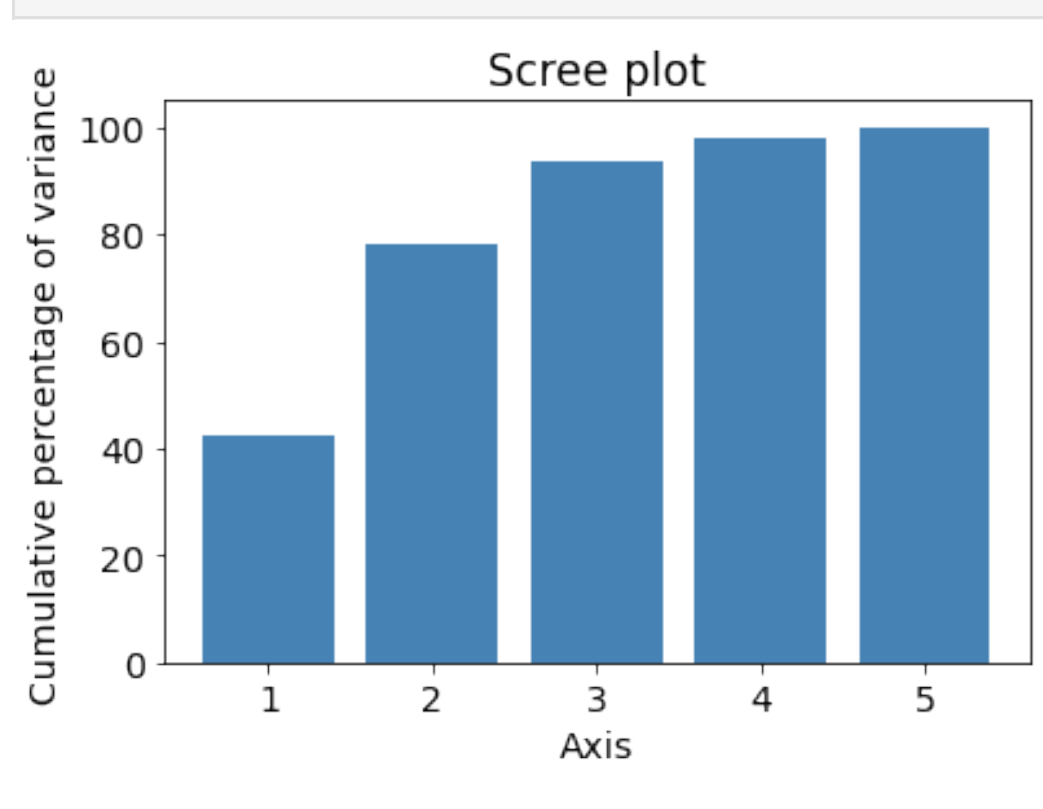
```
In [12]: plt.rcParams.update({'font.size': 14})
         acm.plot_eigenvalues()
```



```
In [13]: plt.rcParams.update({'font.size': 14})
         acm.plot_eigenvalues(type="percentage")
```

