Ecole Nationale Supérieure des Techniques Avancées

Rapport de Projet

STA203

Emma de Charry et Blandine Chabert

Apprentissage staistique

Jazz et classique : quelles différences ?

Table des matières

1 Introduction

Le projet d'apprentissage statistique sur lequel nous travaillons vise à explorer et à appliquer différentes méthodes de classification pour différencier les morceaux de jazz et de musique classique. Les données que nous utilisons sont extraites d'un challenge de reconnaissance de genre de musique, et sont contenues dans le fichier **Music_2023.txt**. Ce jeu de données ne contient que des morceaux de jazz et de musique classique, ce qui en fait un ensemble de données parfait pour entraîner des modèles de classification et les mettre en compétition. Notre objectif est de développer des modèles précis et fiables capables de distinguer avec précision les genres de musique dans ce jeu de données.

2 Partie I

2.1 Question 1 : Etude du comportement des variables

On commence par une analyse descriptive du jeu de données. Comme le nombre de variable est très important on en sélectionne seulement une partie. On trace les boîtes à moustache pour chacune de ces variables puis le plot des corrélations. (les plots sont en annexe) Malgrés la réduction du nombre de variable, il est difficile de tirer des conclusions de cette analyse car les variables sont très nombreuses. On remarque tout de même qu'il existe des variables plus ou moins corrélées. La proportion de musique de Jazz est d'environ 47% contre 53% de musique classique.

On extrait les valeurs concernées et on regarde ce qui pourrait justifier d'appliquer le log. PAR_SC_V prend des grandes valeurs tandis que PAR_ASC_V prend des valeurs proches de 0 ce qui explique qu'il peut être judicieux d'appliquer le log. Concernant les variables 148 à 167 on les supprime car en annexe on voit que c'est les mêmes que 128 à 147.

On utilise la matrice de corrélation triangularisée pour identifier des variables très corrélées. En ce qui concerne les variables très corrélées, il est possible de les supprimer pour éviter la redondance d'informations. On obtient que l'on peut supprimer les variables 37, 72, 164. Les variables PAR_ASE_M, PAR_ASE_MV, PAR_SFM_M et PAR_SFM_MV ce sont des valeurs moyennes, elles dépendent linéairement des autres valeurs donc on peut donc également les retirer.

```
C = cor(new_M[,1:171]) - diag(1,171)
C[upper.tri(C)] <- 0
paires_cor=cbind(which(C>0.99)%/%171 +1, which(C>0.99)%%171)
#on obtient
#[,1] [,2]
#[1,]   36   37
#[2,]   71   72
#[3,]  160  164
```

On peut définir le modèle logistique à l'aide de la fonction glm. Les hypothèses du modèle logistique comprennent l'indépendance des observations.Au vu des dépendances observées entre les variables les hypothèses ne sont donc pas toutes vérifiées.

```
res = glm(GENRE~.,data=complete,family=binomial)
```

2.2 Question 2 : Choix des données d'appprentissage et de test

On définit l'échantillon d'apprentissage.La commande ptoposée permet de créer un vecteur logique de longueur n avec des valeurs TRUE et FALSE en utilisant une distribution de probabilité donnée. Les TRUE et FALSE indiquent si oui ou non les données vont faire partie de l'échantillon d'apprentissage.

2.3 Question 3 : Détermination des modèles

On souhaite maintenant faire une estimation de différents modèles logistiques :

- Mod0 formé des variables PAR_TC, PAR_SC, PAR_SC_V, PAR_ASE_M, PAR_ASE_MV, PAR_SFM_M, PAR_SFM_MV.

```
res = glm(GENRE~., data=complete, family=binomial)
```

- ModT contenant toutes les variables que vous aurez retenues à la question 1. Pour cela on doit d'abbord redéfinir le tableau de données pour effectuer les modifications prévues à la question 1.

```
new_complete_1=complete
new_complete_1$PAR_SC_V=log(complete$PAR_SC_V)
new_complete_1$PAR_ASC_V=log(complete$PAR_ASC_V)
new_complete_2=cbind(new_complete_1[,1:147],new_complete_1[,168:192])
new_complete_3=new_complete_2[,c(-37, -72, -164)]
new_complete <- new_complete_3[, setdiff(names(new_complete_3),
c("PAR_ASE_M", "PAR_ASE_MV", "PAR_SFM_M", "PAR_SFM_MV"))]
```

