Mapping the HDL proteome in Metabolic Syndrome through label-free quantification.

Elvira Márquez Paradas

2023-06-05

We will generate a sample classification model using the sPLS-DA (sparse partial least squares discriminant analysis) method, which finds discriminant relationships between groups in the multivariate data set, also employing a feature selection approach to identify the most relevant subset of proteins in group differentiation.

```
library(mixOmics)
```

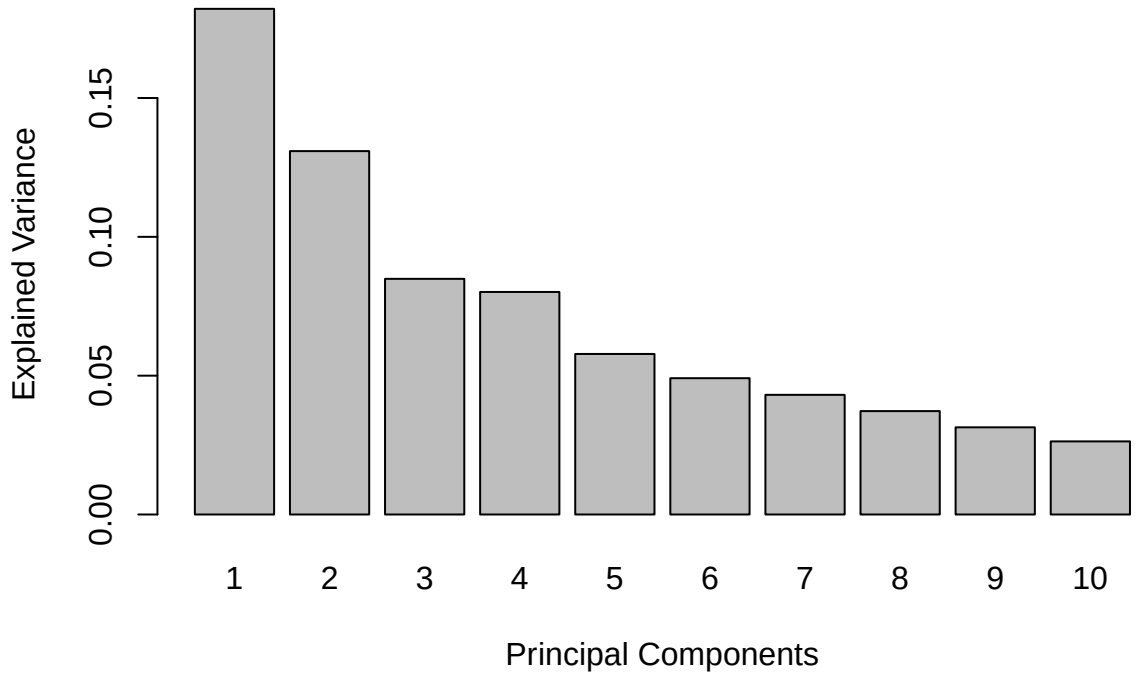
```
## Loading required package: MASS

## Warning: package 'MASS' was built under R version 4.2.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.2.3

##
## Loaded mixOmics 6.22.0
## Thank you for using mixOmics!
## Tutorials: http://mixomics.org
## Bookdown vignette: https://mixomicsteam.github.io/Bookdown
## Questions, issues: Follow the prompts at http://mixomics.org/contact-us
## Cite us:  citation('mixOmics')
```

```
set.seed(5249)

# Creation of the Y variable containing the patient groups.
Y=as.factor(c(rep("H", 17), rep("MetS", 19)))

# Creation of the X variable, contaning the genes with their expression values
genes=read.table("./data/preprocessed_data_cero_imputed.txt", header = T, sep = "\t", na.strings = "NaN
genes=t(genes)
gene_names=genes[1,]
gene_names <- gsub(";.*|_.*", "", gene_names)    # Shorten gene names to improve graphics
gene_names[62]= "IGKV3D-20"   # Modify to avoid having repeated gene names
gene_names[43]= "APOM_APOM"
X=genes[-1,]
colnames(X)=gene_names
X <- apply(X, c(1, 2), as.numeric) #Convert matrix values to numerical values
```