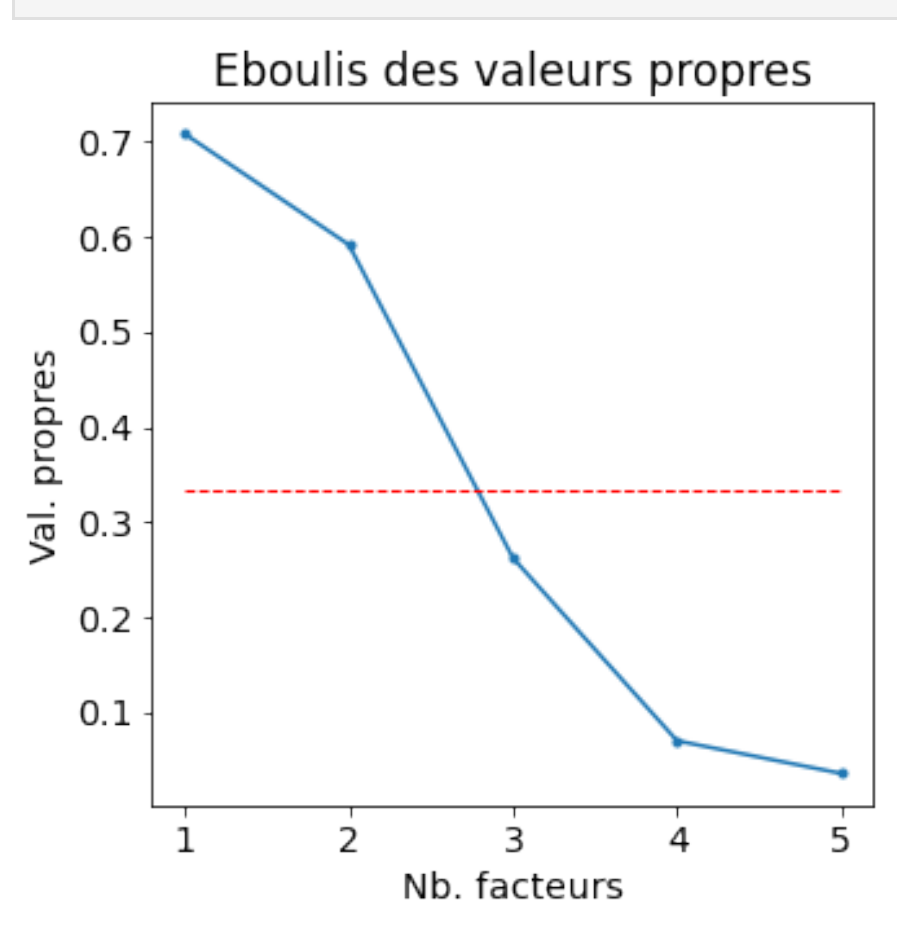


```
In [14]: plt.rcParams.update({'font.size': 14})
         acm.plot_eigenvalues(type="cumulative")
```



```
In [15]: fix,ax = plt.subplots(figsize=(5,5))
         ax.plot(range(1,Fmax+1),acm.eig_[0],".-")
         ax.set_xlabel("Nb. facteurs")
         ax.set_ylabel("Val. propres")
         plt.title("Eboulis des valeurs propres")

         #seuil - Règle de Kaiser
         ax.plot([1,Fmax],[1/p,1/p],"r--",linewidth=1)
         plt.show()
```



Selon la règle de coude, on peut retenir les 3 ou 4 premiers facteurs principaux

