Mamadou NIANG
Dioncounda DIARRA

Projet final d'Analyse de Données

1. Régression binaire

1- Déterminer le meilleur modèle possible pour modéliser P(Y |X1, X2).

On importe les données et on effectue les différentes étapes pour nettoyer la base avant de procéder à la régression binaire : Suppression des valeurs manquantes et transformation de Y en type facteur.

```
simu=read.table("C:/Users/dionc/Desktop/projetf_analyse_donees/donnees/simu.txt",header=TRUE)
library(tidyr)
simu=drop_na(simu,X1,X2,Y)
simu$Y<-as.factor(simu$Y)
```

On commence par estimer le modèle additif complet :

```
mod1 <- glm(Y ~ X1 + X2, data=simu, family = binomial(logit))
mod1
```

```
Call:  glm(formula = Y ~ X1 + X2, family = binomial(logit), data = simu)

Coefficients:
(Intercept)           X1           X2
    -0.3821       0.0949      -0.1368

Degrees of Freedom: 1999 Total (i.e. Null);  1997 Residual
Null Deviance:      2713
Residual Deviance: 2603         AIC: 2609
```

On essaye de trouver le meilleur modèle possible par le critère BIC :

```
library(MASS)
mod1.bic<- stepAIC(mod1,direction="backward",k=log(dim(simu)[1]))
mod1.bic
```

```
Call:  glm(formula = Y ~ X1 + X2, family = binomial(logit), data = simu)

Coefficients:
(Intercept)           X1           X2
    -0.3821       0.0949      -0.1368

Degrees of Freedom: 1999 Total (i.e. Null);  1997 Residual
Null Deviance:      2713
Residual Deviance: 2603         AIC: 2609
```

La sélection du meilleur modèle par le critère BIC nous retourne le modèle additif complet.

On essaye ensuite de trouver le meilleur modèle possible par le critère AIC :

```
mod1.aic<-  stepAIC(mod1, direction="backward",k=2)
mod1.aic
```

```
Call:  glm(formula = Y ~ X1 + X2, family = binomial(logit), data = simu)

Coefficients:
(Intercept)           X1           X2
    -0.3821       0.0949      -0.1368

Degrees of Freedom: 1999 Total (i.e. Null);  1997 Residual
Null Deviance:      2713
Residual Deviance: 2603         AIC: 2609
```

La sélection du meilleur modèle par le critère AIC nous retourne aussi le modèle additif complet.

Afin d'améliorer le modèle, on peut tenir en compte la présence d'interaction entre X1 et X2 :

```
Call:  glm(formula = Y ~ X1 * X2, family = binomial(logit), data = simu)

Coefficients:
(Intercept)           X1           X2        X1:X2
   -0.37899      0.09295     -0.13598     -0.01011

Degrees of Freedom: 1999 Total (i.e. Null);  1996 Residual
Null Deviance:      2713
Residual Deviance: 2600         AIC: 2608
```

L'AIC de ce modèle est très légèrement meilleur.

Regardons si on peut essayer de l'améliorer :

```
mod2.aic<-stepAIC(mod2,direction="backward",k=2)
mod2.aic
```

```
Call:  glm(formula = Y ~ X1 * X2, family = binomial(logit), data = simu)

Coefficients:
(Intercept)           X1           X2        X1:X2
   -0.37899      0.09295     -0.13598     -0.01011

Degrees of Freedom: 1999 Total (i.e. Null);  1996 Residual
Null Deviance:      2713
Residual Deviance: 2600         AIC: 2608
```

Cette étape à garder le même modèle, l'AIC ne peut donc pas être améliorer.

Le modèle mod2 qui tient compte d'interaction entre X1 et X2 est donc le meilleur possible pour modéliser <u>P(Y |X1, X2).</u>

<u>2- Prédictions</u>

On importe les données de l'échantillon « test » puis on estime avec notre modèle mod2 les probabilités associées soit P(Y |X1, X2) :

```
xsimutest=read.table("C:/Users/dionc/Desktop/projetf_analyse_donees/donnees/xsimutest.txt",header=TRUE)

predictions<-cbind(xsimutest, predict(mod2, newdata = xsimutest,type="response", se = TRUE))
head(predictions)
```

	X1	X2	fit	se.fit	residual.scale
1	1.5330012	5.794728	0.2470721	0.02246074	1
2	5.4239783	-1.849424	0.6173002	0.02656623	1
3	-5.1078449	-1.687702	0.3292807	0.02366467	1
4	0.8623463	-2.969889	0.5326832	0.01688372	1
5	-0.8073425	8.193683	0.1822257	0.02264142	1
6	-1.1385742	-2.164999	0.4463715	0.01514560	1

Les probabilités affichées (colonne « fit ») sont les probabilités du succès, en ne précisant rien lorsqu'on a transformé la variable Y en facteur le logiciel R a pris par défaut Y=1 échec et Y=2 succès. Grace à ces probabilités on prédit les valeurs de Y :

```
predictions$pred.Y = ifelse(predictions$fit > 0.5, 2, 1)
head(predictions)
```

