Classification Supervisée

Julien JACQUES

26/09/2018

PLS-DA

PLS-DA

- la méthode PLS a été vu dans le cours de régression : elle consiste à créer parmi les variables explicatives X des composantes PLS qui sont les plus corrélées possibles avec la cible quantitative Y. Cette méthode est généralement notée **PLS1**
- lorsqu'on dispose de plusieurs cibles Y, ou autrement dit lorsque Y est multivariée (disons à q colonnes / dimensions), la régression **PLS2** est construire sur le même principe que **PLS1**, mais en construisant des composantes PLS à la fois dans X et dans Y (un peu comme en analyse canonique)
- la méthode **PLS-DA** est une méthode **PLS2** où la variable catégorielle Y est décomposée en $C - 1$ variable indicatrice.
  - PLS-DA est à la fois descriptive et prédictive
  - les VIP permettent d'évaluer l'importance des différentes variables

PLS-DA sous R

Il existe plusieur librairies permettant de réaliser une PLS-DA, dont :

- *DiscriMiner* : fonction *plsDA* dans laquelle la validation croisée est implémentée mais qui ne permet pas de prédire sur un nouveau jeu de données
- *mixOmics*

Nous utiliserons la librairie *mixOmics* :

```
library(mixOmics)
```

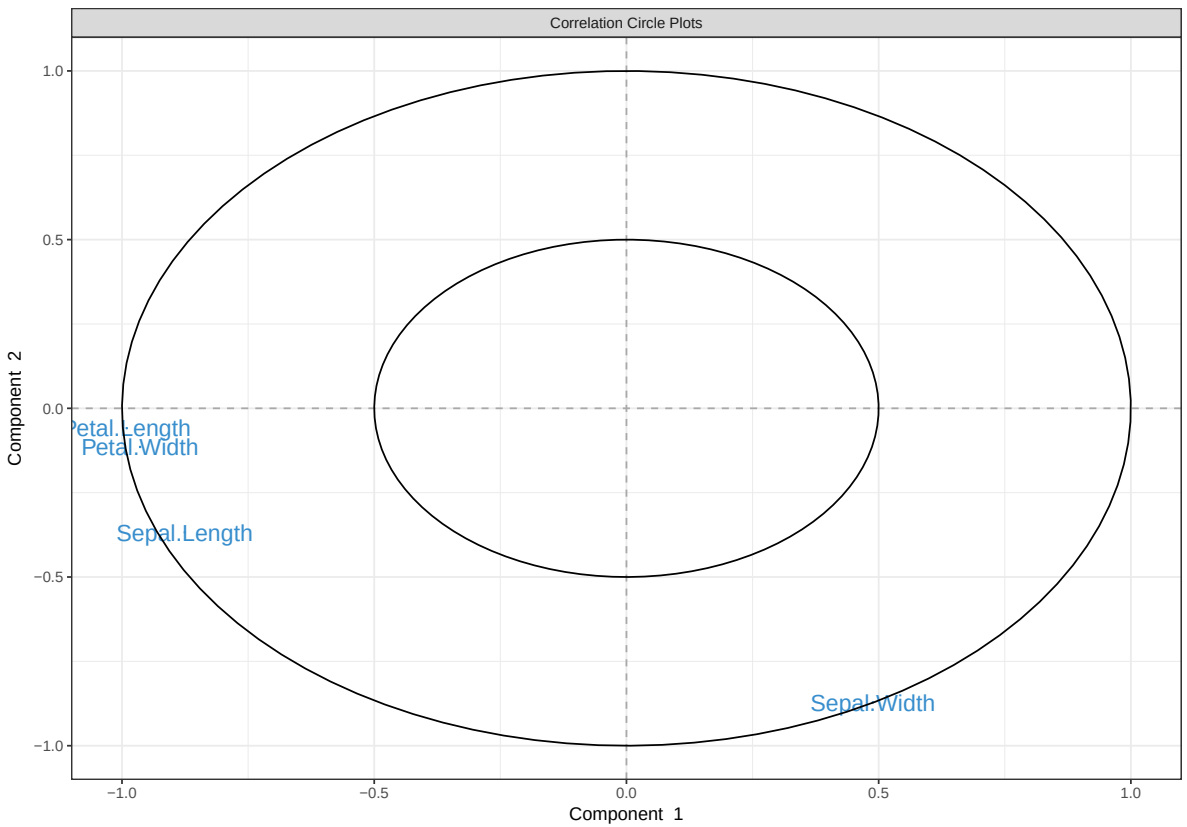
et nous allons la tester sur les données *iris* :

```
X <- iris[,1:4]
Y <- iris[,5]
```