On peut ensuite utiliser à nouveau la fonction glm.

```
ModT <- glm(GENRE ~ ., data = new_complete , family = binomial, subset = train)
```

- Mod1 formé par toutes les variables significatives au niveau 5% dans ModT. De même, Mod2 formé par toutes les variables significatives au niveau 20% dans ModT. On procéde de manière similaire à mod1.

```
train.M <- complete[train,]
test.M <- complete[!train,]
sum.ModT = summary(ModT)
names_0.05 <- names(which(sum.ModT$coefficients[,4] <= 0.05))
names_0.2 <- names(which(sum.ModT$coefficients[,4] <= 0.2))
```

On peut ensuite utiliser à nouveau la fonction glm.

```
Mod1<- glm(GENRE~. ,
    data = train.M[, c("GENRE", names_0.05[-1])],
    family = binomial)

Mod2<- glm(GENRE~. ,
    data = train.M[, c("GENRE", names_0.2[-1])],
    family = binomial)
```

- ModAIC obtenu par sélection de variables stepwise (stepAIC) sur critère AIC. On utilise la fonction stepAIC et on l'applique au modèle modT.

```
st1 = stepAIC(ModT)
ModAIC = glm(GENRE~.,data=st1$model)
```

2.4 Question 4 : Tracé des courbes ROC

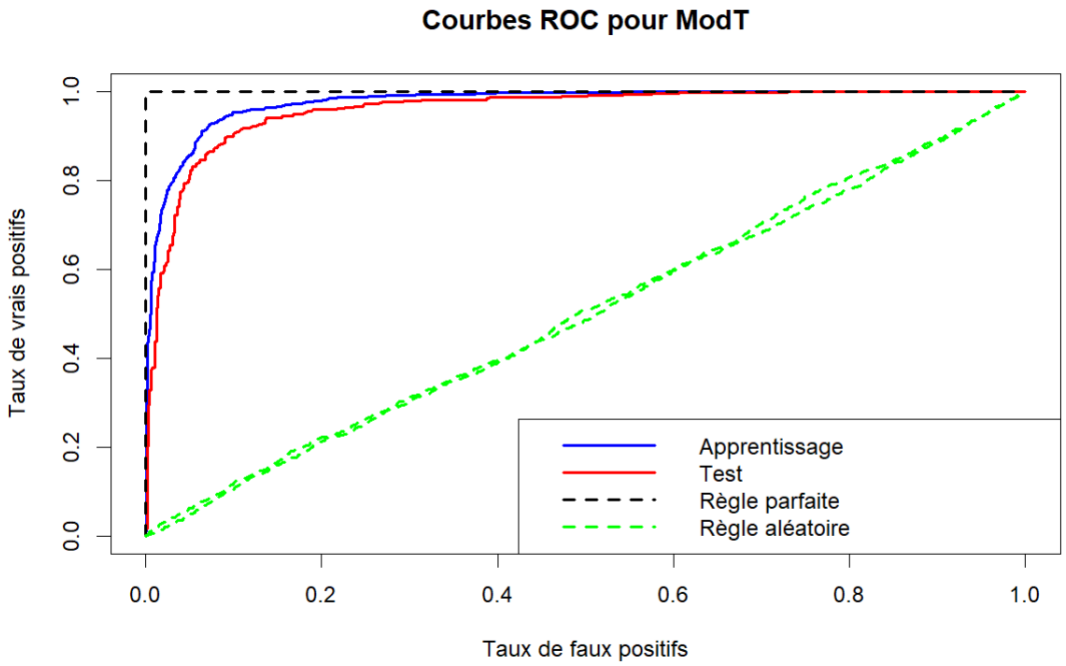
On souhaite maintenant tracer les courbes ROC des modèles obtenus. On sait que la courbe ROC est un graphique qui permet d'évaluer la performance d'un modèle de classification binaire. Elle est tracée en utilisant les taux de vrais positifs (TVP) et les taux de faux positifs (TFP) calculés à différents seuils de classification. Ces taux sont calculée de la manière suivante :

