ACD sous Python avec scientisttools

Duvérier DJIFACK ZEBAZE

Ce tutoriel a pour objectif de présenter rapidement les principales fonctionnalités offertes par le package « scientisttools » pour réaliser une Analyse des Correspondances Discriminante .

Présentation des données

L'analyse des correspondances discriminante (ACD) est le pendant de l'analyse factorielle discriminante pour les descripteurs catégoriels. On la reconnaît sous les traits de l'analyse discriminante barycentrique. Lorsque le nombre de classes est supérieur à 2, l'approche passe par un tableau de contingence particulier soumis à une analyse factorielle des correspondances (AFC).

Importation des données

Nous illustrons l'analyse des correspondances discriminante à l'aide d'un exemple sur les données « Races Canines » extraites de l'ouvrage de Tenenhaus. Il s'agit de prédire la variable « Fonction » (utilite, chasse, compagnie) de ($n = 27$) chiens à partir de leurs caractéristiques (Taille, Poids, etc. 6 variables).

```
# Chargement des données
import pandas as pd
# Données actives
DTrain = pd.read_csv("./donnee/races_canines.txt",sep="\t",encoding='latin-1',
                     index_col=0)
print(DTrain.info())
```

```
## <class 'pandas.core.frame.DataFrame'>
## Index: 27 entries, Beauceron to Terre-Neuve
## Data columns (total 7 columns):
##  #   Column        Non-Null Count  Dtype
## ---  ------        --------------  -----
##  0   Taille        27 non-null     object
##  1   Poids         27 non-null     object
##  2   Velocite      27 non-null     object
##  3   Intelligence  27 non-null     object
##  4   Affection     27 non-null     object
##  5   Agressivite   27 non-null     object
##  6   Fonction      27 non-null     object
## dtypes: object(7)
## memory usage: 1.7+ KB
## None
```

Distribution relative

Nous calculons la distribution relative des classes :

```
# Distribution relative des classes
d = (DTrain.Fonction.value_counts(normalize=True).reset_index()
                .rename(columns={"index":"Fonction","Fonction":"p(k)"}))
```

Table 1 – Distribution relative des classes

Fonction	p(k)
compagnie	0.3703704
chasse	0.3333333
utilite	0.2962963

Analyse bivariée

Une première piste consiste à procéder à une simple analyse bivariée. Nous croisons chaque descripteur avec la variable cible. Nous disposons ainsi d'une première indication sur les liaisons individuelles de chaque descripteur avec « Fonction ».

```
# V de Cramer
import scientistmetrics as st
cramerV = st.scientistmetrics(DTrain)
cramerV = (cramerV.iloc[:6,6].to_frame()
                  .sort_values(by="Fonction",ascending=False).T)
```

Table 2 – V de Cramer entre la cible et les descripteurs

	Affection	Poids	Taille	Agressivite	Velocite	Intelligence
Fonction	0.7393939	0.6722976	0.5503246	0.511811	0.3963569	0.2769003

Nous avons quelques relations qui sont assez fortes : Affection avec un V de Cramer de 0.74 ; Poids avec un V de Cramer de 0.67 ; Taille avec un V de Cramer de 0.55 et Agressivite avec un V de Cramer de 0.51. Il semble donc possible d'expliquer la fonction des chiens à partir de leurs caractéristiques. Mais il faut le faire de manière multivariée c'est - à - dire en tenant compte du rôle simultané de l'ensemble des descripteurs.