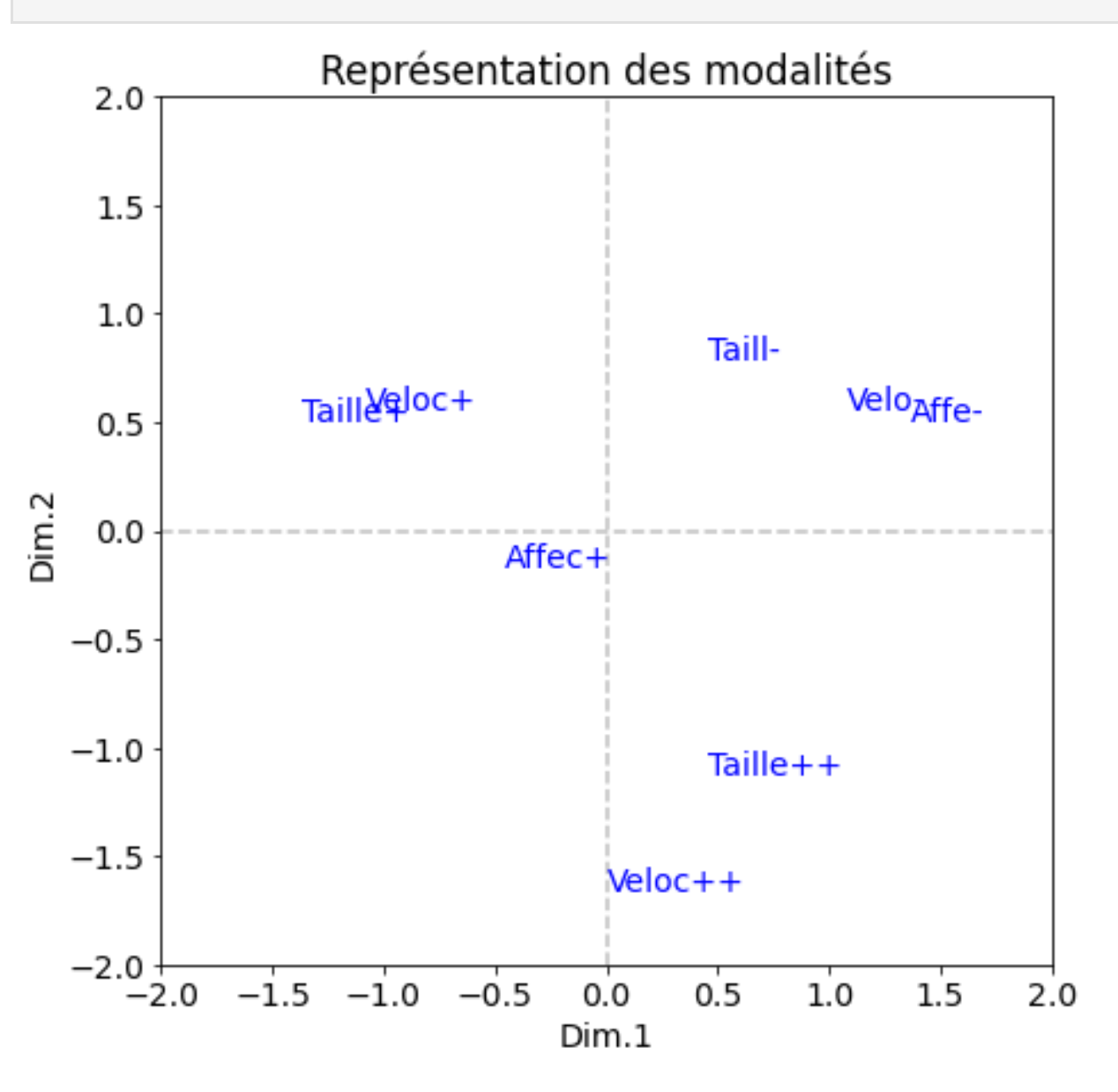
Réprésentation des modalités dans le plan factoriel

```
In [16]: print(pd.DataFrame(acm.col_coord_[:,:2],index=X.columns,columns=['Coord.F1','Coord.F2']))
```

```
              Coord.F1  Coord.F2
Taill-    4.558738e-01  0.788128
Taille+  -1.367621e+00  0.500758
Taille++  4.558738e-01 -1.121966
Velo-     1.081461e+00  0.554740
Veloc+   -1.081461e+00  0.554740
Veloc++   1.779876e-15 -1.664220
Affe-     1.367621e+00  0.500758
Affec+   -4.558738e-01 -0.166919
```