$$TFP = \frac{FP}{FP + TN}$$

$$TVP = \frac{TP}{TP + FN}$$

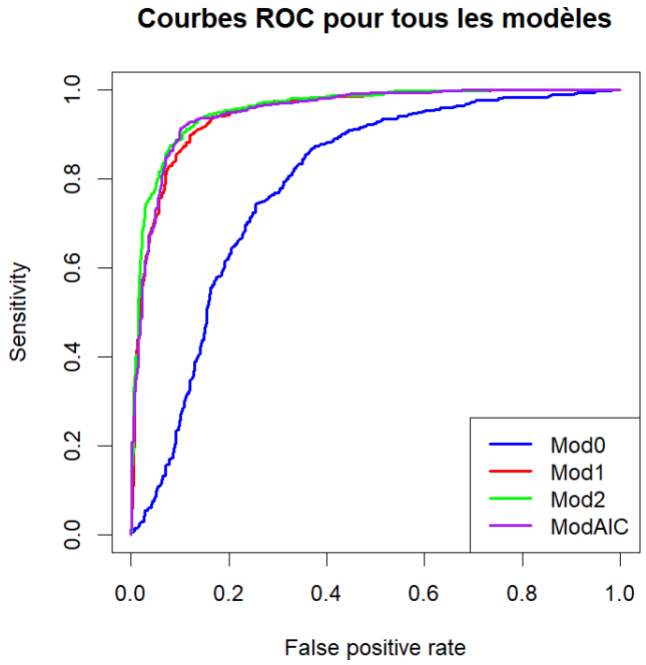
où TP est le nombre de vrais positifs, FP est le nombre de faux positifs, TN est le nombre de vrais négatifs et FN est le nombre de faux négatifs.

On ajoute à titre comparatif la courbe de la règle parfaite et la courbe de la règle aléatoire. Ces courbes correspondent à un classement parfait et un classement aléatoire des données. On obtient :



On observe une meilleure performance du modèle ModT sur les données d'apprentissage que sur les données de test ce qui est logique car le modèle à été conçu pour être représentatif des données d'apprentissage. En effet, le modèle glm est ajusté en maximisant la fonction de vraisemblance pour les données d'apprentissage. Les courbes et rouge sont proches donc le modèle permet tout de même d'avoir une bonne représentation des données test.

On détermine ensuite les courbes ROC pour tout les modèles définis précédements. On obtient :



Pour comparer les modèles on observe l'air sous les courbes ROC. PLus l'air se rapproche de 1 plus on se rapproche du modèle parfait. On peut voir à l'oeil nu que le modèle Mod0 est netement moins pérformant. Pour comparer les pérformances des autres modèles il faut calculer précisement l'aire sous la courbe ROC pour chacun des modèles. On obtient que ModAIC et ModT sont les plus performants puis vient Mod2 puis Mod1 et enfin Mod0. L'equilibre entre Mod1 et Mod2 correspond à l'ajustement lié au compromis biais/variance. Mod2 a une plus grand variance car un plus grand nombre de paramètres.

2.5 Question 5 : Choix du meilleur modèle

On affiche le calcul de l'erreur sur les différents échantillons. On obtient que le modèle qui a une bonne courbe ROC tout en ayant une erreur pas trop élevée est Mod2.
On remarquera que l'on a pas pu calculer l'erreur pour ModAIC car nous avons sans faire exprès run tout le code à nouveau et nous n'avons pas le temps de refaire le calcul du modèle AIC.

3 Partie II

3.1 Question 1

La régression **ridge** est une technique de régression linéaire qui permet de régulariser les coefficients du modèle en ajoutant une pénalité sur leur magnitude. Cela revient à un problème de minimisation sous contraintes. Cette pénalité est contrôlée par un paramètre λ, qui permet de contrôler le degré de régularisation.

Dans le contexte de notre étude sur la classification des genres de musique jazz et classique, la régression ridge pourrait améliorer la performance des modèles de classification en réduisant l'impact de variables non pertinentes ou redondantes. La régression ridge peut aider à réduire la variance du modèle en stabilisant les coefficients et en réduisant les effets de sur-ajustement.

Plus précisément, la régression ridge peut être utilisée pour sélectionner les caractéristiques les plus discriminantes dans le jeu de données, en réduisant l'influence des caractéristiques moins importantes. Elle peut également aider à contrôler la taille des coefficients pour éviter les problèmes de colinéarité et améliorer la stabilité des prévisions.