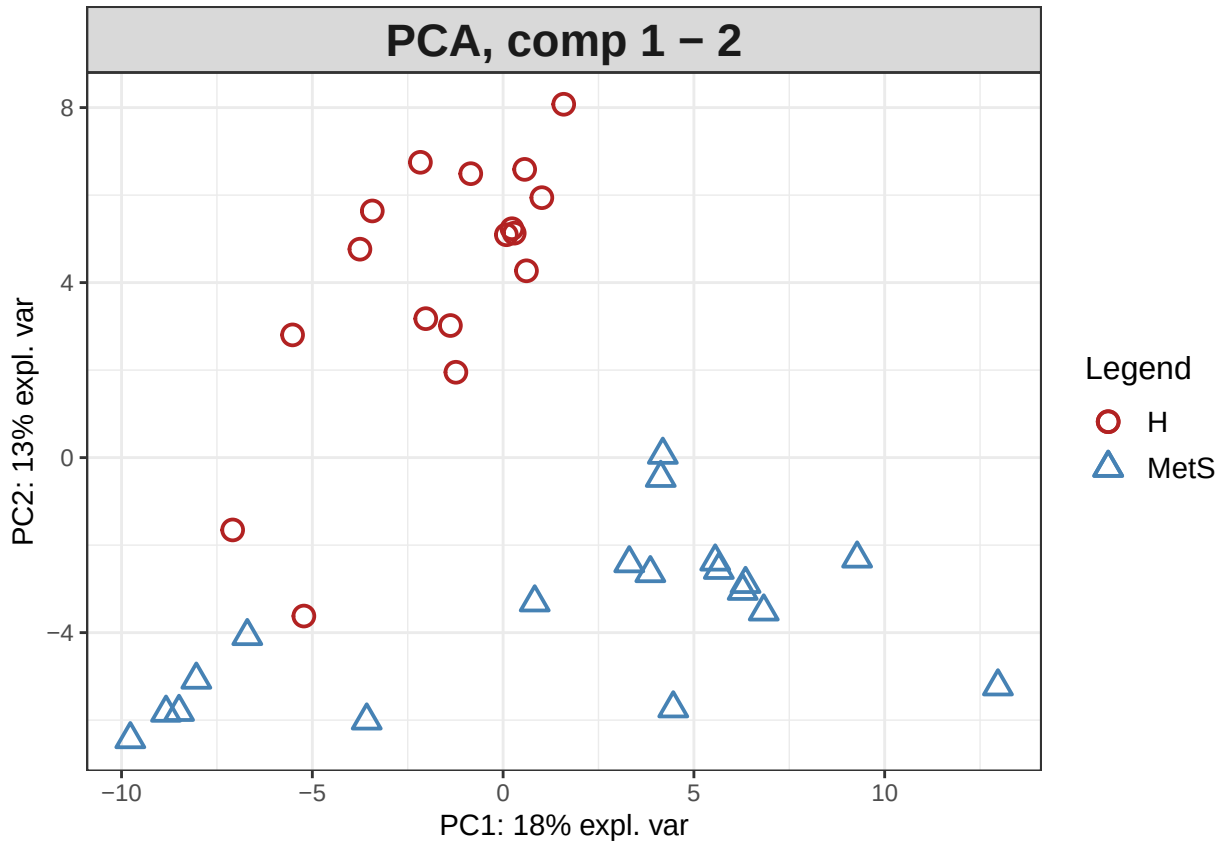
PRINCIPAL COMPONENT ANALYSIS (PCA)

We visualise the behaviour of the samples by dimensionality reduction analysis

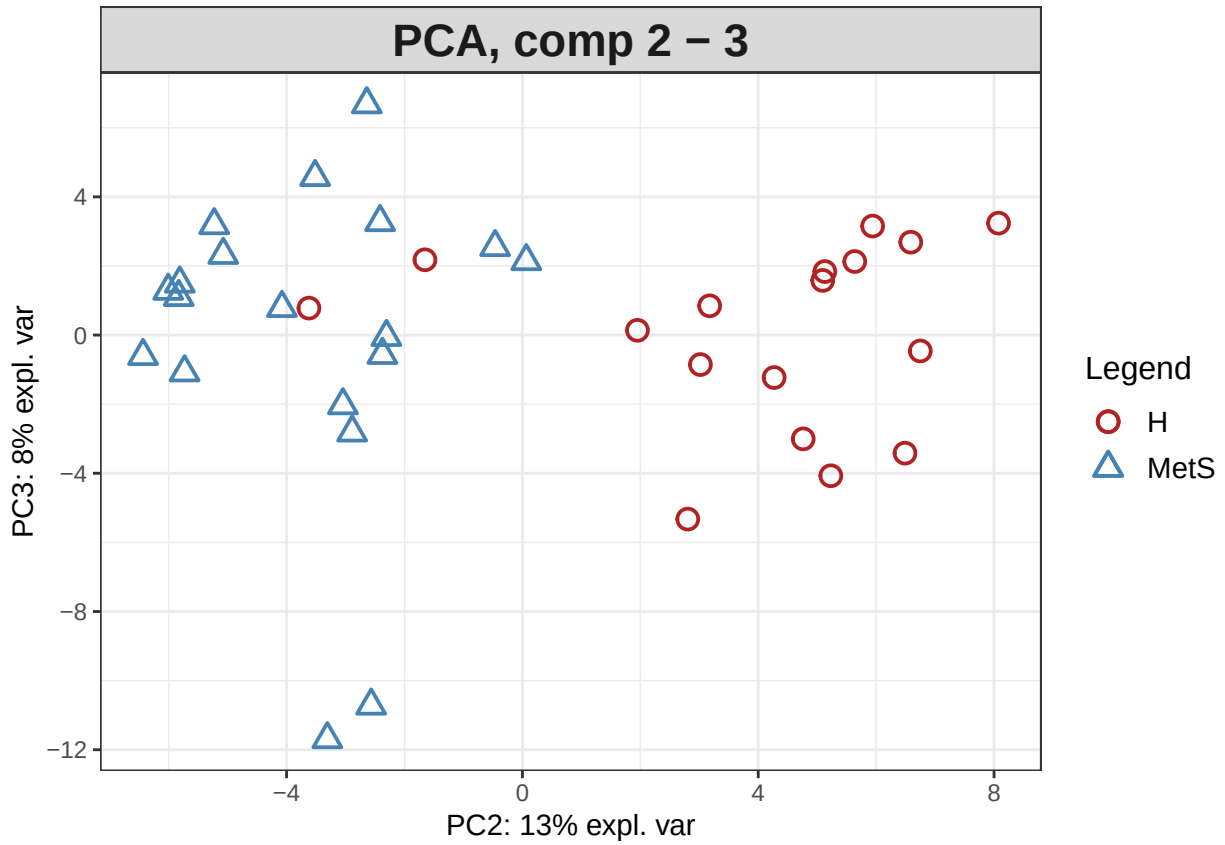
```
pca = pca(X, ncomp = 10, center = F, scale = F)

# Number of components
######
plot(pca)
```




```
# Component 1-2
#####
plotIndiv(pca, comp=c(1,2), group = Y, ind.names = FALSE,
          legend = TRUE, title = 'PCA, comp 1 - 2',col.per.group = c("firebrick","steelblue"))
```




```
# Component 2-3
#####
plotIndiv(pca, comp=c(2,3), group = Y, ind.names = FALSE,
          legend = TRUE, title = 'PCA, comp 2 - 3',col.per.group = c("firebrick","steelblue"),)
```