Analyse avec scientisttools

Modélisation avec scientisttools

Sage précaution avec les packages pour Python, nous affichons le numéro de la version de « scientisttools » utilisée dans ce tutoriel.

```
# version
import scientisttools
print(scientisttools.__version__)
```

```
## 0.0.9
```

Nous fonctionnons avec la version « 0.0.9 ».

```
# Importation
from scientisttools.discriminant_analysis import DISCA
```

On crée une instance de la classe DISCA, en lui passant ici des étiquettes pour les variables explicatives et la variable cible.

```
# Instanciation
disca = DISCA(n_components=None,
              target=["Fonction"],
              features_labels=DTrain.columns[:-1].values,
              matrix_type="completed",
              priors=None,
              parallelize=False)
```

On estime le modèle en appliquant la méthode `.fit` de la classe DISCA sur le jeu de données.

```
# Entraînement du modèle
disca.fit(DTrain)
```

```
## DISCA(features_labels=array(['Taille', 'Poids', 'Velocite', 'Intelligence', 'Affection',
##        'Agressivite'], dtype=object),
##       target=['Fonction'])
```

Inspection de l'objet DISCA

— `priors_` correspond à la distribution relative des classes.

```
# distribution des classes
print(lda.priors_)
```

Table 3 – Distribution relative pour chaque classe

compagnie	chasse	utilite
0.3703704	0.3333333	0.2962963

— `statistics_test_` correspond aux tests statistiques entre variables qualitatives

```
# Tests statistiques
stats_test = disca.statistics_test_
print(stats_test.keys())
```

```
## dict_keys(['chi2', 'log-likelihood-test', "cramer's V", "tschuprow's T", 'pearson'])
```

— `mod_stats` correspond à la distribution absolue et relative des classes

Table 4 – Test statistique de chi2

	statistic	df	pvalue
Taille	20.161309	4	0.0004641
Poids	25.634866	4	0.0000375
Velocite	11.775726	4	0.0190993
Intelligence	4.067405	4	0.3969605
Affection	18.860828	2	0.0000802
Agressivite	7.690039	2	0.0213860

```
# distribution absolue et relative des classes
print(disca.mod_stats)
```

Table 5 – Distribution absolue et relative pour chaque classe

	n(l)	p(l)
Taille_Taille+	5	0.0308642
Taille_Taille++	15	0.0925926
Taille_Taille-	7	0.0432099
Poids_Poids+	14	0.0864198
Poids_Poids++	5	0.0308642
Poids_Poids-	8	0.0493827
Velocite_Veloc+	8	0.0493827
Velocite_Veloc++	9	0.0555556
Velocite_Veloc-	10	0.0617284
Intelligence_Intell+	13	0.0802469
Intelligence_Intell++	6	0.0370370
Intelligence_Intell-	8	0.0493827
Affection_Affec+	14	0.0864198
Affection_Affec-	13	0.0802469
Agressivite_Agress+	13	0.0802469
Agressivite_Agress-	14	0.0864198

Analyse des classes

Coordonnées des classes

L'objet « disca » fournit les coordonnées des points - classes.

```
# Coordonnées des points - classes
disca.gcoord_
```

Table 6 – Coordonnées factorielles des classes

	Dim.1	Dim.2
chasse	0.1671133	0.4767969
compagnie	-0.7146513	-0.1626452
utilite	0.7053116	-0.3330900

On projette ces points - classes dans le plan :

```
# Projection des points classes
from plotnine import *
gcoord = disca.gcoord_
```

```
p = (ggplot(gcoord,aes(x="Dim.1",y="Dim.2",label=gcoord.index))+
        geom_point(aes(color=gcoord.index))+
        geom_text(aes(color=gcoord.index),
                  adjust_text={'arrowprops': {'arrowstyle': '-','lw':1.0}})+
        geom_hline(yintercept=0,colour="black",linetype="--")+
        geom_vline(xintercept=0,colour="black",linetype="--")+
        theme(legend_direction="vertical",legend_position=(0.2,0.7))+
        labs(color="Fonction"))
print(p)
```

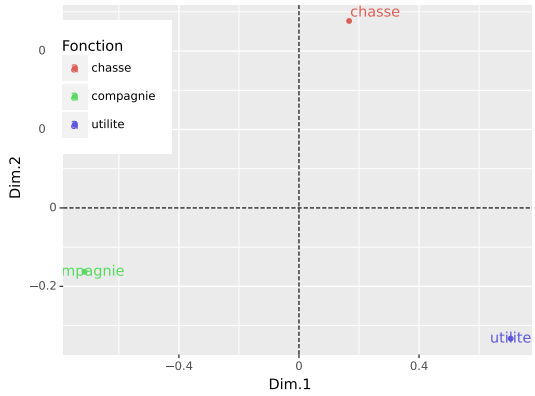

Figure 1 – Carte des points - classes

Visiblement, « compagnie » et « utilite » s'opposent sur le premier facteur. « chasse » se démarque des deux autres sur le second facteur.

