Rapport fouilles de données en R

Dans ce rapport, nous allons expliquer notre démarche pour effectuer une analyse factorielle discriminante sur Rstudio avec différents jeu de données. Le but ici est d'expliquer les différentes étapes tout en analysant et exploitant les résultats obtenues pour notre data frame prematures.xlxs (on va également comparer les résultats obtenues avec ceux de VINQUALITE.txt). La dernière partie sera consacrée à la comparaison entre notre AFD et l'analyse linéaire discriminante.

Nous avons donc réalisé une fonction générale (encapsulation) capable de prendre en paramètre une data frame et d'en réaliser une analyse factorielle discriminante.
La première étape cruciale est la récupération des données et transformation du facteur à prédire en données numériques.

```
#-> .. on cherche la variable NON numÃ©rique ..
id_string  <- which(!sapply(df, is.numeric))

#-> pour faciliter et uniformiser les recherches des classes,
names(df)[id_string] <- 'fac'

#-> détection automatique des variables num?riques
id_num <- which(sapply(df, is.numeric))
X <- df[,id_num]
```

La donnée que l'on cherche à prédire est PREMATURE. On distingue deux classes qui sont les suivantes : positif et négatif. Pour les manipuler plus facilement, on les numérises en classe 1 et 2 respectivement.
Au tout début de notre code, on charge plusieurs librairies comme par exemple formattable qui permet d'obtenir des tableaux beaucoup plus soignées et non standards ou encore la librairie ggplot qui permettra par la suite de tracer des graphiques.

Pour prédire une variable, il faut étudier les moyennes, les variances ainsi et les utiliser pour calculer les inerties intra, inter et totale. C'est donc ce que nous avons fait :
⇒ calcul des moyennes par colonne : `mG <- colMeans(X)`
Ensuite on crée une boucle for qui permet de séparer les 2 classes, de retourner le nombre d'observation par classe, la moyenne par classe des différentes variable, les variances mais aussi les inerties intra, inter et totale :

```
df_List <- list()
W <- 0
B <- 0
for(i in 1:nlevels(df$fac))
{
  levels(df$fac)[i] <- i
  df_List[[i]] <- list(data=subset(X,df$fac==i), N = nrow(subset(X,df$fac==i)),mean = colMeans(subset(X,df$fac==i)),var = var(subset(X,df$f
  W <- W + (df_List[[i]]$N * (df_List[[i]]$var))
  B <- B + ((df_List[[i]]$N)*(unlist(df_List[[i]]$mean)-mG))%*%t(unlist(df_List[[i]]$mean)-mG)
}
```

Par exemple, pour la classe 1 :
-le nombre d'observation est le suivant : `[1] 124`
-les moyennes par colonne de la classe 1 :

```
[[1]]$mean
      GEST     DILATE     EFFACE     CONSIS      CONTR    MEMBRAN        AGE
30.6612903  0.6693548 26.8064516  2.1774194  1.0645161  1.9758065 27.0564516
```

Chacune des variables correspondent à des facteurs prénataux étudiés lors d'accouchement prématuré. Par exemple, la variable membrane permet de dire si les membranes sont rupturés ou non. On voit ici que la moyenne de cette variable se rapproche se 2, cela veut dire qu'en moyenne, chez les 124 femmes dont celles la pré-maturation a été positif n'ont pas eu leurs membranes rupturées.

-enfin le calcul de la variance a donné le résultat suivant :