	X1	X2	fit	se.fit	residual.scale	pred.Y
1	1.5330012	5.794728	0.2470721	0.02246074	1	1
2	5.4239783	-1.849424	0.6173002	0.02656623	1	2
3	-5.1078449	-1.687702	0.3292807	0.02366467	1	1
4	0.8623463	-2.969889	0.5326832	0.01688372	1	2
5	-0.8073425	8.193683	0.1822257	0.02264142	1	1
6	-1.1385742	-2.164999	0.4463715	0.01514560	1	1

On sauvegarde ces prédictions dans un fichier « .txt ».

```
write.table(predictions$pred.Y,"C:/Users/dionc/Desktop/projetf_analyse_donees/donnees/predictions.txt",row.names=F,col.names=F)
```

2. Races de chiens

Importation des données et préparation à l'analyse des correspondances multiples

On importe les données et on transforme les différentes variables renseignées en variables catégorielles.

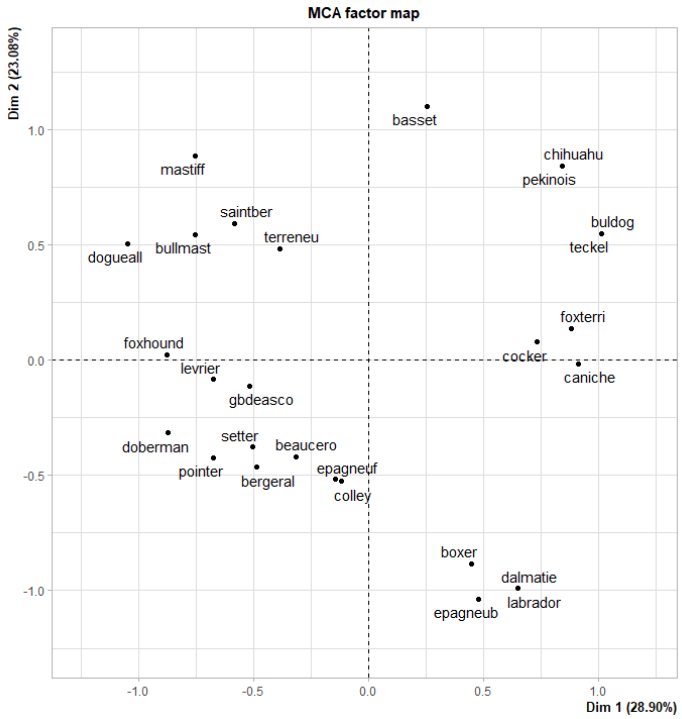
```
chiens=read.table("C:/Users/dionc/Desktop/projetf_analyse_donees/donnees/chiens",header=TRUE)

chiens$TAI=as.factor(chiens$TAI)
chiens$POI=as.factor(chiens$POI)
chiens$VEL=as.factor(chiens$VEL)
chiens$INT=as.factor(chiens$INT)
chiens$AFF=as.factor(chiens$AFF)
chiens$AGR=as.factor(chiens$AGR)
chiens$FON=as.factor(chiens$FON)
```

Analyse des correspondances multiples.

```
library(FactoMineR)
library(factoextra)
res.mca<-MCA(chiens, quali.sup=7, graph=TRUE)
```

On représente les individus dans le 1er plan factoriel :

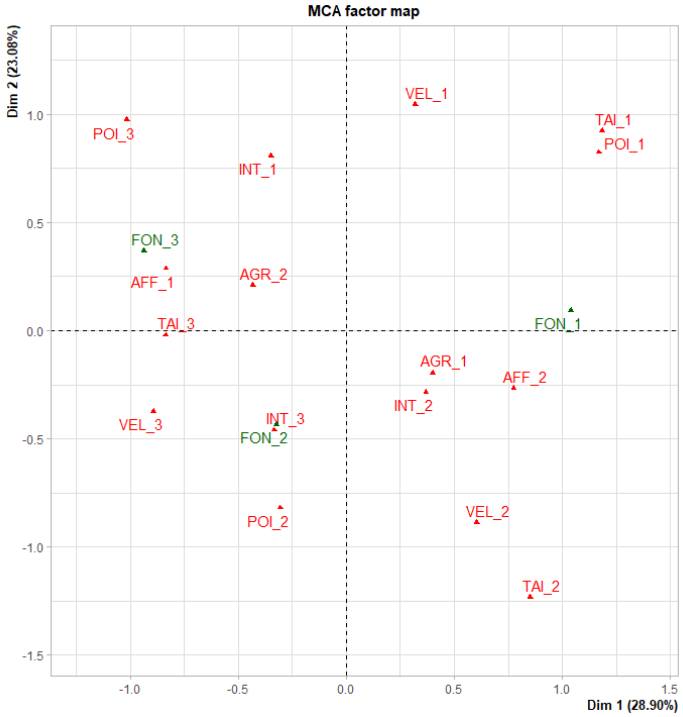
```
plot(res.mca, choix="ind",invisible=c("var","quali.sup"))
```



Le nuage de points des individus représenté dans le 1 er plan factorielle révèle une forme particulière. En effet, on voit 4 groupes d'individus se formé.