Distances entre centres de classes

Les distances entre centres de classes permettent de situer les proximités entre les groupes sur l'ensemble des facteurs. La distance euclidienne entre les classes dans le répère factoriel est la suivante :

```
# Distance euclidienne
DE = pd.DataFrame(disca.ca_model_.row_dist_,columns=disca.classes_,
                  index=disca.classes_)
```

Table 7 – Distance euclidienne entre les classes

	chasse	compagnie	utilite
chasse	0.0000000	1.186395	0.9455743
compagnie	1.1863951	0.000000	2.0453462
utilite	0.9455743	2.045346	0.0000000

Les trois types de fonctions forment un triangle approximativement isocèle dans le plan factoriel.

Ajoutons ces distances sur le plan factoriel :

```
# Projection des points classes avec distances entre classes
p = (ggplot(gcoord,aes(x="Dim.1",y="Dim.2",label=gcoord.index))+
        geom_point(aes(color=gcoord.index))+
        geom_text(aes(color=gcoord.index),
                  adjust_text={'arrowprops': {'arrowstyle': '-','lw':1.0}})+
        geom_hline(yintercept=0,colour="black",linetype="--")+
        geom_vline(xintercept=0,colour="black",linetype="--")+
        theme(legend_direction="vertical",legend_position=(0.2,0.7))+
        annotate("segment",x=gcoord.iloc[0,0],y=gcoord.iloc[0,1],
                           xend=gcoord.iloc[1,0],yend=gcoord.iloc[1,1],
                           color="blue")+
        annotate("segment",x=gcoord.iloc[0,0],y=gcoord.iloc[0,1],
                           xend=gcoord.iloc[2,0],yend=gcoord.iloc[2,1],
                           color="blue")+
        annotate("segment",x=gcoord.iloc[1,0],y=gcoord.iloc[1,1],
                           xend=gcoord.iloc[2,0],yend=gcoord.iloc[2,1],
                           color="blue")+
        # Add test
        annotate('text', x = -0.3, y = 0.2,label = DE.iloc[0,1].round(2),
                 size = 10, angle='35')+
        annotate('text', x = 0.4, y = 0.2,label = DE.iloc[0,2].round(2),
                 size = 10, angle='-60')+
        annotate('text', x = 0, y = -0.25,label = DE.iloc[2,1].round(2),
                 size = 10, angle='-10')+
        labs(color="Fonction"))
print(p)
```

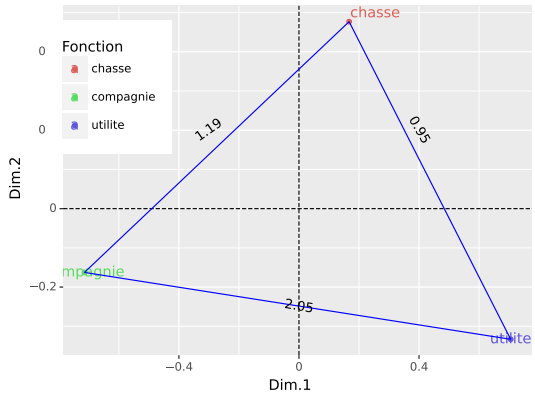

Figure 2 – Carte des points - classes

Qualité de la représentation des classes

Il suffit de passer les coordonnées au carré et de diviser par la somme en ligne. Sous scientisttools, elles correspondent à la qualité de représentation des points - lignes de l'analyse factorielle des correspondances.

```
# Qualité de représentation
gcos2 = pd.DataFrame(disca.ca_model_.row_cos2_,
                     index=disca.ca_model_.row_labels_,
                     columns = disca.ca_model_.dim_index_)
```