Le tableau suivant donne la projection des modalités sur les deux premiers facteurs principaux

```
In [17]: fix,ax = plt.subplots(figsize=(7,7))
         ax.axis([-2,+2,-2,+2])
         ax.plot([-2,+2],[0,0],color='silver',linestyle='--')
         ax.plot([0,0],[-2,+2],color='silver',linestyle='--')
         ax.set_xlabel("Dim.1")
         ax.set_ylabel("Dim.2")
         plt.title("Représentation des modalités")
         for i in range(X.shape[1]):
             ax.text(acm.col_coord_[i,0],acm.col_coord_[i,1],X.columns[i],color='blue')
         plt.show()
```



La représentation des modalités permet d'extraire les connaissances suivantes:

- On observe trois groupes de variables reliés: (Velo-, Affec-), (Taille++, Veloc++) et (Taille+, Veloc+).
- Affec+ est une variable un peu isolée.

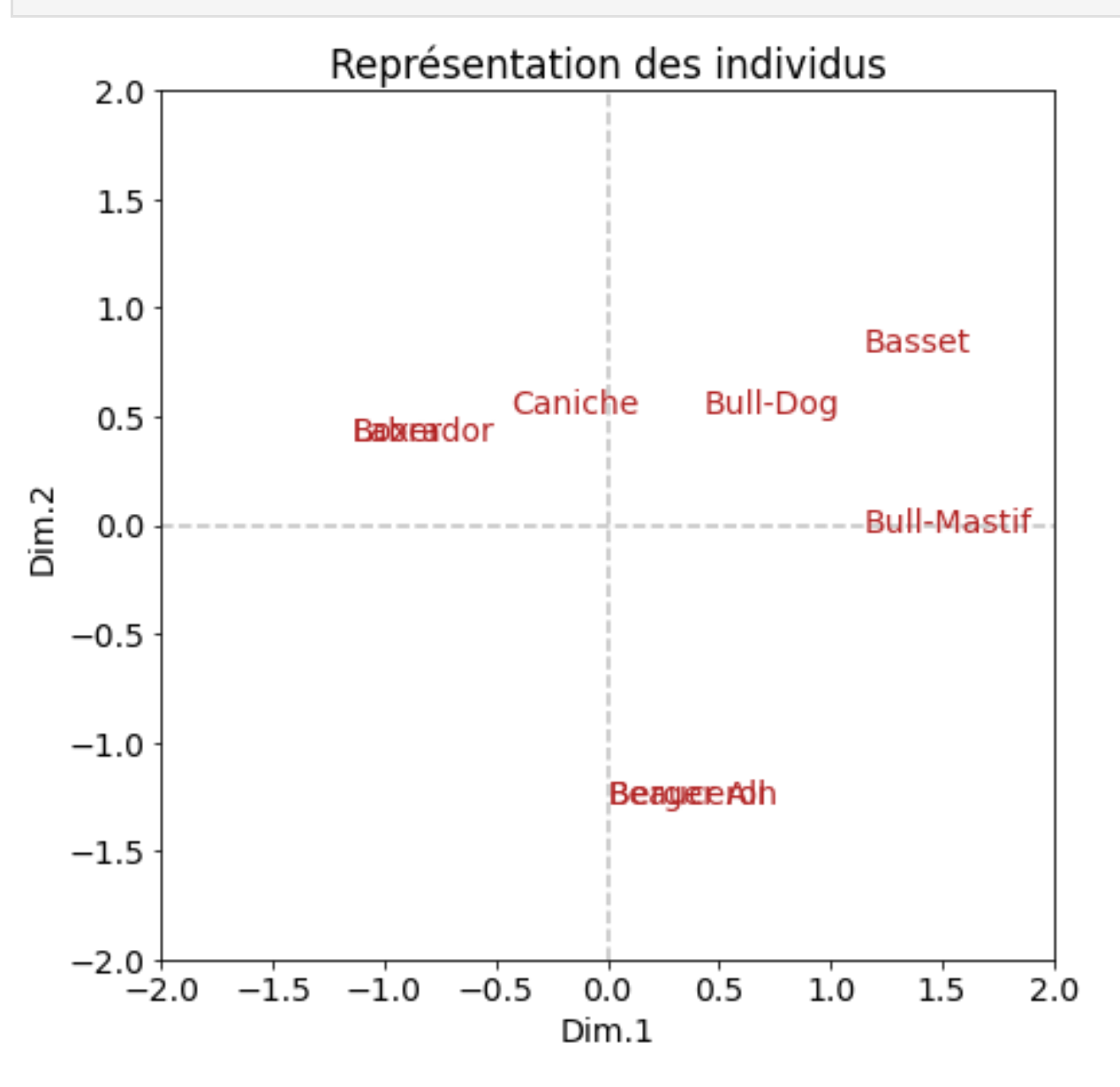
Réprésentation des individus dans le plan factoriel

```
In [18]: print(pd.DataFrame(acm.row_coord_[:,:2],index=X.columns,columns=['Coord.F1','Coord.F2']))
```

```
              Coord.F1  Coord.F2
Taill-    2.113837e-15 -1.279924
Taille+   1.150779e+00  0.799057
Taille++  2.047779e-15 -1.279924
Velo-    -1.150779e+00  0.385124
Veloc+    4.284137e-01  0.509675
Veloc++   1.150779e+00 -0.028809
Affe-    -4.284137e-01  0.509675
Affec+   -1.150779e+00  0.385124
```

```
In [19]: fix,ax = plt.subplots(figsize=(7,7))
         ax.axis([-2,+2,-2,+2])
         ax.plot([-2,+2],[0,0],color='silver',linestyle='--')
         ax.plot([0,0],[-2,+2],color='silver',linestyle='--')
         ax.set_xlabel("Dim.1")
         ax.set_ylabel("Dim.2")
         plt.title("Représentation des individus")
         for i in range(X.shape[0]):
             ax.text(acm.row_coord_[i,0],acm.row_coord_[i,1],X.index[i],color='firebrick')

         plt.show()
```



La représentation des individus permet d'extraire les connaissances suivantes:

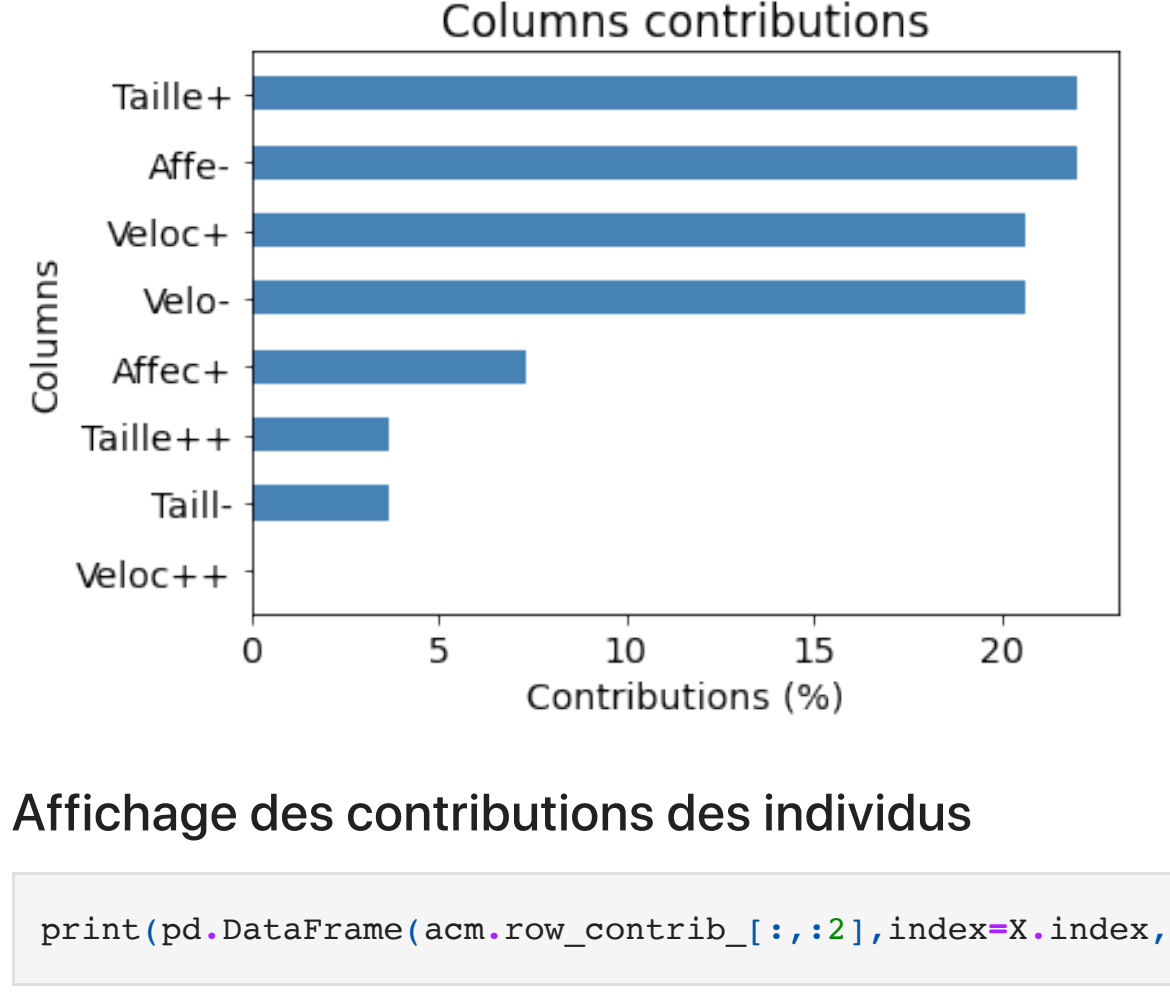
- On observe deux paires d'individus superposées : (Boxer, Labrador) et (Beauceron, Berger All).
- Les caractéristiques du Caniche est plus proche de Basset que le Beauceron.
- Le bull-Massif semble contibuer fortement au 1er axe factoriel.

Affichage des contributions des modalités

```
In [20]: print(pd.DataFrame(acm.col_contrib_[:,:2],index=X.columns,columns=['Contrib.F1','Contrib.F2']))
```

```
            Contrib.F1  Contrib.F2
Taill-    3.668993e+00   13.126725
Taille+   2.201396e+01    3.532865
Taille++  3.668993e+00   26.602515
Velo-     2.064806e+01    6.503426
Veloc+    2.064806e+01    6.503426
Veloc++   3.728600e-29   39.020556
Affe-     2.201396e+01    3.532865
Affec+    7.337985e+00    1.177622
```

```
In [21]: acm.plot_col_contrib(num_axis=1)
```