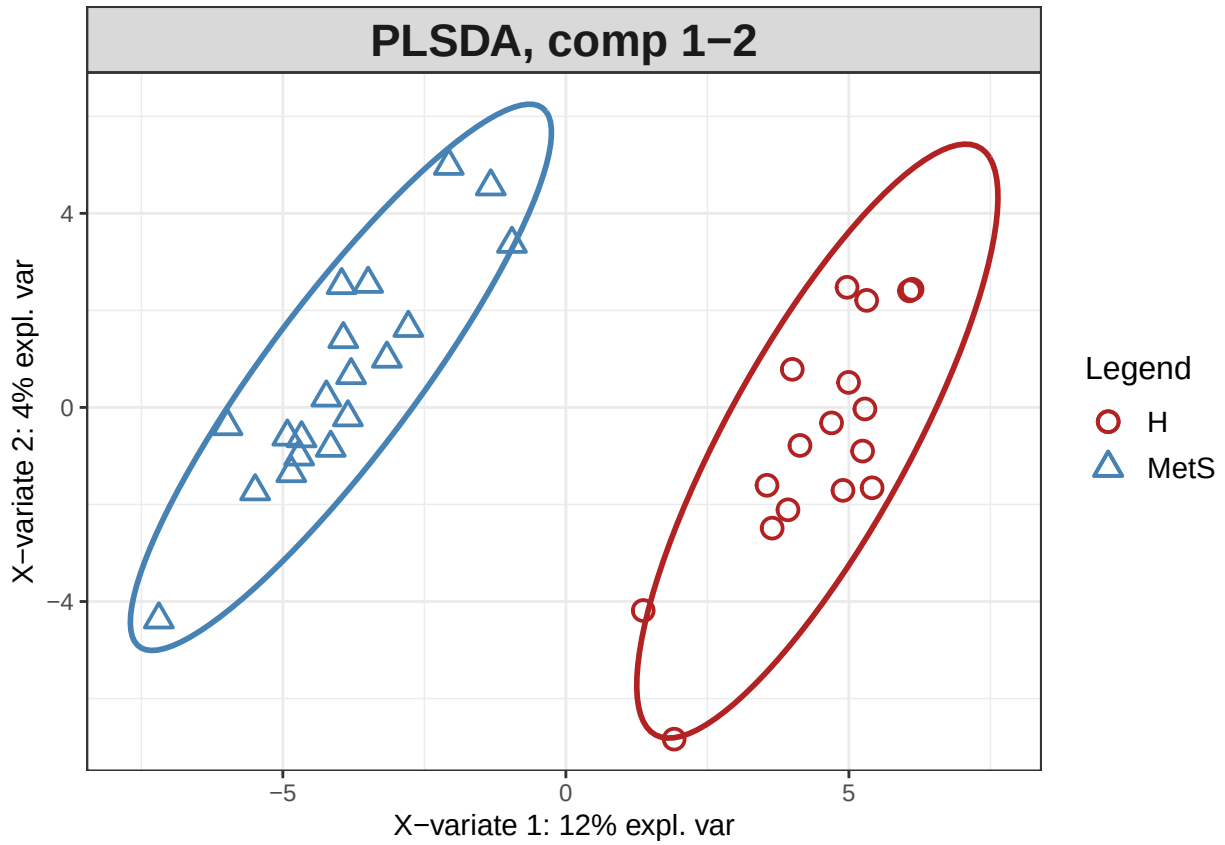


The first 3 components explain 39% of the variance. The main component responsible for explaining the difference between groups is component 2.