	GEST	DILATE	EFFACE	CONSIS	CONTR	MEMBRAN	AGE	STRAT	GRAVID	PARIT
GEST	8.99817898	0.234781478	10.5231530	0.495577523	-0.220083247	0.136966701	-0.81152445	2.11745578	-0.52159209	-0.585848075
DILATE	0.23478148	0.608415713	3.1376171	0.155437045	-0.018990635	0.008129553	-0.10230229	0.04207856	0.15998959	0.091311134
EFFACE	10.52315297	3.137617066	677.1883455	6.566597294	-0.318158169	1.051768991	-3.66649324	2.36550468	3.23881374	0.264308012
CONSIS	0.49557752	0.155437045	6.5665973	0.565296566	-0.019510926	0.004292404	-0.21162851	0.09170135	-0.02133195	-0.070239334
CONTR	-0.22008325	-0.018990635	-0.3181582	-0.019510926	0.076482830	-0.022632674	-0.01170656	-0.05463059	0.06555671	0.030176899
MEMBRAN	0.13696670	0.008129553	1.0517690	0.004292404	-0.022632674	0.071995317	-0.12766649	0.03258325	-0.06087409	-0.071800208
AGE	-0.81152445	-0.102302289	-3.6664932	-0.211628512	-0.011706556	-0.127666493	27.76294225	-0.10828564	3.01300728	2.586888658
STRAT	2.11745578	0.042078564	2.3655047	0.091701353	-0.054630593	0.032583247	-0.10828564	0.58136056	-0.07908429	-0.083766909
GRAVID	-0.52159209	0.159989594	3.2388137	-0.021331946	0.065556712	-0.060874089	3.01300728	-0.07908429	2.07232050	1.395681582
PARIT	-0.58584807	0.091311134	0.2643080	-0.070239334	0.030176899	-0.071800208	2.58688866	-0.08376691	1.39568158	1.369927159
DIAB	0.07010926	-0.035054631	-1.3228408	-0.079734651	-0.010665973	0.009040062	-0.10711498	0.03440427	-0.01144641	0.006763788
TRANSF	0.02523413	-0.036810614	1.0374610	-0.002081165	-0.004422477	0.008714880	0.13826743	0.01755983	0.04344433	0.026014568
GEMEL	0.08350676	0.018730489	0.3449532	0.026534860	-0.002081165	0.000780437	-0.08246618	0.02107180	0.01664932	0.015088450

	DIAB	TRANSF	GEMEL
GEST	0.070109261	0.025234131	0.083506764
DILATE	-0.035054631	-0.036810614	0.018730489
EFFACE	-1.322840791	1.037460978	0.344953174
CONSIS	-0.079734651	-0.002081165	0.026534860
CONTR	-0.010665973	-0.004422477	-0.002081165
MEMBRAN	0.009040062	0.008714880	0.000780437
AGE	-0.107114984	0.138267430	-0.082466181
STRAT	0.034404266	0.017559834	0.021071800
GRAVID	-0.011446410	0.043444329	0.016649324
PARIT	0.006763788	0.026014568	0.015088450
DIAB	0.409664412	0.020421436	-0.001300728
TRANSF	0.020421436	0.212539022	0.001821020
GEMEL	-0.001300728	0.001821020	0.031217482

La variance est pertinente car elle permet d'évaluer la dispersion des valeurs au sein de notre échantillon. Nous allons réutiliser la variance pour calculer les inerties.
⇒ La boucle for s'applique également sur les valeurs de la classe 2 donc tous les éléments au-dessus sont de même calculer pour cette classe.
Par exemple, on constate que pour les patientes avec pré-maturation négative, la moyenne de la variable transfert à l'hôpital tend vers 2 tandis pour les patientes de la classe 1, cette moyenne tend plus vers 1 soit un transfert à l'hôpital.

Nous avons commencer par calculer l'inertie intra W qui correspond à la multiplication entre l'inverse du nombre d'observation totale et la somme du nombre d'observation par classe multiplié à la variance par classe. Ensuite nous calculons l'inertie inter qui correspond au produit entre l'inverse du nombre d'observation totale et la multiplication entre la différence des moyennes par classe et des moyennes générale, et la transposée de cette différence.