3.2 Question 2

λ variant de 10^{10} à 10^{-2}, on définit **grid** qui est le vecteur des valeurs de la pénalité, de la manière suivante :

```
grid = 10^seq(10,-2,length=100)
```

Pour λ proche de 10^{10} les coefficients de la régression sont très petits, et d'un autre côté pour λ proche de 10^{-2} les coefficients ne sont pas forcéments petits, ils peuvent prendre différentes valeurs. Donc si les coefficients ne nous importent pas nous choisissons un λ petit.
Le graphique tracé par la fonction **plot** appliquée à l'objet sorti par **glmnet** est le suivant :

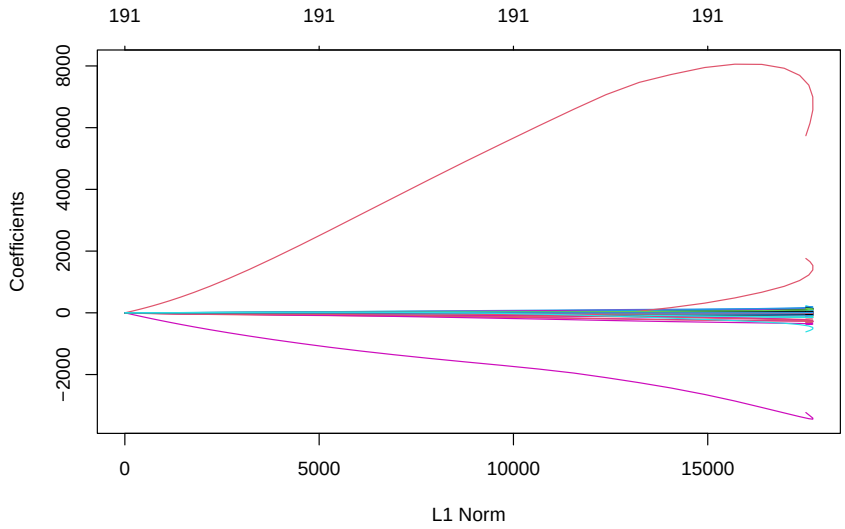


FIGURE 1 – Régression Ridge appliquée à Music_2023

Le graphique présente l'évolution des coefficients de régression pour chaque prédicteur en fonction de la norme L1 du paramètre. L'analyse de ces courbes permet de sélectionner les prédicteurs les plus pertinents pour prédire la variable dépendante "GENRE" et de déterminer la valeur optimale de la pénalité pour obtenir un modèle de régression efficace et adapté. En observant le graphique on remarque que sont mises en évidence deux courbes (rouge et violette) nettement plus élevées que les autres, ce qui peut indiquer que les variables correspondantes sont plus importantes pour prédire la variable "GENRE" que le reste des variables.

3.3 Question 3

L'algorithme de validation croisée en 10 segments **cv.glmnet** s'explique de la manière suivante : l'algorithme consiste à diviser l'échantillon d'apprentissage en 10 sous-échantillons, puis pour chaque valeur de λ de la grille, l'algorithme entraîne un modèle sur 9 sous-échantillons et calcule l'erreur de prédiction sur le $10^{\text{ème}}$ sous-échantillon. Cette procédure est répétée 10 fois, de manière à ce que chaque sous-échantillon soit utilisé une fois comme échantillon de validation. L'erreur de validation croisée pour chaque λ est ensuite calculée en moyennant les 10 erreurs de prédiction. Le paramètre de régularisation retenu est celui qui minimise l'erreur de validation croisée.

```
bestlam = cvfit$lambda.min  # 0.01
log(bestlam)                # -4.60517
```

La valeur de λ minimisant l'erreur de prédiction par validation croisée est 0.01. Cette valeur est faible donc, comme expliqué précédemment, les coefficients ne seront pas forcément faibles. D'un autre côté, la valeur de l'erreur quadratique moyenne est 0.09585632, cela signifie que le modèle est capable de prédire les valeurs de la variable de réponse avec une précision relativement élevée.