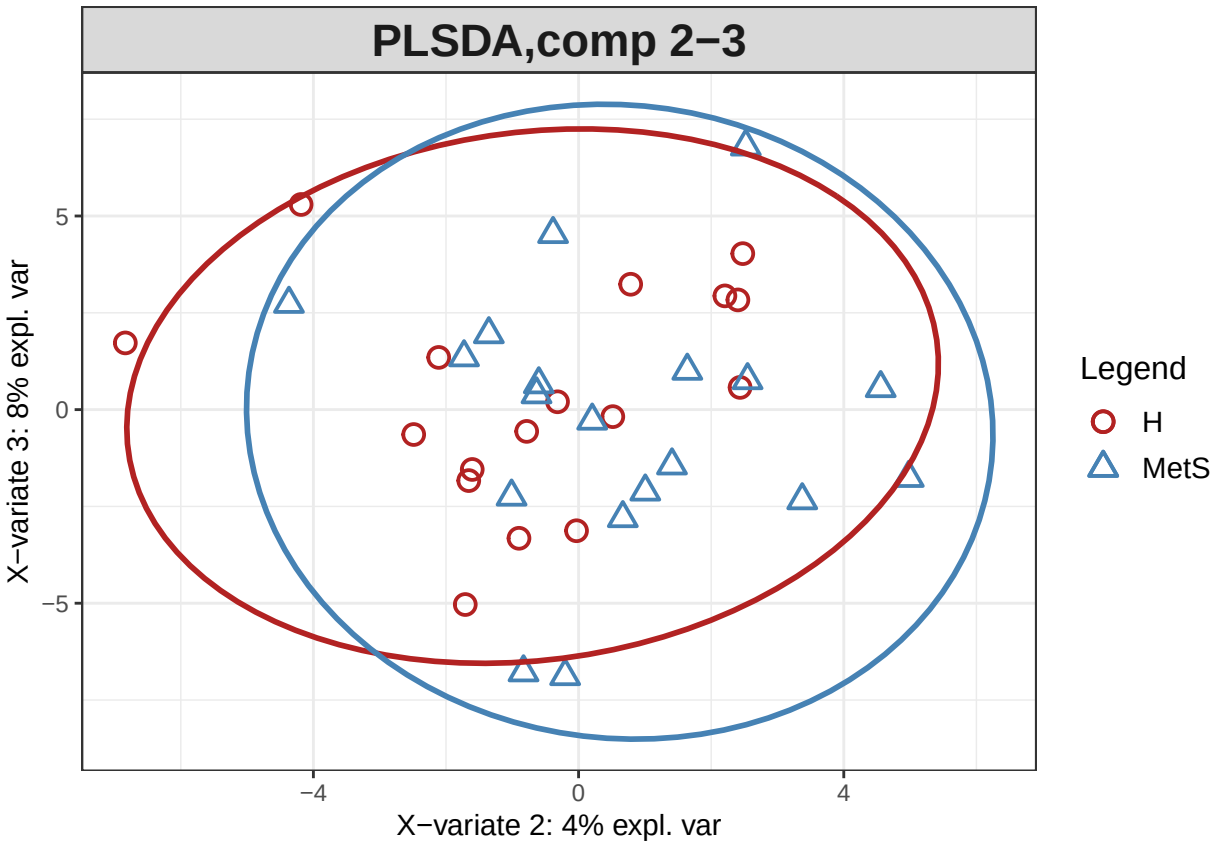
INITIAL PLS-DA MODEL

In PLS-DA, we seek to find a linear combination of predictor variables (X) that is highly correlated with the class variables (Y), thus maximizing the separation between classes. First of all, we will generate a initial PLS-DA.

```
splsda <- splsda(X, Y, ncomp = 10)

plotIndiv(splsda , comp = 1:2,
          group = Y, ind.names = FALSE,
          col.per.group = c("firebrick","steelblue"),
          ellipse = TRUE,
          legend = TRUE, title = 'PLSDA, comp 1-2',
          legend.labels = legend_labels)
```




```
plotIndiv(splsda , comp = 2:3,
          group = Y, ind.names = FALSE,
          ellipse = TRUE,
          col.per.group = c("firebrick","steelblue"),
          legend = TRUE, title = 'PLSDA,comp 2-3',
          legend.labels = legend_labels,
          cex.main=0.8)
```