⇒ On somme ces deux inerties pour obtenir l'inertie totale :

```
[1] "Inertie Totale(Tot): "
```

	GEST	DILATE	EFFACE	CONSIS	CONTR	MEMBRAN	AGE	STRAT	GRAVID	PARIT
GEST	9.86742932	-0.15754109	12.2600526	0.191558185	-0.044089415	-0.0864168310	0.727587114	2.400959895	-0.678724523	-0.361485865
DILATE	-0.15754109	1.70600920	24.1587903	0.336213018	0.033543721	-0.0319000657	-0.495016437	-0.059513478	-0.142156476	-0.044286654
EFFACE	12.26005260	24.15879027	1209.9790664	10.792662722	0.981433268	-1.8271268902	-7.308389218	2.245654175	-4.739947403	-5.194266930
CONSIS	0.19155819	0.33621302	10.7926627	0.520453649	0.003806706	-0.0122879684	-0.141479290	0.029428008	-0.136646943	-0.100670611
CONTR	-0.04408941	0.03354372	0.9814333	0.003806706	0.096127548	-0.0222945431	-0.014404997	-0.008586456	-0.008191979	-0.007284681
MEMBRAN	-0.08641683	-0.03190007	-1.8271269	-0.012287968	-0.022294543	0.2302498356	-0.204464168	-0.027245233	0.013583169	-0.017896121
AGE	0.72758711	-0.49501644	-7.3083892	-0.141479290	-0.014404997	-0.2044641683	26.455496384	0.184589086	2.217330703	1.626482577
STRAT	2.40095989	-0.05951348	2.2456542	0.029428008	-0.008586456	-0.0272452334	0.184589086	0.661906640	-0.140065746	-0.070756082
GRAVID	-0.67872452	-0.14215648	-4.7399474	-0.136646943	-0.008191979	0.0135831690	2.217330703	-0.140065746	2.098198554	1.048770546
PARIT	-0.36148586	-0.04428665	-5.1942669	-0.100670611	-0.007284681	-0.0178961210	1.626482577	-0.070756082	1.048770546	1.012912558
DIAB	-0.08724523	-0.04720579	-1.9994740	-0.043392505	-0.007560815	-0.0051939513	0.134845496	-0.016042078	-0.012886259	0.013346483
TRANSF	0.11251808	-0.14791584	-2.3457857	-0.072426036	-0.018369494	0.0330309007	0.490032873	0.024155161	0.112518080	0.034727153
GEMEL	-0.14820513	0.09128205	1.0610256	0.035128205	0.011025641	-0.0007692308	-0.005897436	-0.030256410	0.003076923	-0.011282051

	DIAB	TRANSF	GEMEL
GEST	-0.087245233	0.11251808	-0.1482051282
DILATE	-0.047205786	-0.14791584	0.0912820513
EFFACE	-1.999474030	-2.34578567	1.0610256410
CONSIS	-0.043392505	-0.07242604	0.0351282051
CONTR	-0.007560815	-0.01836949	0.0110256410
MEMBRAN	-0.005193951	0.03303090	-0.0007692308
AGE	0.134845496	0.49003287	-0.0058974359
STRAT	-0.016042078	0.02415516	-0.0302564103
GRAVID	-0.012886259	0.11251808	0.0030769231
PARIT	-0.013346483	0.03472715	-0.0112820513

L'inertie est très importante à étudier dans notre cas car elle va nous permettre d'étudier les dispersions entre les différents groupes et va nous aider dans la détermination du nombres d'axes utiles pour garder le maximum d'information.

Pour regrouper les individus qui se ressemblent, et séparer ceux qui ne se ressemblent pas, on doit examiner l'ensemble des informations dont on dispose et en calculer l'inertie afin de prédire la variable prématuré. C'est en rassemblant les individus qui se ressemblent que l'on peut dire si celui

ci fait partie de la classe 1 ou 2. L'étude de la corrélation entre variables est donc extrêmement pertinente ici.