Table 8 – Qualité de représentation des classes - COS2

	Dim.1	Dim.2
chasse	0.1094046	0.8905954
compagnie	0.9507549	0.0492451
utilite	0.8176421	0.1823579

Le graphique (Figure 1) ne laissait aucun doute, mais c'est toujours mieux quand les chiffres confirment : les informations portées par « compagnie » et « utilite » sont bien captées par le premier facteur. « chasse » est mieux situé sur le second facteur. Et la somme en ligne dans le tableau des COS2 fait bien 100%.

Contributions des classes

Sous scientisttools, elles correspondent aux contributions des points - lignes de l'analyse factorielle des correspondances.

```
# Contribution des groupes
gcontrib = pd.DataFrame(disca.ca_model_.row_contrib_,
                        index=disca.ca_model_.row_labels_,
                         columns = disca.ca_model_.dim_index_)
```

Table 9 – Contributions des classes

	Dim.1	Dim.2
chasse	2.691509	63.975158
compagnie	54.691459	8.271504
utilite	42.617032	27.753338

Le premier axe oppose les fonctions « compagnie » et « utilite ». Elles déterminent (**contributions** = 54.69% + 42.62%) 97.31% de l'information portée par le facteur. Elles sont aussi très bien représentées puisque 95.08% (resp. 81.76%) de l'information véhiculée par « compagnie » (resp. « utilite ») est restrancrite sur cet axe.

Le second axe permet surtout de distinguer la fonction « chasse » des deux premiers.

Structures canoniques

Les structures canoniques correspondent aux représentations des modalités colonnes du tableau de contingence - et donc des modalités des variables prédictives - dans le répère factoriel.

Poids, distance à l'origine et inertie

```
# Informations sur les modalités
from scientisttools.extractfactor import get_ca_col
mod_infos = get_ca_col(disca.ca_model_)["infos"]
```

Table 10 – Caractéristiques des modalités

	d(k,G)	p(k)	I(k,G)
Taille_Taille+	0.4520000	0.0308642	0.0139506
Taille_Taille++	0.4520000	0.0925926	0.0418519
Taille_Taille-	1.0448980	0.0432099	0.0451499
Poids_Poids+	0.2585459	0.0864198	0.0223435
Poids_Poids++	2.3750000	0.0308642	0.0733025
Poids_Poids-	1.1140625	0.0493827	0.0550154
Velocite_Veloc+	0.4250000	0.0493827	0.0209877
Velocite_Veloc++	0.2925926	0.0555556	0.0162551
Velocite_Veloc-	0.2450000	0.0617284	0.0151235
Intelligence_Intell+	0.1223373	0.0802469	0.0098172
Intelligence_Intell++	0.2520833	0.0370370	0.0093364
Intelligence_Intell-	0.1296875	0.0493827	0.0064043
Affection_Affec+	0.5076531	0.0864198	0.0438713
Affection_Affec-	0.5887574	0.0802469	0.0472460
Agressivite_Agress+	0.2821006	0.0802469	0.0226377
Agressivite_Agress-	0.2432398	0.0864198	0.0210207

Coordonnées des points modalités

Les coordonnées des points modalités sont fournies par l'objet `ca_model_`.

```
# Coordonnées des points modalités
mod_coord = pd.DataFrame(disca.ca_model_.col_coord_,
                         index=disca.ca_model_.col_labels_,
                         columns = disca.ca_model_.dim_index_)
```