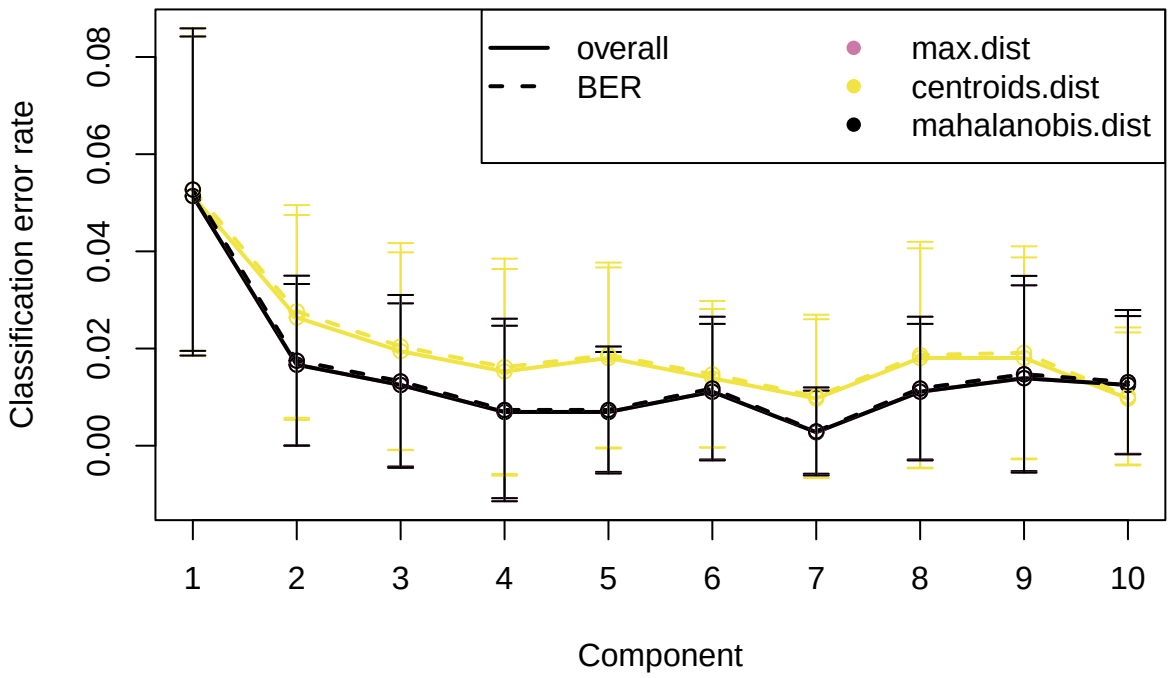


The first 3 components explain 24% of the variance of the data. Component 1 is responsible for group separation.

TUNING sPLS-DA

Next, we will validate the performance of the previously created model using the M-Fold cross-validation method and select the optimal number of components and variables.

```
# Evaluation of the performance of the model created
#####
perf.splsda <- perf(splsda, validation = "Mfold",
                            folds = 5, nrepeat = 20,
                            progressBar = FALSE, auc = TRUE)

plot(perf.splsda, col = color.mixo(5:7), sd = TRUE,
     legend.position = "horizontal")
```




```
# From this, it seems 7 components are the optimal.
#####
perf.splsda$choice.ncomp
```