Une fois que l'on dispose de tous ces éléments, on va pouvoir procéder à la diagonalisation. On cherche à calculer les axes qui discriminent au mieux les groupes, en maximisant la dispersion inter-groupe et minimisant la dispersion intra groupe (raison pour laquelle on a calculé les inerties). Pour cela, la méthode de Fisher est très utile car elle va nous permettre d'étudier la projection B/W et d'en calculer les axes factoriels.

```
#DIAGONALISATION#
B2 <- B * (N/(k-1))
W2 <- W * (N/(N-k))
rapportBW <- as.matrix(B2%*%solve(W2))
```

On pondère les inerties et on procède au calcul du rapport. On enchaîne avec le calcul des valeurs propres à l'aide de la fonction eigen() ainsi que des vecteurs propres normés avec la fonction sweep(). Les vecteurs propres calculés par la fonction sweep() permettent de réduire et de centrer les données d'où la normalisation.

```
#VALEUR PROPRE#
u <- eigen(rapportBW)
uReal <- Re(u$vectors)

reduxUvectors <- uReal[,1:2]
reduxUvalues <- Re(u$values[1:2])

print("Valeur propre: ")
print(reduxUvalues)


normeVal <- sqrt(diag(t(reduxUvectors)%*%W%*%reduxUvectors))
normeVec <- reduxUvectors/normeVal

VecP <- sweep(reduxUvectors,2,normeVal,'/')
```

Les vecteurs propres correspondent aux vecteurs directeurs des axes factoriels et les valeurs propres correspondent à la dispersion, soit la variance projetée sur l'axe factoriel.

```
[1] "Valeur propre: "
[1]  1.287774e+02 -1.011766e-14
```

Étant donné que notre jeu de donnée ne comporte que 2 classes (positif ou négatif), le nombres d'axes à retenir est égale à 1. Les valeurs propres obtenues ci-dessus sont celles des deux axes factoriels. Comme on le constate, il n'y a quasi pas d'informations expliquer par l'axe 2. En effet, la part de variance expliquée par le premier axe est de 128.7774/128.7774 – 1.011e-14 ce qui est environ égale à 1 soit 100 % de la dispersion inter expliqué par le premier axe à lui tout seul. Cela témoigne de la pertinence de la diminution de dimensionnalité.

Pour être sur de notre choix des axes factoriels, nous allons construire un test statistique qui va justement venir valider le caractère discriminant des axes factoriels.

La construction du test se fait en plusieurs étapes :
⇒ la part de dispersion expliqué par les axes Iw
⇒ la corrélation canonique Ncorr
⇒ la quantité delta i par axe à l'aide de la fonction prod
⇒ calcul de la valeur test Epsilon
⇒ détermination du degré de liberté
⇒ calcul de la p-valeur avec le test du Chi2 au seuil de 0,05

```
p <- ncol(df) - 1
liberty <- (p - seq(0,(ncol(VecP)-1),1)) * (k-1-seq(0,(ncol(VecP)-1),1))
inf_List <- list()
for (y in 1:(k-1))
{
  Iw = reduxUvalues[y]/sum(reduxUvalues)

  Ncorr = sqrt(reduxUvalues[y]/(1 + reduxUvalues[y]))

  A = prod(1 - Ncorr**2)

  E = -(N-((p+k)/2)-1)*log(A)

  p_value <- 1-pchisq(E,liberty)

  inf_List[[y]] <- list(Iw=Iw,Ncorr = Ncorr,A = A,E = E,p_value = p_value[1])

}
```

La p-valeur permettre de dire si la part de dispersion expliqué par l'axe 1 est supérieure ou non que la part de dispersion expliquée par l'axe 2 et donc par conséquent de rejeter ou accepter l'hypothèse correspondant.

```
[[1]]
[[1]]$Iw
[1] 1

[[1]]$Ncorr
[1] 0.9961398

[[1]]$A
[1] 0.007705499

[[1]]$E
[1] 1856.311

[[1]]$p_value
[1] 0
```

Les résultats du test n'étant pas pertinent pour l'axe 2, nous l'avons afficher uniquement pour l'axe 1.

Une fois que l'on a déterminé nos axes factoriels, on va pouvoir calculer les coordonnées des individus dans les axes : les scores.