3.4 Question 4

```
# Estimation du modele sur tout le jeu de donnees avec le meilleur lambda
set.seed(4658)
ridge.fit.fin = glmnet(x, y, alpha=0, lambda=bestlam, thresh=1e-12)
prediction=predict(ridge.fit.fin,s=bestlam,newx=x[train==FALSE,])
```

```
class = ifelse(prediction >0.5,1,0)
mean(class!=y.test) # erreur de classification de 0.08374734 :
                    # modele a une erreur de 8,37 %
```

La fonction **ifelse**() est utilisée pour prédire la classe des observations en fonction des coefficients estimés. Les observations dont le coefficient est supérieur à 0,5 sont classées dans la classe 1 et les autres dans la classe 0. La performance du modèle est évaluée en comparant les prédictions avec les vraies valeurs de la variable de réponse (**style_en_num**) en calculant la moyenne des prédictions correctes.

Le résultat indique que le modèle de régression ridge est performant sur les données de test lorsqu'on utilise la totalité des variables. En effet, l'erreur de classification obtenue est de 8,37%, ce qui signifie que le modèle prédit correctement la classe de 91,63% des observations de test. C'est une performance satisfaisante.

4 Partie III

```
# creation du modele avec K=1
# donnees d'entrainement
model_k1_train=knn(train=xtrain, test=xtrain, cl=y.train1, k=1)
error1_train=mean(modk1_train!=y.train1) # 0
# donnees de test
model_k1_test=knn(train=xtrain,test=xtest,cl=y.train1,k=1)
error1_test=mean(modk1_test!=y.test1)  # 0.3804273
```

L'erreur de classification pour les données d'entraînement est de 0, ce qui peut indiquer que le modèle a mémorisé les données d'entraînement et est surajusté. Cela signifie que le modèle peut être trop complexe pour les données d'entraînement et ne généralisera pas bien sur de nouvelles données.

En effet, l'erreur de classification pour les données de test est de 0,3804273, ce qui signifie que le modèle a une erreur de classification de 38,04% sur les données de test. Cela peut être considéré comme un taux d'erreur élevé et indique que le modèle ne généralise pas bien aux nouvelles données.

Lorsque nous procédons pour chercher le meilleur K parmi [|1,20|], nous nous rendons compte que K=1 est le meilleur, ce qui indique que le modèle est surajusté ou qu'il y a une forte variabilité dans les données. D'autres modèles seraient probablement plus intéressants pour améliorer les performances de classification.

5 Conclusion

Finalement, nous choisissons de garder Mod2 qui a une bonne performance de généralisation que nous avons calculée précédement (cf calcul de l'aire sous la courbe ROC en partie 1). On peut à partir de ce modèle prédire le genre des musiques du fichier de musique à tester qui nous est fourni.