Table 11 – Coordonnées des points modalités

	Dim.1	Dim.2
Taille_Taille+	-0.6154469	0.2706015
Taille_Taille++	0.6722783	0.0064729
Taille_Taille-	-1.0009914	-0.2071572
Poids_Poids+	0.1589724	0.4829841
Poids_Poids++	1.1993018	-0.9678198
Poids_Poids-	-1.0277654	-0.2403349
Velocite_Veloc+	-0.4655129	0.4563965
Velocite_Veloc++	0.5242948	0.1330699
Velocite_Veloc-	-0.0994550	-0.4848801
Intelligence_Intell+	-0.3119926	-0.1581073
Intelligence_Intell++	0.4918395	-0.1008824
Intelligence_Intell-	0.1381083	0.3325862
Affection_Affec+	-0.6560651	-0.2779059
Affection_Affec-	0.7065316	0.2992833
Agressivite_Agress+	0.4309258	-0.3104892
Agressivite_Agress-	-0.4001454	0.2883114

```
# Ajout de la variable
modcoord = mod_coord.copy()
modcoord.loc[:,"variable"] = [x.split("_")[0] for x in mod_coord.index]
```

```
# Projection des points modalités
p = (ggplot(modcoord,aes(x="Dim.1",y="Dim.2",label=mod_coord.index))+
        geom_point(aes(color=modcoord.variable))+
        geom_text(aes(color=modcoord.variable),
                  adjust_text={'arrowprops': {'arrowstyle': '-','lw':1.0}})+
        geom_hline(yintercept=0,colour="black",linetype="--")+
        geom_vline(xintercept=0,colour="black",linetype="--")+
        theme(legend_direction="vertical",legend_position=(0.2,0.2))+
        labs(color="Variable"))
print(p)
```

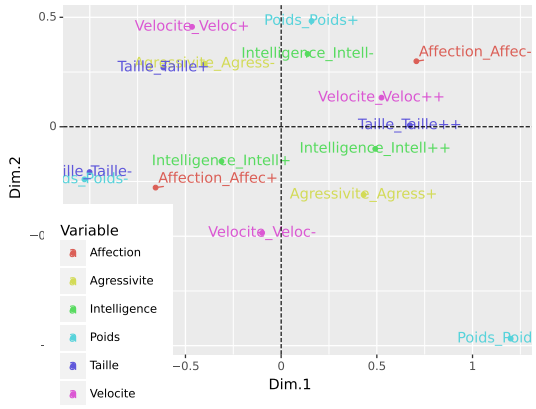

Figure 3 – Carte des points - modalités

Contributions des points modalités aux facteurs

Les contributions des points modalités sont :

```
# Contributions des points modalités
mod_contrib = pd.DataFrame(disca.ca_model_.col_contrib_,
                           index=disca.ca_model_.col_labels_,
                           columns = disca.ca_model_.dim_index_)
```

Affectation des classes

Fonction discriminante canonique

L'exécution de la méthode `disca.fit(DTrain)` provoque le calcul de plusieurs attributs parmi lesquels `disca.coef_`. Ce champ nous intéresse particulièrement car il correspond aux coefficients des fonctions de classement. Ces fonctions canoniques permettent de projeter des individus non étiquetés dans l'espace factoriel.

```
# Coefficients des fonctions discriminantes canoniques
print(disca.coef_)
```

Table 12 – Contribution des points modalités

	Dim.1	Dim.2
Taille_Taille+	3.3801114	1.9080116
Taille_Taille++	12.0995527	0.0032752
Taille_Taille-	12.5181065	1.5654847
Poids_Poids+	0.6314682	17.0194100
Poids_Poids++	12.8353248	24.4067368
Poids_Poids-	15.0819613	2.4080990
Velocite_Veloc+	3.0940922	8.6841100
Velocite_Veloc++	4.4154343	0.8305250
Velocite_Veloc-	0.1765359	12.2523506
Intelligence_Intell+	2.2584563	1.6935499
Intelligence_Intell++	2.5904659	0.3182242
Intelligence_Intell-	0.2723387	4.6115743
Affection_Affec+	10.7547848	5.6347465
Affection_Affec-	11.5820759	6.0681885
Agressivite_Agress+	4.3085212	6.5311108
Agressivite_Agress-	4.0007697	6.0646029