Pour cela on commence par stocker dans Z la matrice des observations centrées qui correspond à la différence entre la data frame sans la colonne fac et la matrice mG avec les moyennes générale par colonne. Ensuite multiplie cette matrice avec la matrice des vecteurs propres VecP, ce qui va nous permettre d'obtenir les scores sur les deux axes. On regroupe les scores et la colonne fac dans une même matrice et on l'affiche :

```
df_matrix <- unname(as.matrix(sapply(X,as.numeric)))
mg_matrix <- matrix(unlist(mG), nrow=N, ncol = C, byrow = TRUE)

Z <- as.matrix(df_matrix - mg_matrix)
Z

pre_score = Z%*%VecP
score = cbind(pre_score,df[ncol(df)])
score
name_list <- list()
name_list[1] <- "Axe_1"
for (i in 2:ncol(score))
{
  if (i != ncol(score))
  {
    name_list[i] <- paste(c("Axe",i),collapse = "_")
  }else
  {
    name_list[i] <- "fac"
  }
}
print("Score: ")
print(score)

names(score) <- unlist(name_list)
names(score)[ncol(score)] <- 'class' <- 'class'
score
```

```
[1] "Score: "
           1           2 fac
1  -1.71220609  1.70947356   2
2   1.33085079 -1.34459381   2
3  -1.70970718  1.71139069   2
4  -0.94946086  0.95131814   2
5  -0.96603625  0.94056333   2
6  -1.71126905  1.70723044   2
7  -0.95024312  0.94474144   2
8  -0.18066209  0.18627843   2
9   0.57594044 -0.58692410   1
10  1.34126769 -1.34185958   2
11 -1.70972100  1.71496331   2
12 -1.71738470  1.70340873   2
13 -0.18236530  0.18833663   2
14 -0.19614082  0.18178528   2
15  0.58774025 -0.57451779   1
16 -0.19249359  0.17861234   2
17  1.34037328 -1.34581277   2
18 -1.71378949  1.70853714   2
19  1.34990701 -1.33950847   1
20  1.33297496 -1.35171705   2
21  1.34199777 -1.34685186   1
```

Le ggplot qui suis ce code permet tout simplement de représenter les individus dans le plan factoriel.

Nous allons réutiliser les scores dans le calcul des centres de gravité. En effet, le centre de gravité est le point où l'inertie totale est minimal. On récupère et on regroupe les données des axes factoriels à l'aide de la fonction aggregate() pour appliquer la fonction mean() dessus.

```
axes <- NULL
for( i in 1: (ncol(score)-1))
{
  axes <- cbind(axes,as.numeric(score[,i]))
}

G <- cbind(pre_score,df[ncol(score)])
G_fac <- aggregate(axes,list(df$fac),mean)

for(i in 2:ncol(G_fac))
{
  names(G_fac)[i] <- paste(c("Axe",i-1),collapse = "_")
}

print("Centre de gravitÃ©: ")
print(G_fac)
```

Ce code nous permet d'obtenir les centre de gravité des deux classes en fonction des différents axes factoriels tel que :

```
[1] "Centre de gravitÃ©: "
  Group.1      Axe_1      Axe_2
1       1  0.5261960 -0.5241269
2       2 -0.2452944  0.2443298
```

On voit qu'en moyenne la classe 1 se trouve dans la partie négative de l'axe 2 et dans la partie positive de l'axe 1 et inversement pour la classe 2. Il y a donc bel et bien une séparation entre les deux groupes, et par conséquent la moyenne de chaque observation de chaque groupe est localisé différemment.

Enfin, on calcule les distances euclidiennes entre les individus et les centres de gravité. On calcule la distance pour chaque observation au centre de gravité de chaque classe dans le plan factoriel.