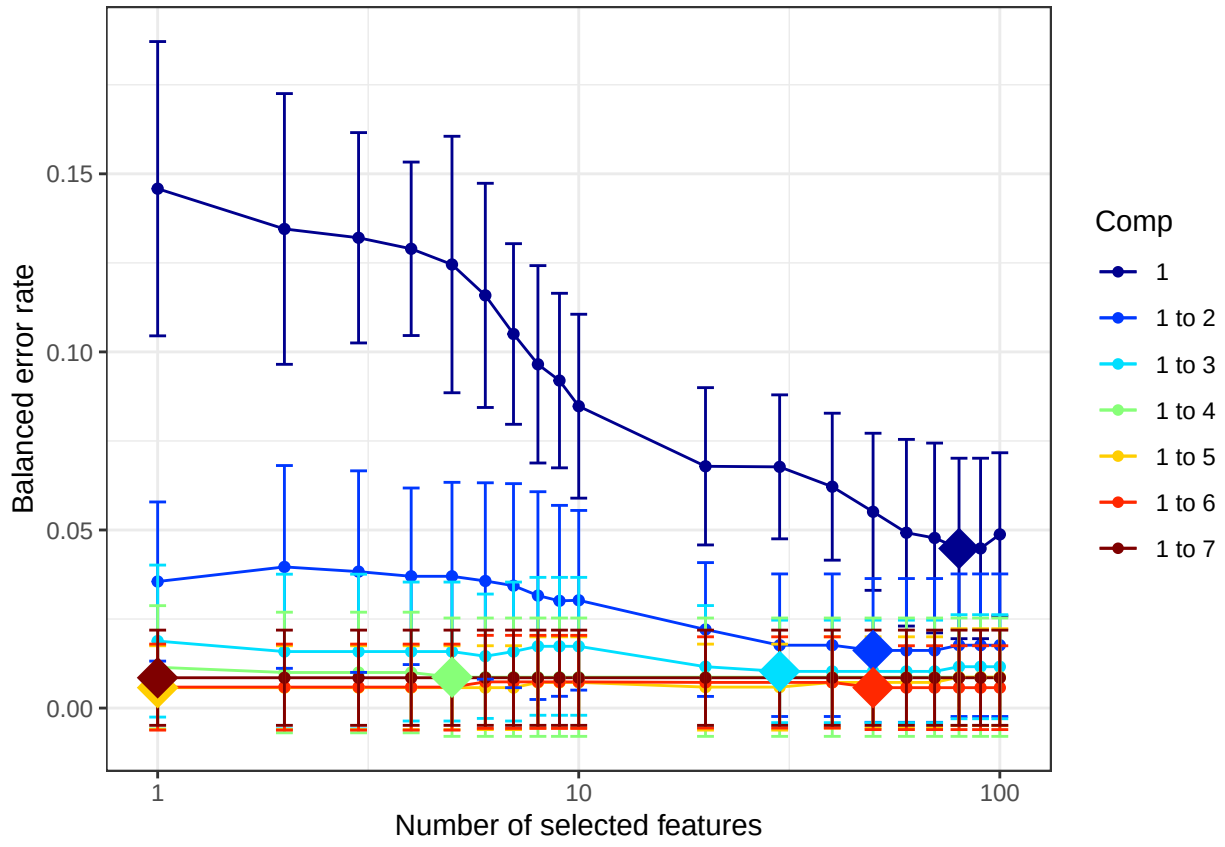
```
##         max.dist centroids.dist mahalanobis.dist
## overall        7              7                7
## BER            7              7                7
```

```
# Selecting the number of variables
#####
list.keepX <- c(1:10,  seq(20, 100, 10))
list.keepX
```

```
##  [1]   1   2   3   4   5   6   7   8   9  10  20  30  40  50  60  70  80  90 100
```

```
# Undergo the tuning process to determine the optimal number of variables
#####
tune.splsda <- tune.splsda(X, Y, ncomp = 7,
                                 validation = 'Mfold',
                                 folds = 5, nrepeat = 20,
                                 dist = "max.dist",
                                 measure = "BER",
                                 test.keepX = list.keepX,
                                 cpus = 2)
```

```
plot(tune.splsda, col = color.jet(7)) # Plot output of variable number tuning
```




```
tune.splsda$choice.ncomp$ncomp # First 2 components selected
```

```
## [1] 2
```

```
# In summary:
#####
optimal.ncomp <- tune.splsda$choice.ncomp$ncomp
optimal.ncomp # First 2 components selected
```

```
## [1] 2
```

```
optimal.keepX <- tune.splsda$choice.keepX[1:optimal.ncomp]
optimal.keepX # First component with 80 proteins and second one with 50 proteins
```

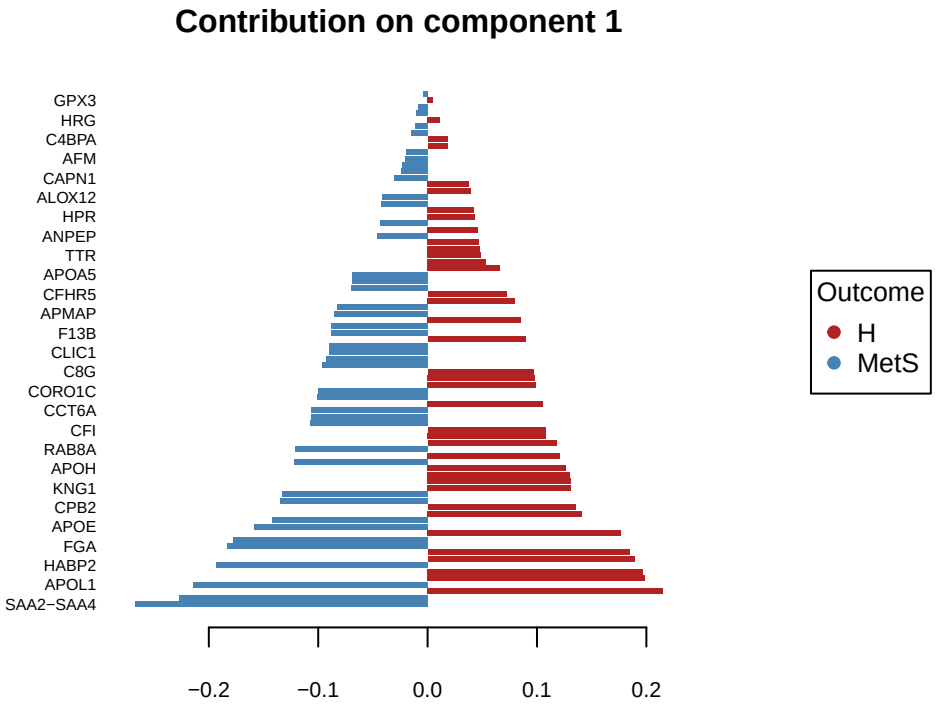
```
## comp1 comp2
##    80    50
```

The optimal model will have two components with 80 and 50 variables, respectively.