6 ANNEXES

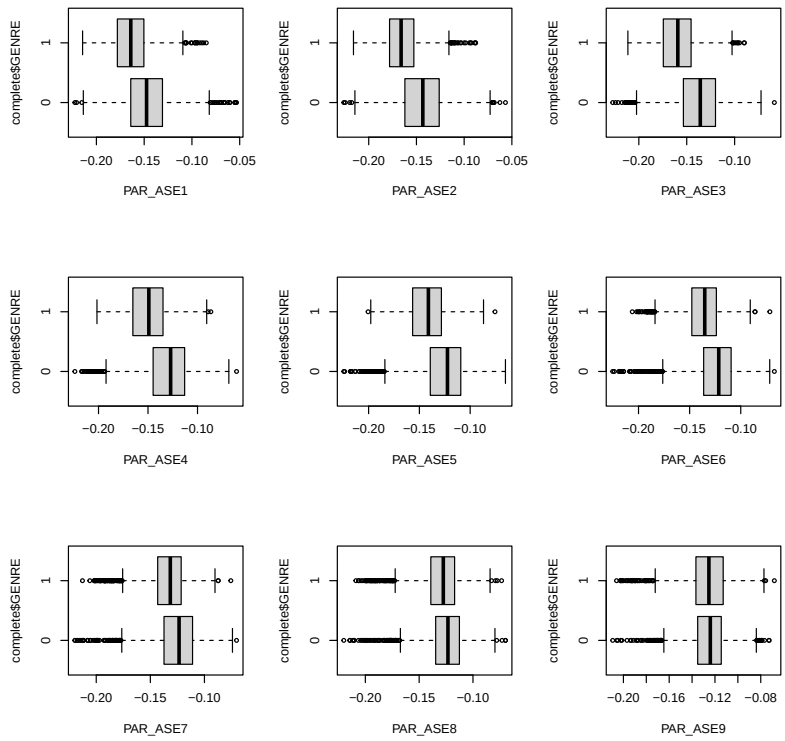


FIGURE 2 – ANNEXE 1 : Analyse descriprive : Boxplot

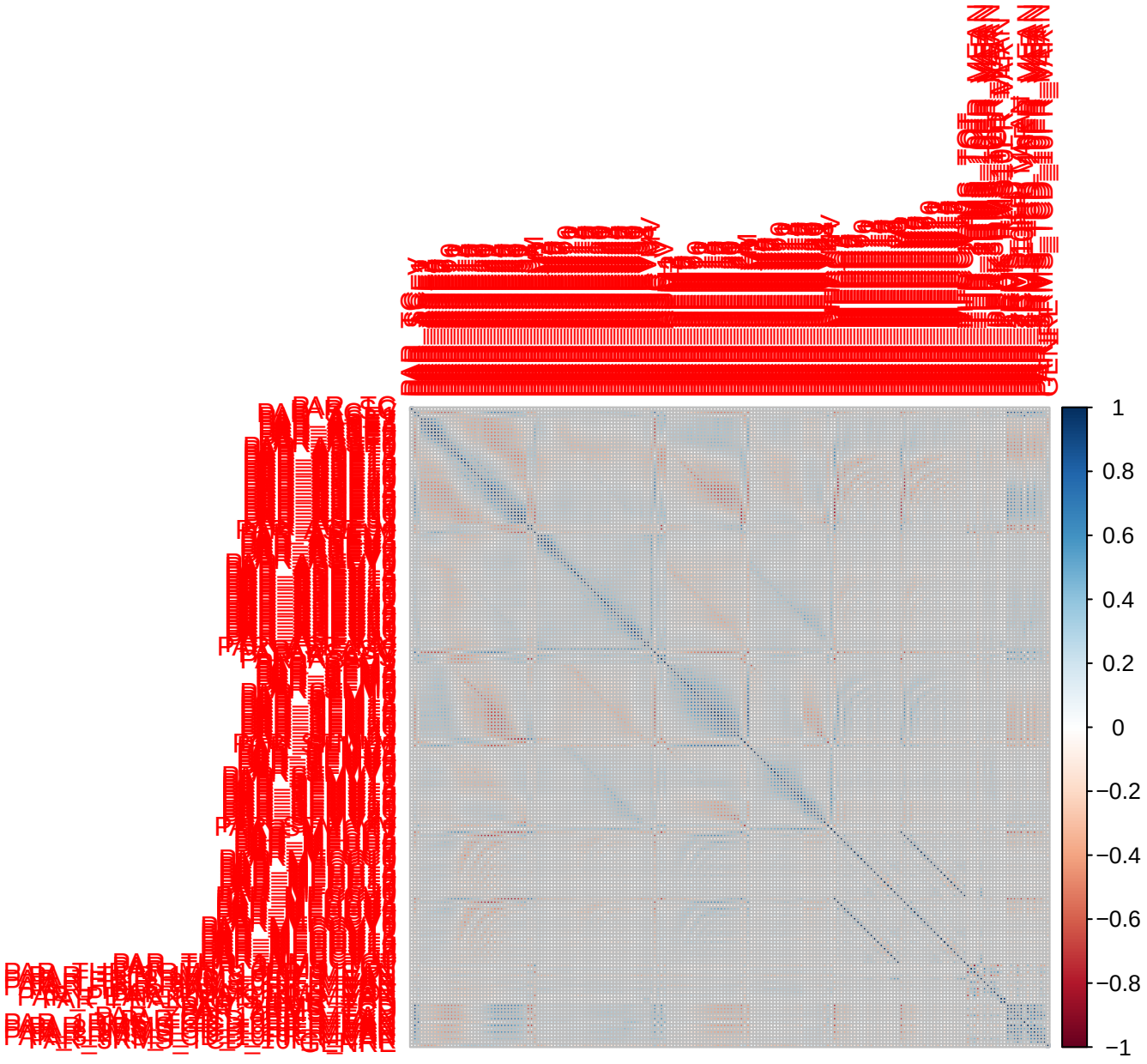


FIGURE 3 – ANNEXE 2 : Analyse descriprive : Correplot