Comme on peut le voir dans le code ci-dessous, on crée une boucle for dans laquelle on stock dans la variable delta la différence entre la matrice des scores et la matrice des centres de gravités, et on injecte ce delta dans la fonction qui permet de calculer la distance euclidienne, le résultat étant stocké dans le variable distance.

```
distance <- NULL

for (i in 1:nrow(G))
{ delta <- score[1:ncol(score)-1] - matrix(G[i,],ncol = ncol(G), nrow = N, byrow = T)
distance <- cbind(distance,apply(delta,1,function(x){return(sqrt(sum(x*x)))}))
}

df_distance <- as.data.frame(distance) ; names(df_distance) <- G_fac[,1]
df_distance
print("Distance individus/centre de gravitÃ©: ")
print(df_distance)
```

Les distances obtenues en fonction des classes sont les suivantes :

```
         1         2
3.16218512 2.07327668
1.14918901 2.23805985
3.16177192 2.07286576
2.08674414 0.99783914
2.09095060 1.00210277
3.15993754 2.07102873
2.08265372 0.99374503
1.00215968 0.08687517
0.08011244 1.16850746
1.15456861 2.24347408
3.16430870 2.07540555
3.16157700 2.07267007
1.00482005 0.08423363
1.00999130 0.07954804
0.07954208 1.16810020
1.00516520 0.08430131
1.15674215 2.24563919
3.16264502 2.07373601
1.15902848 2.24793487
1.15576721 2.24461571
```

Avec toutes les données précédentes, nous allons enfin pouvoir prédire l'appartenance des individus aux différents groupes. À l'aide de toutes les étapes précédentes, nous avons réalisé progressivement une analyse discriminante factorielle, l'étape suivante est celle de la prédiction :
⇒ On affecte à chaque individu le groupe dont le distance euclidienne est la plus petite possible :

```
Quality_pred <- factor(apply(df_distance,1,function(x){return(names(x)[which.min(x)])}))
Quality <- head(df$fac, "ind")

Qtab <- as.data.frame(cbind(Quality= as.character(Quality),Quality_pred = as.character(Quality_pred)))
```

Une fois avoir prédit le groupe, nous allons le comparer au gold standard. C'est pourquoi on regroupe dans Qtab, la colonne fac du gold standard, et le colonne fac prédite avec l'AFD, le résultat étant le suivant :

```
              Quality
Quality_pred   1   2
           1  76  83
           2  48 183
```

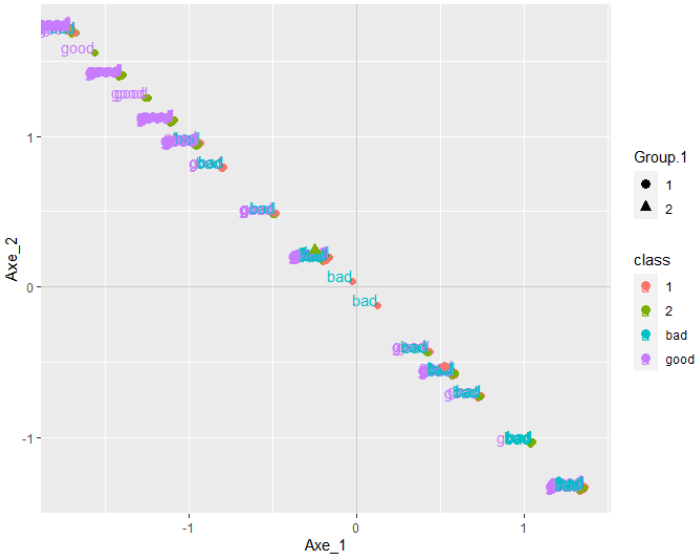
On voit alors notre AFD n'est pas forcément la meilleure technique à utiliser pour prédire les données. En effet, on constate que pour la classe 1 (positive), il y a 76 prédictions exactes contre 83 mauvaises prédictions ce qui n'est vraiment pas terrible. Pour la classe 2 (négative), il y a 183 prédictions exactes contre 48 prédictions fausses, ce qui est beaucoup mieux qu'avant.