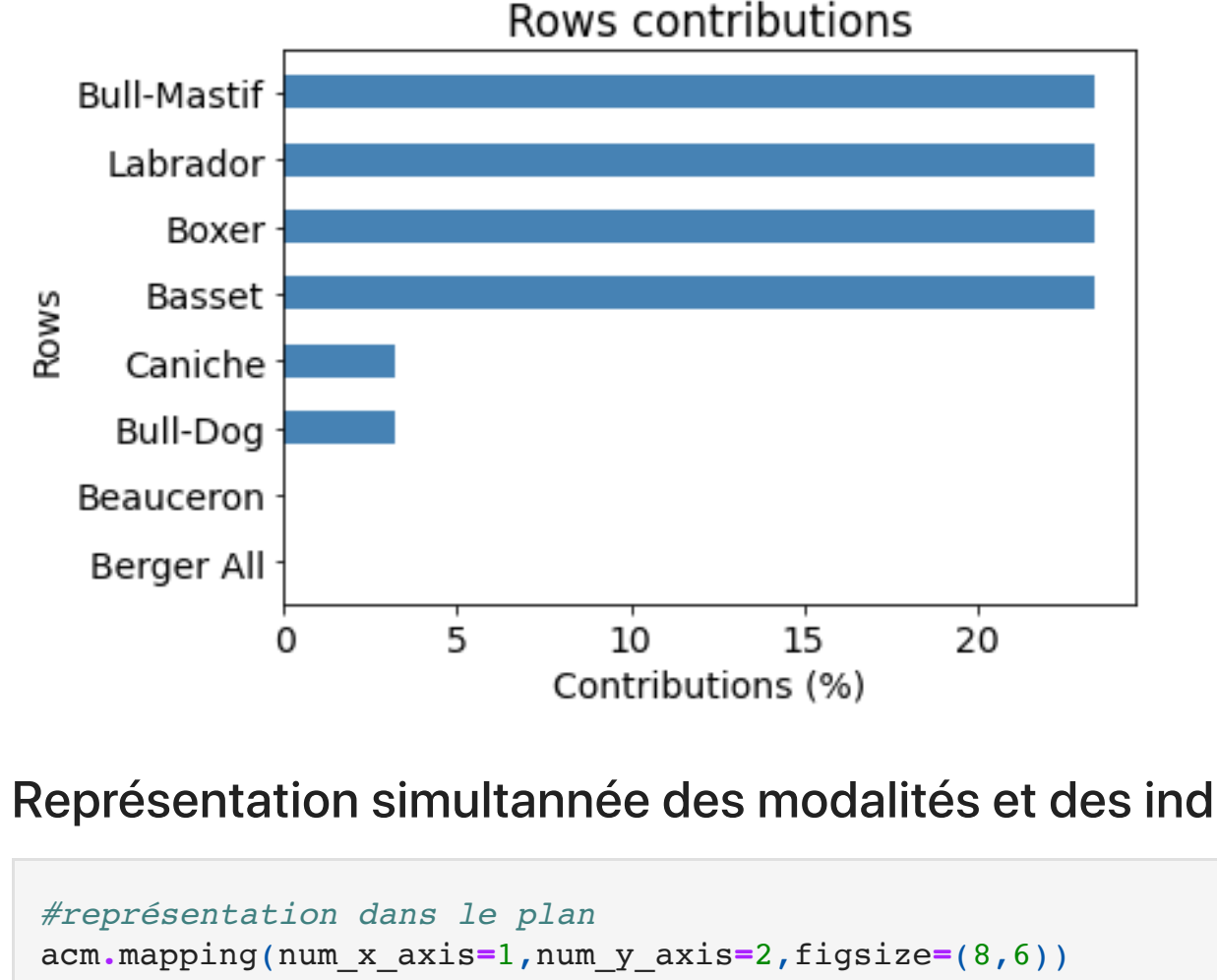


Affichage des contributions des individus

```
In [22]: print(pd.DataFrame(acm.row_contrib_[:,:2],index=X.index,columns=['Contrib.F1','Contrib.F2']))
```