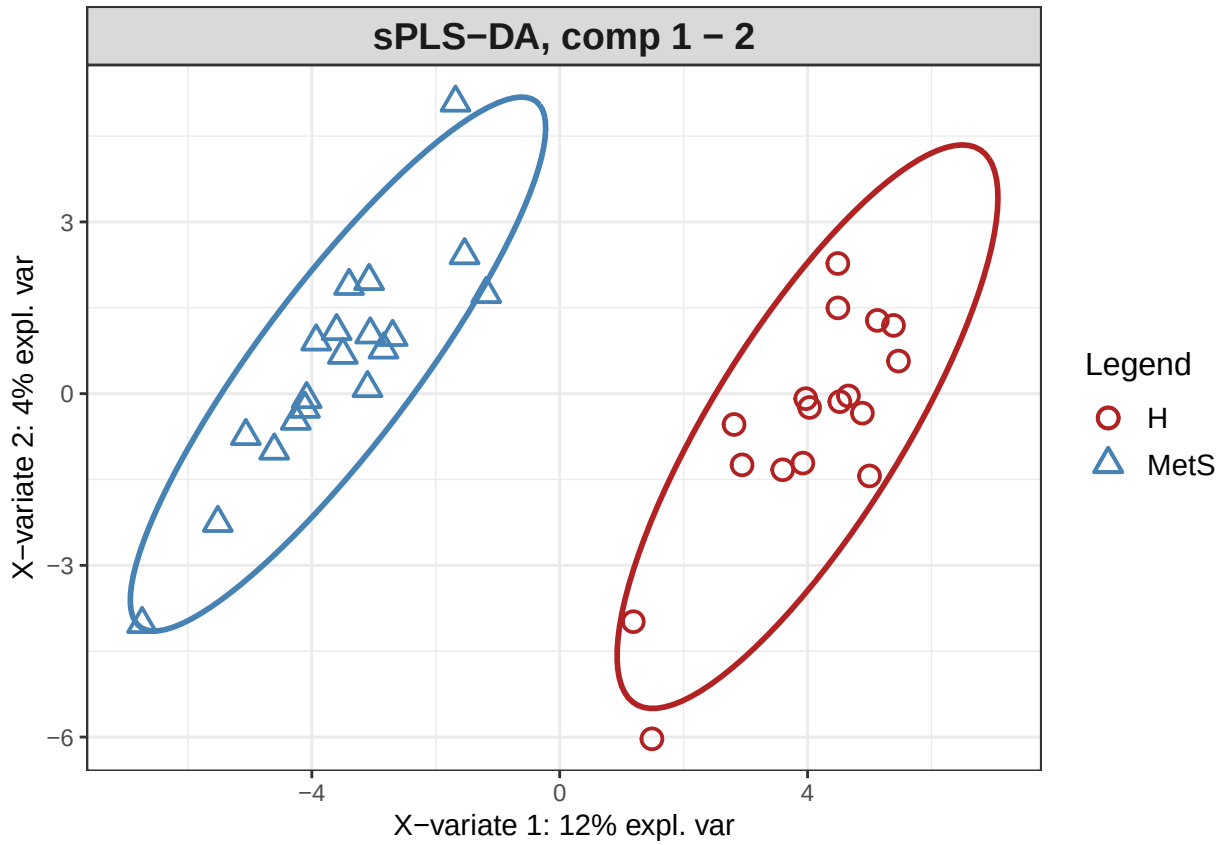
FINAL sPLS-DA

Finally, we will create the final model with the optimal number of components and variables variables selected in the training and validation process.

```
# Model generation
#####
final.splsda <- splsda(X, Y,
                       ncomp = optimal.ncomp,
                       keepX = optimal.keepX)

# Visualization of the proteins most involved in group separation
#####
plotLoadings(final.splsda, comp = 1, method = 'mean', contrib = 'max',
             legend.color = c("firebrick","steelblue"),title="Contribution on component 1",
             size.legend=0.8, size.title=1,size.name = 0.5)
```




```
plotIndiv(final.splsda, comp = c(1,2),
          group = Y, ind.names = FALSE,
          ellipse = TRUE, legend = TRUE,
          col.per.group = c("firebrick","steelblue"),
          title = 'sPLS-DA, comp 1 - 2',size.title = 14)
```