C'est pourquoi nous faisons la matrice de confusion pour évaluer l'exactitude de notre AFD :

```
               Accuracy : 0.6641
                 95% CI : (0.6148, 0.7109)
    No Information Rate : 0.6821
    P-Value [Acc > NIR] : 0.793240

                  Kappa : 0.2798

 Mcnemar's Test P-Value : 0.002972

            Sensitivity : 0.6129
            Specificity : 0.6880
         Pos Pred Value : 0.4780
         Neg Pred Value : 0.7922
             Prevalence : 0.3179
         Detection Rate : 0.1949
   Detection Prevalence : 0.4077
      Balanced Accuracy : 0.6504

       'Positive' Class : 1
```

L'exactitude de la prédiction s'élève à 66 % ce qui n'est pas catastrophique, mais ça reste quand même compliquer de considérer cette AFD comme 'correcte'. Cela veut dire qu'il y a forcément des failles dans notre jeu de données du départ.

L'utilisation de l'analyse discriminante factorielle n'est pas forcément pertinente ici.



Le graphique à gauche correspond au placement des individus dans le plan factoriel. Il est tout à fait normal que l'on obtient une droite car comme vu précédent, on travaille uniquement avec 2 classes, soit un nombre d'axe à retenir égal à 1. Ce graphique n'est donc sans intérêt (affichage des coordonnées uniquement sur un seul axe donc aucune comparaison possible avec d'autres axes) car on ne peut tirer aucune conclusion pertinente de celui-ci. Par ailleurs, une AFD à deux classes correspond tout simplement à une MANNOVA !

Comparaison avec l'analyse linéaire discriminante :
⇒ On va tout simplement calculer le modèle LDA en utilisant la fonction lda déjà existante, et en renseignant les paramètres nécessaires. On commence la variable à expliquer et les variables explicatives, ensuite on renseigne la data frame sans la colonne fac, suivi d'un vecteur contenu les nombres d'observation par classe, et en indiquant FALSE dans CV.
⇒ On récupère la variable prédite en faisant pred$class, et l'on crée une table avec la prédiction LDA et le gold standard pour pouvoir comparer. Enfin, on réalise la confusion matrix pour évaluer l'exactitude de la LDA sur notre jeu de données :

```
prior <- list()
#LDA#
for (i in 1:(length(df_List)))
{
  prior[i] <- as.vector(df_List[[i]]$N/N)
}
model <- lda(Quality~., df[,-ncol(df)],prior = unlist(prior), CV = FALSE)
pred <- predict(model)
predclass <- pred$class
pred$posterior

lda_Reference <- table(predclass,Quality)
lda_confusion <- confusionMatrix(lda_Reference)
print("Matrice de confusion LDA: ")
print(lda_confusion)
```

```
         Quality
predclass   1   2
        1  61  37
        2  63 229
```

On voit que les prédictions sont beaucoup + réussis qu'avec l'AFD : la classe 1 a 61 prédictions correctes contre 37 fausses et la classe 2, 229 exactes et 63 fausses. Les résultats sont donc bien meilleurs.

```
               Accuracy : 0.7436
                 95% CI : (0.6972, 0.7862)
    No Information Rate : 0.6821
    P-Value [Acc > NIR] : 0.004701

                  Kappa : 0.3738

 Mcnemar's Test P-Value : 0.012419

            Sensitivity : 0.4919
            Specificity : 0.8609
         Pos Pred Value : 0.6224
         Neg Pred Value : 0.7842
             Prevalence : 0.3179
         Detection Rate : 0.1564
   Detection Prevalence : 0.2513
      Balanced Accuracy : 0.6764

       'Positive' Class : 1
```

L'exactitude de la LDA s'élève à 75 % ce qui acceptable, et même bien mieux que les 66 % de l'AFD. On constate alors une meilleure caractérisation des données avec l'analyse linéaire discriminante, on peut donc rejeter l'AFD pour ce jeu de données. La LDA est correcte, et peut même être optimisée.

Dayana BAKTHAVATCHALAM
Bastien BAGUR