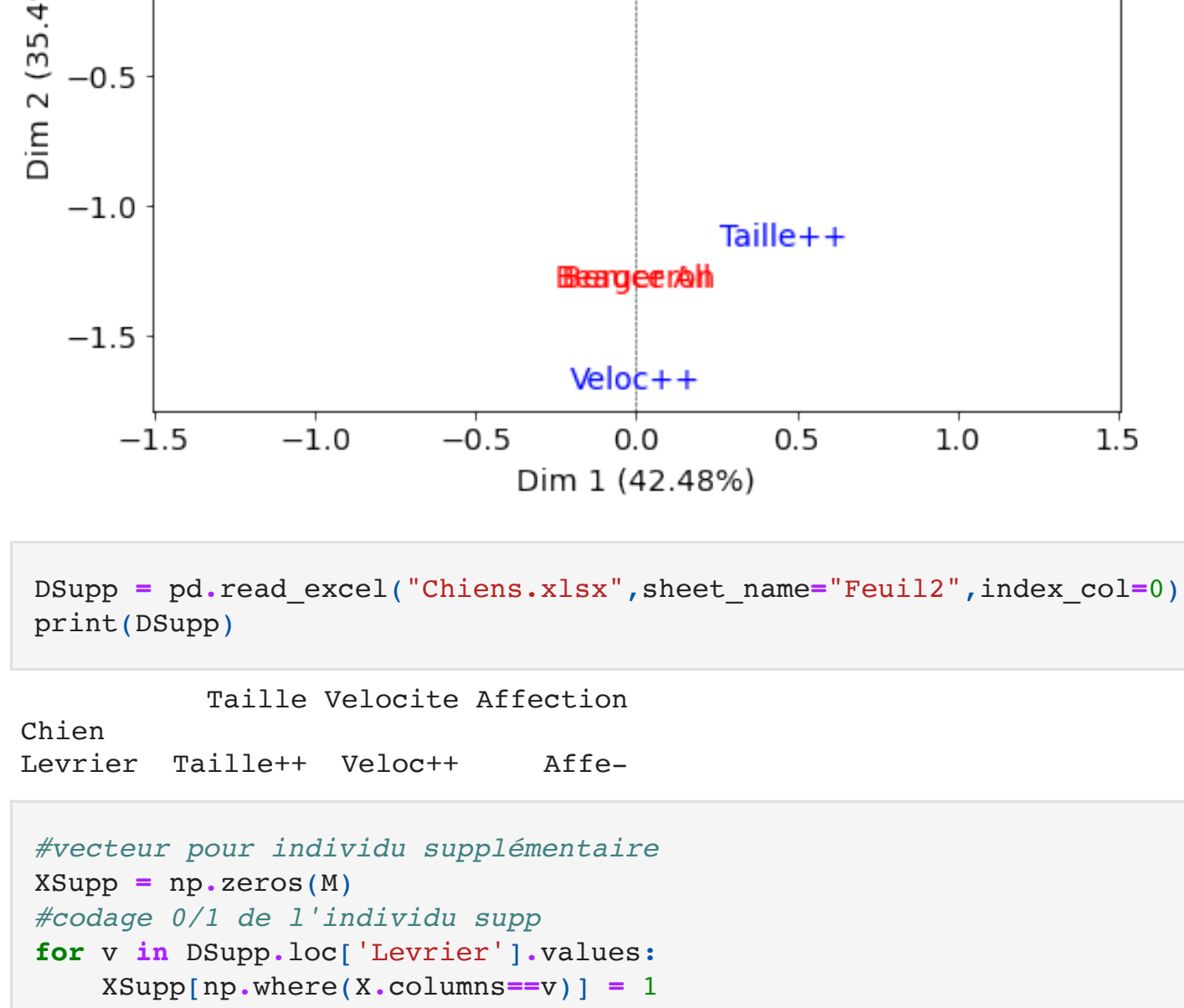
```
               Contrib.F1  Contrib.F2
Chien                                
Beauceron    7.888609e-29   34.620365
Basset       2.337985e+01   13.493329
Berger All   7.403275e-29   34.620365
Boxer        2.337985e+01    3.134479
Bull-Dog     3.240293e+00    5.489722
Bull-Mastif  2.337985e+01    0.017539
Caniche      3.240293e+00    5.489722
Labrador     2.337985e+01    3.134479
```

```
In [23]: acm.plot_row_contrib(num_axis=1)
```



Représentation simultannée des modalités et des individus

```
In [24]: #représentation dans le plan
         acm.mapping(num_x_axis=1,num_y_axis=2,figsize=(8,6))
```



```
In [25]: DSupp = pd.read_excel("Chiens.xlsx",sheet_name="Feuil2",index_col=0)
         print(DSupp)
```

```
           Taille Velocite Affection
Chien                               
Levrier  Taille++  Veloc++     Affe-
```

```
In [26]: #vecteur pour individu supplémentaire
         XSupp = np.zeros(M)
         #codage 0/1 de l'individu supp
         for v in DSupp.loc['Levrier'].values:
             XSupp[np.where(X.columns==v)] = 1
```

Profil-ligne de l'individu supplémentaire

```
In [27]: XSupp = XSupp / p
         print(XSupp)
```

```
[0.         0.         0.33333333 0.         0.         0.33333333
 0.33333333 0.        ]
```