PLS-DA sous R

Réalisons une PLS-DA avec 2 composantes (nb par défaut) et représentons les corrélations entre les composantes PLS et les variables initiales

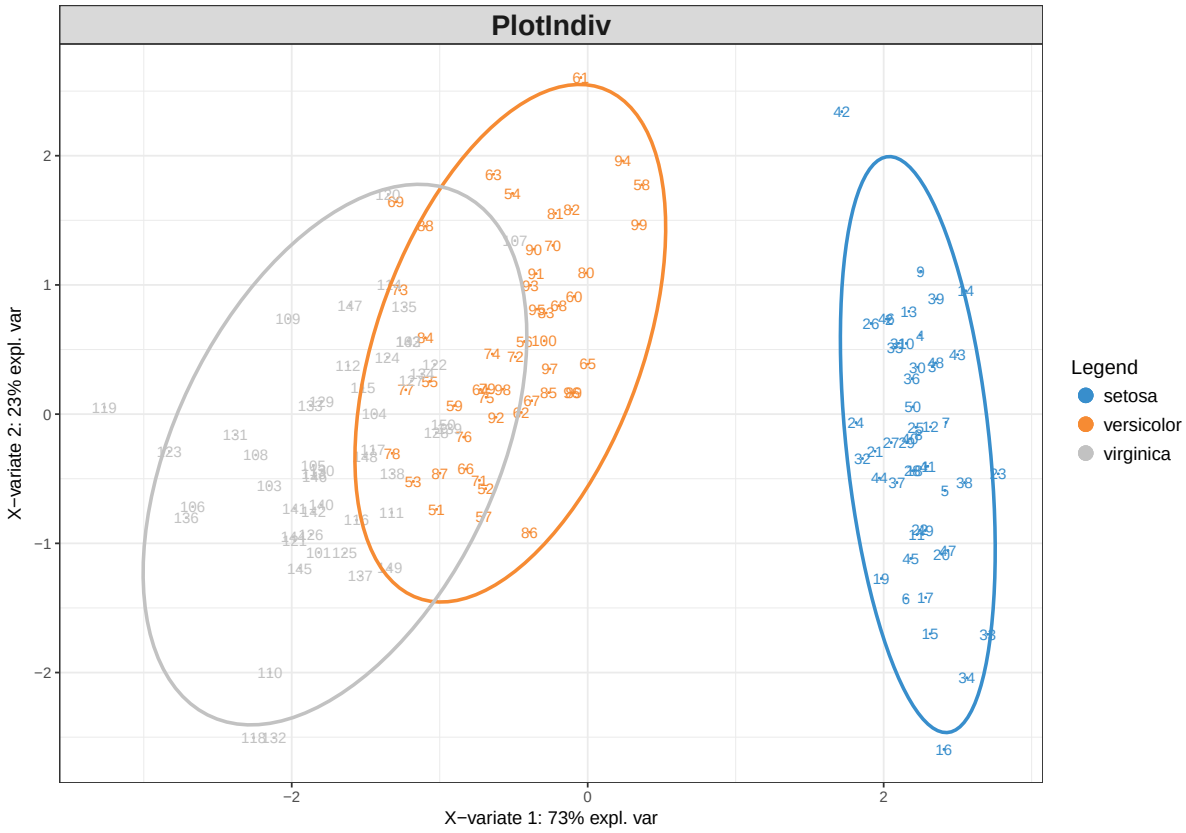
```
my_pls <- plsda(X,Y,ncomp=2)
plotVar(my_pls)
```



PLS-DA sous R

On poeut également représenter les individus dans le plan des deux premières composantes PLS

```
plotIndiv(my_pls,ind.names=T,ellipse=T,legend=T)
```



PLS-DA sous R

Evaluons la qualité de prédiction en créant un échantillon d'apprentissage et un échantillon test

```
samp <- sample(1:3, nrow(X), replace = TRUE)
test <- which(samp == 1)
train <- setdiff(1:nrow(X), test)
my_pls <- plsda(X[train,],Y[train],ncomp=2)
test.predict <- predict(my_pls,X[test,],dist = "max.dist")
Prediction <- test.predict$class$max.dist[, 2]
print(table(Y=as.character(Y[test]),Prediction))
```

```
##                Prediction
## Y               setosa versicolor virginica
##   setosa            14          0         0
##   versicolor         0         12         8
##   virginica          0          2        16
```

La qualité est assez mauvaise sur ces données connues comme *faciles*...