Next, we extract the variables or proteins selected for each component:

```
comp = 1
comp1= which(final.splsda$loadings$X[, comp] != 0)
comp1=as.data.frame(comp1)
comp1=rownames(comp1)
comp1
```

```
##  [1] "EHD1"        "TTR"         "GPX3"        "SAA2-SAA4"   "ILK"
##  [6] "IGLL5"       "C2"          "APOA5"       "SERPING1"    "HSPD1"
## [11] "CFB"         "CFI"         "HSPA8"       "KLKB1"       "APOC4-APOC2"
## [16] "APOC1"       "CLIC1"       "APOL1"       "FCN3"        "LDHA"
## [21] "F13A1"       "F2"          "HP"          "HPR"         "F12"
## [26] "A2M"         "C3"          "KNG1"        "IGHV3-21"    "APOE"
## [31] "APOA2"       "FGA"         "FGG"         "APCS"        "APOH"
## [36] "RBP4"        "AMBP"        "AHSG"        "C4BPA"       "VTN"
## [41] "APOB"        "LCAT"        "HRG"         "KRT6B"       "SERPINA5"
## [46] "F13B"        "SERPIND1"    "IGKV4-1"     "APOA4"       "ENO1"
## [51] "C8A"         "C8B"         "C8G"         "CAPN1"       "HSP90AB1"
## [56] "CFH"         "SAA1"        "C7"          "HSPA5"       "PYGB"
## [61] "ANPEP"       "ALOX12"      "ITIH2"       "ITIH1"       "SERPINA4"
## [66] "CCT6A"       "AFM"         "LIMS1"       "RAB8A"       "ACTR3"
## [71] "HABP2"       "ITIH4"       "APOM"        "UNC13D"      "CPB2"
## [76] "CFHR5"       "PARVB"       "APMAP"       "EHD3"        "CORO1C"
```

```
comp = 2
comp2=which(final.splsda$loadings$X[, comp] != 0)
comp2=as.data.frame(comp2)
comp2=rownames(comp2)
comp2
```

```
##  [1] "GPX3"      "IGHG2"     "C1R"       "APOA5"     "AGT"       "APOD"
##  [7] "SERPINA10" "KLKB1"     "CP"        "F2"        "HP"        "HPR"
## [13] "SERPINA1"  "SERPINA3"  "C5"        "KNG1"      ""          "IGHV3-21"
## [19] "C1QC"      "HRG"       "A1BG"      "GAPDH"     "SERPINA5"  "APOA4"
## [25] "CFH"       "C1S"       "C4A"       "CLU"       "C6"        "PKM"
## [31] "ALOX12"    "VCL"       "MSN"       "SERPINF1"  "CCT6A"     "PLTP"
## [37] "RAB8A"     "RAP1B"     "TUBA1B"    "APOF"      "PON3"      "APOM"
## [43] "UNC13D"    "CPB2"      "CNDP1"     "PGLYRP2"   "CFHR5"     "PARVB"
## [49] "SAR1A"     "EHD3"
```

PREDICTION

Before evaluating the predictive capability of the model, we will segment the PLS-DA(s)data into training and test data sets. First, we will train the model with the training dataset, and then evaluate its predictive capability with the test dataset.

```
train <- sample(1:nrow(X), 30) # randomly select 30 samples in training
test <- setdiff(1:nrow(X), train)

# Store matrices into training and test set
#####
X.train <- X[train, ]
X.test <- X[test,]
Y.train <- Y[train]
Y.test <- Y[test]

# Train the model on the training data
#####
train.splsda.srbct <- splsda(X.train, Y.train, ncomp = optimal.ncomp, keepX = optimal.keepX)

# The model is then applied on the test set
#####
predict.splsda.srbct <- predict(train.splsda.srbct, X.test,
                                dist = "max.dist")

# Evaluate the prediction accuracy for the first two components
#####
predict.comp2 <- predict.splsda.srbct$class$max.dist[,2]
table(factor(predict.comp2, levels = levels(Y)), Y.test)
```

```
##       Y.test
##        H MetS
##   H    1    0
##   MetS 0    5
```

MODEL EVALUATION

We will use the calculation of the area under the curve to evaluate the accuracy of the model.

```
pdf("AUC.pdf")
auc.splsda = auroc(final.splsda, roc.comp = 1, print = TRUE)
```

```
## $Comp1
##           AUC   p-value
## H vs MetS   1 3.095e-07
##
## $Comp2
##           AUC   p-value
## H vs MetS   1 3.095e-07
```

```
auc.splsda = auroc(final.splsda, roc.comp = 2, print = TRUE)
```

```
## $Comp1
##           AUC   p-value
## H vs MetS   1 3.095e-07
##
## $Comp2
##           AUC   p-value
## H vs MetS   1 3.095e-07
```

```
dev.off()
```

```
## pdf
##   2
```

The area under the cyrva obtained for both components is 1, which indicates the presence of overfitting. This implies that the model generated is only valid for the data with which we have created it, and that it will not be able to generalize to other data sets. A larger number of samples will probably be needed to improve the accuracy and quality of the model.