On représente les modalités dans le 1er plan factoriel :

```
plot(res.mca,choix="ind",invisible="ind")
```



Le 1er axe oppose les races de chiens grand, lourds, agressives avec une faible affectuosité aux races de chiens petits, légers, non agressive et affectueux. On peut donc supposer ici que le 1er axe oppose les races de chiens « imposantes » aux races de chiens « faibles ». Le second axe lui oppose les modalités (INT_1) aux modalités (INT_2, INT_3), il oppose donc les races de chiens intelligentes aux races de chiens peu intelligentes.

<u>Description automatique des axes :</u>

```
dimdesc(res.mca)
```

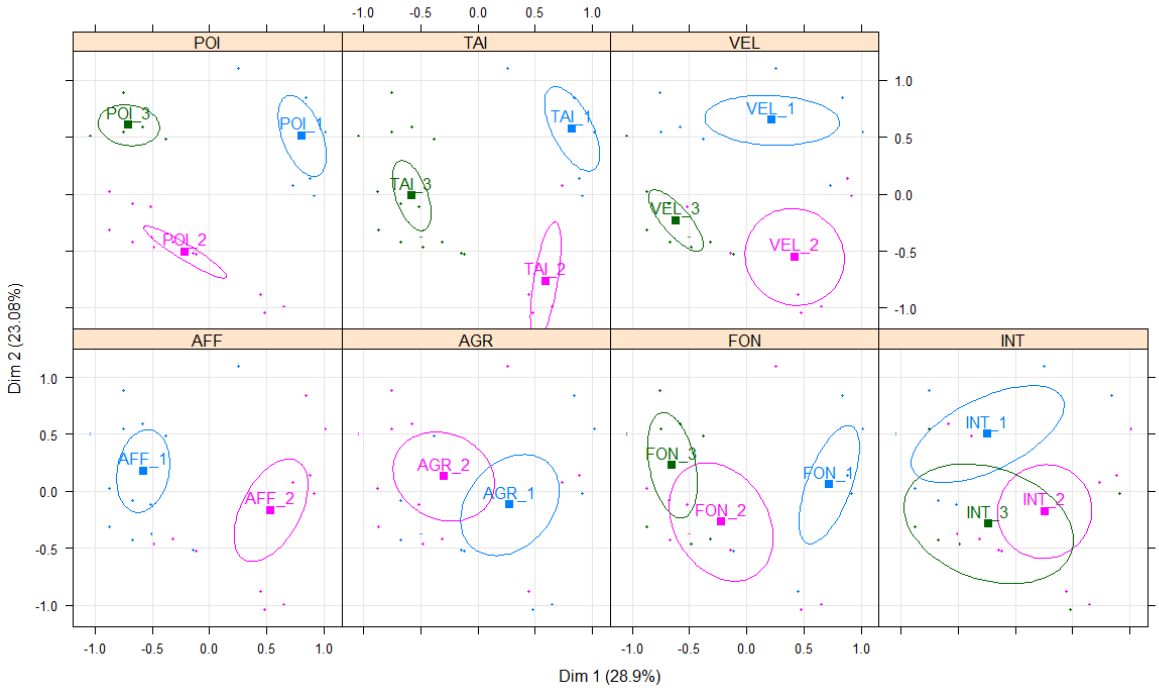
```
$`Dim 1`
$quali
            R2      p.value
TAI 0.8870733 4.300901e-12
AFF 0.6476559 4.184726e-07
FON 0.6945841 6.587193e-07
POI 0.6440465 4.137327e-06
VEL 0.4111741 1.737173e-03
AGR 0.1729238 3.098130e-02
```

```
$`Dim 2`
$quali
            R2      p.value
POI 0.7246877 1.896299e-07
VEL 0.6840074 9.911136e-07
TAI 0.5024857 2.299680e-04
INT 0.2798701 1.945048e-02
```

Ici le 1er axe est caractérisé par les variables taille, poids, affectuosité, agressivité et vélocité ainsi que la variable supplémentaire fonction.
Le 2nd axe lui est caractérisé par la variable intelligence et les variables taille, poids et vélocité.

Ellipses de confiance :

plotellipses(res.mca)



Finalement l'analyse des correspondances multiples des différentes races de chiens nous amène a considéré 4 types/groupes de races de chiens.

- Les races de chiens « imposantes » et peu intelligentes généralement utilisé comme chiens de garde (ex : Saintber, Mastiff, Terreneu).
- Les races de chiens « imposantes » et intelligentes généralement utilisé pour la chasse (ex : Espagneuf, Setter, Pointer).
- Les races de chiens « faibles » et peu intelligentes généralement utilisé comme chiens de compagnie (ex : Chihuahua, Pekinois, Tekkel, buldog).
- Les races de chiens « faibles » et intelligentes qui peuvent jouer le rôle de chien de compagnie ou chien de chasse (ex : Labrador, Dalmatie, Espagneub, Boxer).