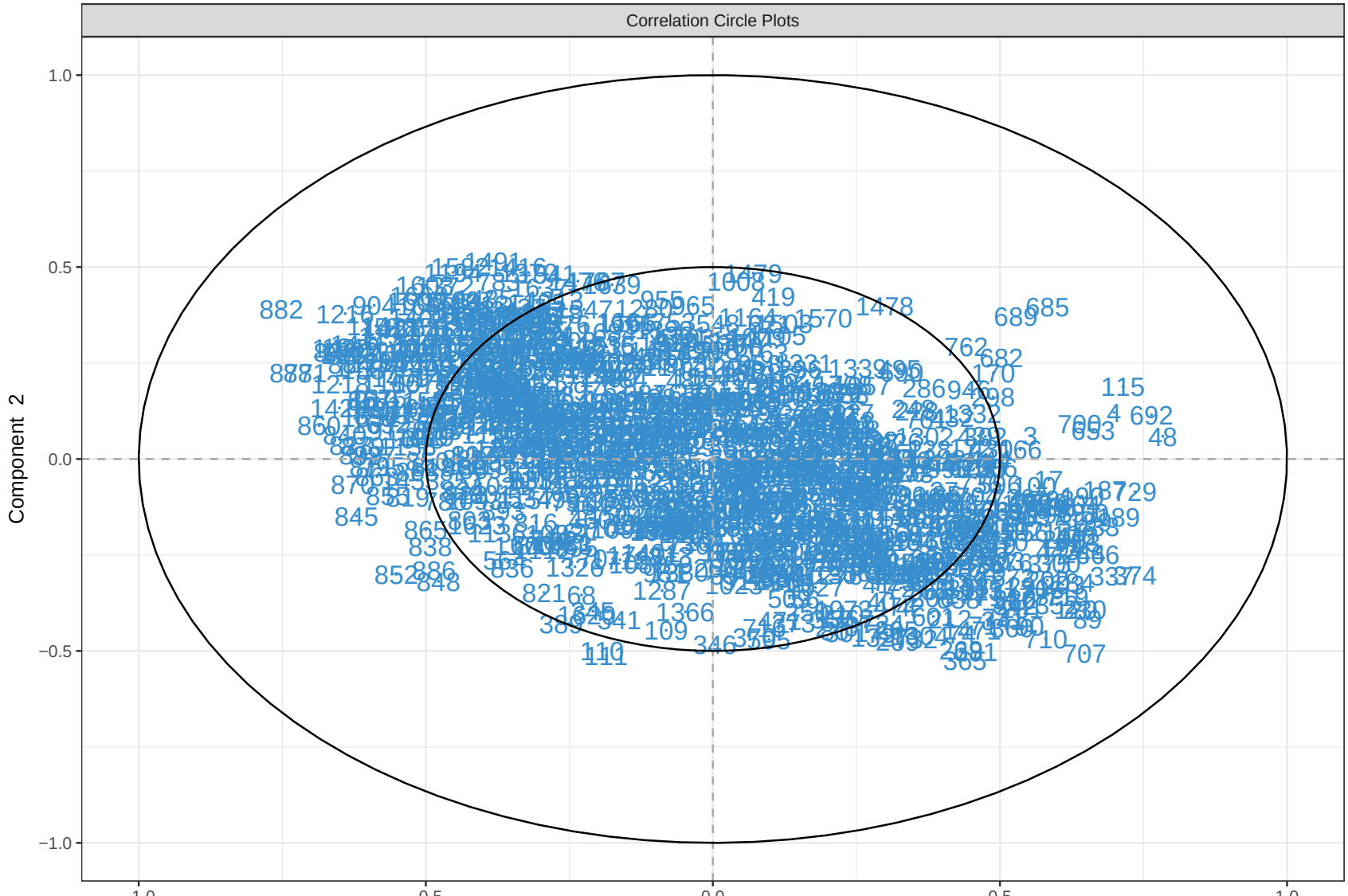
Un exemple en grande dimension

Nous allons utiliser le jeu de données breast.tumors qui contient 47 individus pour 1000 variables, avec un certain nombre de variables manquantes.

```
library(mixOmics)
data(breast.tumors)
X <- breast.tumors$gene.exp
Y <- breast.tumors$sample$treatment
```