Table 13 – Coefficients des fonctions discriminantes canoniques

	Dim.1	Dim.2
Taille_Taille+	-0.1744162	0.1310424
Taille_Taille++	0.1905221	0.0031346
Taille_Taille-	-0.2836786	-0.1003186
Poids_Poids+	0.0450524	0.2338915
Poids_Poids++	0.3398793	-0.4686797
Poids_Poids-	-0.2912663	-0.1163854
Velocite_Veloc+	-0.1319252	0.2210161
Velocite_Veloc++	0.1485839	0.0644409
Velocite_Veloc-	-0.0281853	-0.2348097
Intelligence_Intell+	-0.0884179	-0.0765656
Intelligence_Intell++	0.1393861	-0.0488537
Intelligence_Intell-	0.0391396	0.1610593
Affection_Affec+	-0.1859273	-0.1345797
Affection_Affec-	0.2002294	0.1449319
Agressivite_Agress+	0.1221233	-0.1503585
Agressivite_Agress-	-0.1134003	0.1396186

Coordonnées des individus

A partir des fonctions discriminantes canoniques, on détermine les coordonnées des individus.

```
# Coordonnées factorielles des individus
row_coord = disca.row_coord_
```

```
# Ajout de la colonne Fonction
rowcoord = pd.concat([row_coord,DTrain["Fonction"]],axis=1)
# Projection des points modalités
p = (ggplot(rowcoord,aes(x="Dim.1",y="Dim.2",label=rowcoord.index))+
        geom_point(aes(color=rowcoord.Fonction))+
        geom_text(aes(color=rowcoord.Fonction),
                  adjust_text={'arrowprops': {'arrowstyle': '-','lw':1.0}})+
        geom_hline(yintercept=0,colour="black",linetype="--")+
        geom_vline(xintercept=0,colour="black",linetype="--")+
        theme(legend_direction="vertical",legend_position=(0.5,0.2))+
        labs(color="Fonction")+
        annotate("text",x=gcoord["Dim.1"].values,y=gcoord["Dim.2"].values,
```

Table 14 – Coordonnées des individus

	Dim.1	Dim.2
Beauceron	0.2319365	-0.0600368
Basset	-0.2416379	-0.2958810
Berger All	0.4597406	-0.0323249
Boxer	-0.4135109	0.2244462
Bull-Dog	-0.9908756	-0.5230403
Bull-Mastif	0.9639549	-0.7546350
Caniche	-0.8668115	-0.0395026
Chihuahua	-0.8633181	-0.2854154
Cocker	-0.6460896	-0.5816565
Colley	-0.0035871	0.2299404
Dalmatien	-0.6490345	0.5144234
Doberman	0.8458972	0.2471867
Dogue All	1.0404775	-0.2454715
Epag. Breton	-0.4212304	0.5421353
Epag. Français	0.1020604	0.6660272
Fox-Hound	0.7456506	0.4570997
Fox-Terrier	-0.8590920	-0.3571917
Gd Bleu Gasc	0.4651415	0.6136749
Labrador	-0.6490345	0.5144234
Levrier	0.5101270	0.7470769
Mastiff	0.8637083	-0.5447220
Pekinois	-0.8633181	-0.2854154
Pointer	0.6103736	0.5371639
St-Bernard	0.7361508	-0.7823469
Setter	0.3825695	0.5094520
Teckel	-0.9908756	-0.5230403
Terre-Neuve	0.5006272	-0.4923697

```
                label=gcoord.index,color=["red","green","violet"]))
print(p)
```

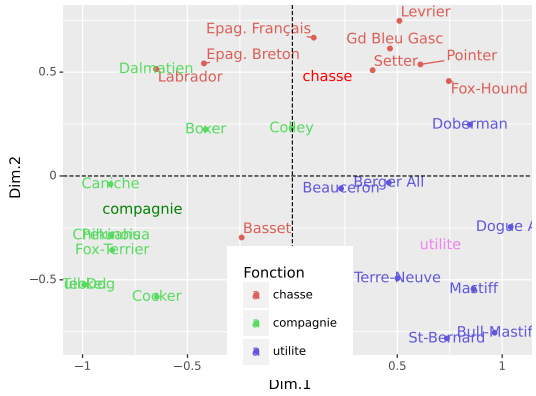

Figure 4 – Carte des individus

Valeurs propres associées aux facteurs

Les valeurs propres associées aux facteurs sont celles issues de l'analyse factorielle des correspondances.

```
# Valeurs propres
from scientisttools.extractfactor import get_eigenvalue
eig = get_eigenvalue(disca.ca_model_)
```