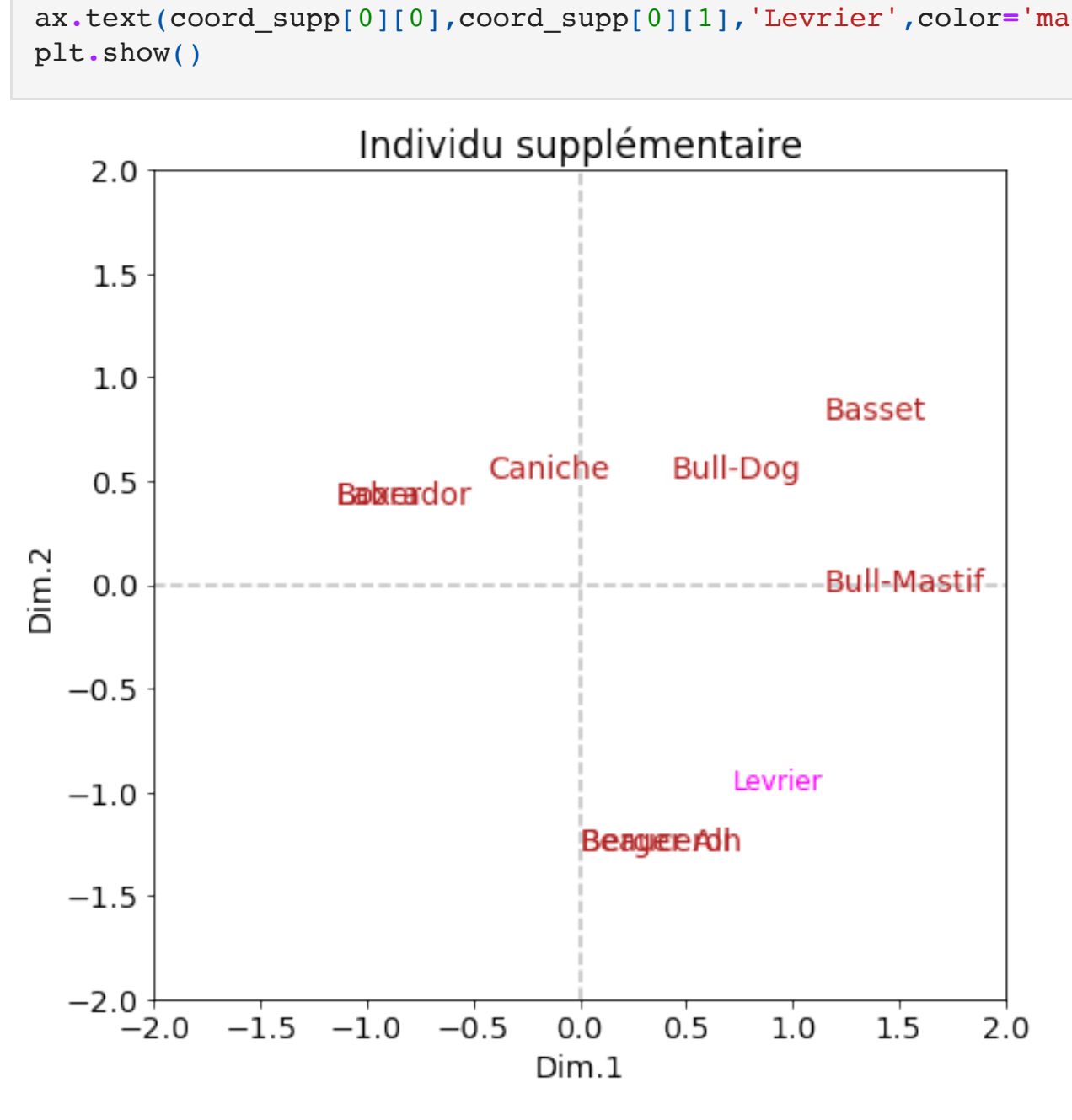
Projection de l'individu supplémentaire

```
In [28]: coord_supp = np.dot(np.reshape(XSupp,(1,M)),acm.col_coord_[:,:2])
         coord_supp = coord_supp / np.sqrt(acm.eig_[0][:2])
         print(coord_supp)
```

```
[[ 0.72236576 -0.99054221]]
```

Représentation dans le plan factoriel

```
In [29]: fix,ax = plt.subplots(figsize=(7,7))
         ax.axis([-2,+2,-2,+2])
         ax.plot([-2,+2],[0,0],color='silver',linestyle='--')
         ax.plot([0,0],[-2,+2],color='silver',linestyle='--')
         ax.set_xlabel("Dim.1")
         ax.set_ylabel("Dim.2")
         plt.title("Individu supplémentaire")
         #individus actifs
         for i in range(X.shape[0]):
             ax.text(acm.row_coord_[i,0],acm.row_coord_[i,1],X.index[i],color='firebrick')
         #individu supplémentaire
         ax.text(coord_supp[0][0],coord_supp[0][1],'Levrier',color='magenta',fontsize=12)
         plt.show()
```



La projection de l'individu supplémentaire permet d'extraire les connaissances suivantes:

- Le levier est situé dans le voisinage du Berger All et du Beauceron.
- Le retour aux données initiale montre que ces trois races partagent les caractéristiques suivantes : Taille ++, Velocite++.

```
In [ ]:
```

```
In [ ]:
```