Un exemple en grande dimension

```
my_pls <- plsda(X,Y,ncomp=2)
plotVar(my_pls)
```



Un exemple en grande dimension

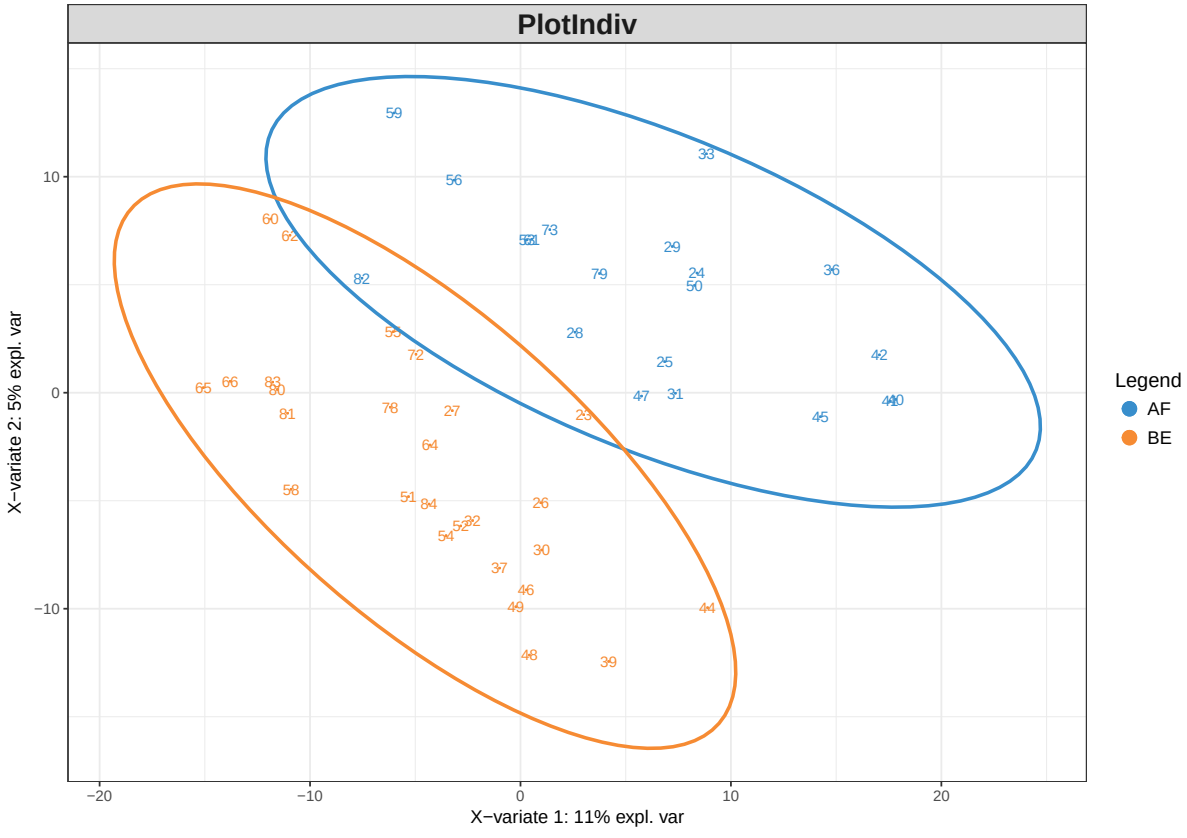
On peut calculer les VIP

```
VIP=vip(my_pls)
rownames(VIP)=breast.tumors$genes$name
print(head(sort(VIP[,1],decreasing = T)))
```

```
##      CTGF       GEM    CDK5R1       FOS      ATF3    DPYSL3
## 4.032304 3.990062 3.669865 3.507805 3.292696 3.241185
```

Un exemple en grande dimension

```
plotIndiv(my_pls,ind.names=T,ellipse=T,legend=T)
```



Les classes ont l'air super bien séparées. . .