Table 15 – Valeurs propres associées aux facteurs

	eigenvalue	difference	proportion	cumulative
Dim.1	0.3458638	0.227414	74.48927	74.48927
Dim.2	0.1184498	NaN	25.51073	100.00000

La valeur propre (λ) indique l'inertie (la variance) expliquée par l'appartenance aux groupes sur chaque axe. En les additionnant, nous avons l'inertie expliquée par l'appartenance aux groupes dans l'espace complet soit 0.4643136. Cette inertie indique la quantité d'information que l'on peut modéliser dans la relation entre la cible Fonction et les descripteurs. Le premier facteur explique 74.49% de l'inertie totale.

On peut représenter graphiquement ces valeurs propres

```
# Scree plot
from scientisttools.ggplot import fviz_screeplot
p = fviz_screeplot(disca.ca_model_,choice="proportion",add_labels=True)
print(p)
```

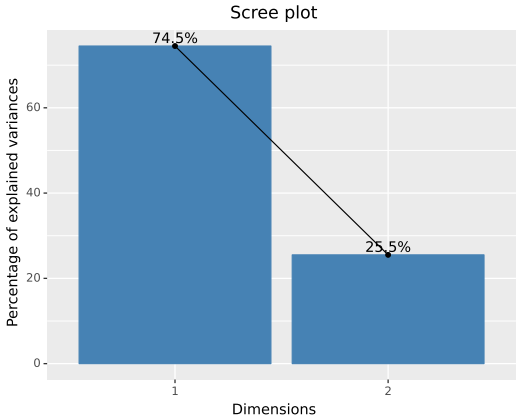

Figure 5 – Scree plot

Rapport de corrélation

Le champ `correlation_ratio_` correspond aux carrés des rapports de corrélation.

```
# Rapport de corrélation
print(disca.correlation_ratio_)
```

Table 16 – Rapport de correlation

	Dim.1	Dim.2
correl. ratio	0.7351037	0.5180404

Le rapport de corrélation est le ratio entre la variance expliquée par l'appartenance aux groupes et la vaiance totale de l'axe. Il indique la qualité de discrimination des classes sur le facteur. Nous avons $\eta_1^2 = 0.7351037$, c'est - à - dire 73.51% de la variabilité des observations est expliquée par l'appartenance aux groupes sur le premier facteur. L'indicateur varie entre 0 (discrimination nulle, les sous - populations sont complètement mélangées) et 1 (discrimination parfaite, elles sont agglutinées) sur les centres de classes qui sont distincts les uns des autres.

Corrélation canonique

La corrélation canonique est la racine carré du rapport de corrélation.

Table 17 – Corrélation canonique

	Dim.1	Dim.2
correl. ratio	0.8573819	0.7197503

Traitement d'individus supplémentaires

Les fonctions discriminantes canoniques nous permettent de positionner les individus suppémentaires dans le répère factoriel.

Importation des données

Nous chargeons les individus supplémentaires.

```
# Individus supplémentaires
Dsup = pd.read_excel("./donnee/races_canines_acm.xls",
                     header=0,sheet_name=1,index_col=0)
```

Table 18 – Individus supplémentaires

	Taille	Poids	Velocite	Intelligence	Affection	Agressivite
Medor	Taille+	Poids-	Veloc-	Intell++	Affec-	Agress+
Djeck	Taille++	Poids++	Veloc+	Intell+	Affec+	Agress-
Taico	Taille-	Poids+	Veloc++	Intell++	Affec+	Agress+
Rocky	Taille+	Poids+	Veloc+	Intell-	Affec+	Agress-
Boudog	Taille-	Poids-	Veloc++	Intell+	Affec-	Agress+
Wisky	Taille+	Poids++	Veloc-	Intell-	Affec+	Agress+