Un exemple en grande dimension

Vu la taille d'échantillon, nous implémentons une validation croisée LOO pour évaluer la qualité de prédiction

```
bonclass=0
pred=NULL
nbcomposantes=2
for (i in 1:nrow(X)){
  my_pls <- plsda(X[-i,],Y[-i],ncomp=nbcomposantes)
  tmp <- predict(my_pls,X[i,,drop=F],dist = "max.dist")
  pred=c(pred,tmp$class$max.dist[,nbcomposantes])
}
cat('Taux de bon classement : ',mean(pred==Y))
```

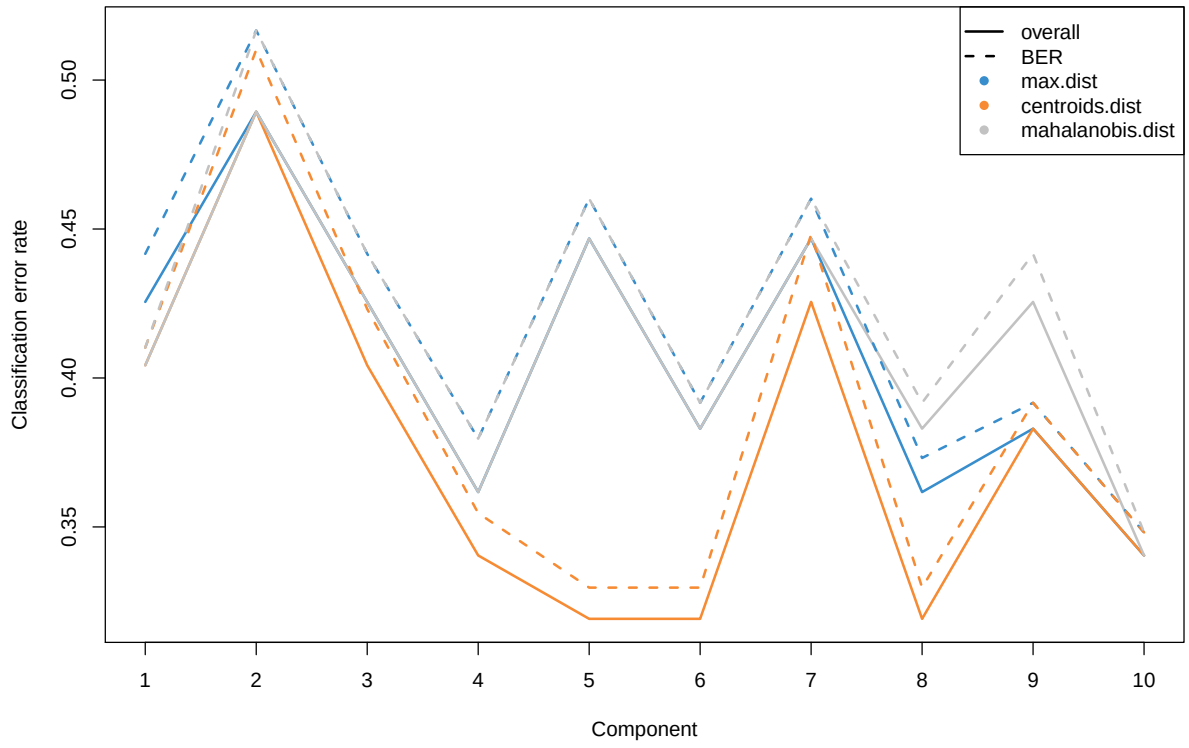
```
## Taux de bon classement :  0.5744681
```

Le taux de bon classement est faible par rapport à ce qu'on aurait pu s'attendre en regardant la séparation des classes : mais attention cette séparation des classes étaient sur les données d'apprentissage; La PLS-DA a tendance ici a faire du sur-apprentissage

Un exemple en grande dimension

On peut tester un nombre de composantes plus important . . .

```
my_pls <- plsda(X,Y,ncomp=10)
plot(perf(my_pls,progressBar =F))
```



Les différences entre les modèles ne sont pas flagrantes. . .