Coordonnées des individus supplémentaires

L'objet « DISCA » contient la fonction `transform()` bien connue des utilisateurs de scikit-learn. Elle permet d'obtenir les coordonnées des individus dans l'espace factoriel.

```
# Coordonnées des individus supplémentaires
row_sup_coord = disca.transform(Dsup)
```

Table 19 – Coordonnées des individus supplémentaires

	Dim.1	Dim.2
Medor	-0.0321289	-0.2744329
Djeck	0.0107306	-0.3160556
Taico	-0.0144601	-0.1357781
Rocky	-0.5214770	0.7520483
Boudog	-0.1924262	-0.2342553
Wisky	0.1126134	-0.6963258

On rajoute ces individus au plan factoriel

```
# Projection des points modalités
p = (ggplot(rowcoord,aes(x="Dim.1",y="Dim.2",label=rowcoord.index))+
        geom_point(aes(color=rowcoord.Fonction))+
        geom_text(aes(color=rowcoord.Fonction),
                  adjust_text={'arrowprops': {'arrowstyle': '-','lw':1.0}})+
        geom_hline(yintercept=0,colour="black",linetype="--")+
        geom_vline(xintercept=0,colour="black",linetype="--")+
        theme(legend_direction="vertical",legend_position=(0.5,0.2))+
        labs(color="Variable")+
        annotate("text",x=row_sup_coord["Dim.1"].values,
                        y=row_sup_coord["Dim.2"].values,
                    label=row_sup_coord.index))
print(p)
```

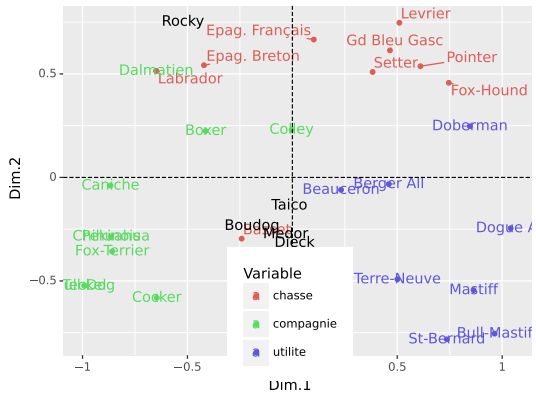

Figure 6 – Carte des individus

Distances euclidiennes aux classes

La fonction `decision_function()` permet de calculer les distances euclidiennes aux centres de classes.

```
# Distances euclidiennes aux classes
disca.decision_function(Dsup)
```

```
##           chasse  compagnie   utilite
## Medor   0.604044   0.478333  0.547259
## Djeck   0.653071   0.549714  0.482733
## Taico   0.408217   0.490990  0.557003
## Rocky   0.549920   0.873981  2.682536
## Boudog  0.634864   0.277847  0.815701
## Wisky   1.379187   0.969182  0.483231
```

Probabilités d'affectation

L'objet « scientisttools » calcule les probabilités d'affectation aux classes avec `predict_proba()`.

```
# probabilité d'affectation
print(disca.predict_proba(Dsup))
```

```
##           chasse  compagnie   utilite
## Medor   0.322822   0.381961  0.295217
## Djeck   0.318679   0.372868  0.308453
## Taico   0.345874   0.368724  0.285402
## Rocky   0.444261   0.419785  0.135955
## Boudog  0.318441   0.422972  0.258588
## Wisky   0.266300   0.363212  0.370487
```

Prédiction

On effectue la prédiction à partir de la matrice des explicatives des individus supplémentaires.

```
# Prediction des individus supplémentaires
ypred = disca.predict(Dsup)
ypred
```

```
##           predict
## Chien
## Medor   compagnie
## Djeck   compagnie
## Taico   compagnie
## Rocky      chasse
## Boudog  compagnie
## Wisky     utilite
```