Evaluation de la qualité du prétraitement avec PepsNMR - Human Serum dataset

B. Govaerts & M. Martin

May 18, 2018,14:16

Contents

Etude de la répétabilité spectrale d'un prétraitement

```
#============ PARAMETRES A MODIFIER quand vous utilisez MBXUCL
## A. Paramètres globaux liés aux noms et chemin d'accès des dataset

dataname <- "HumanSerum" # nom du jeu de données

## Mettre ici le nom du chemin d'accès du répertoire où sont les spectres prétraités
# spectres prétraités
data.path <- "/Users/manon/Documents/Conférences/RFMF2018/Atelier_PepsNMR/RMD"

# load le fichier .RData créé précédemment
load(file.path(data.path,paste0(dataname,".RData")))

dataset <- Re_Spectrum_data
# ou
# dataset <- read.table(file = file.path(data.path,paste0(dataname,"_Spectra.csv")),
#                       sep=";", dec=" ", row.names = 1, header = TRUE,
#                       check.names =FALSE)

dataset <- as.matrix(dataset)


## B. Définition des groupes de spectres par sujet.

# group_HS est loadé avec le fichier .RData créé précédemment
```

```
# ou
# dataGroup.path <-  system.file("extdata", package = "PepsNMR") # données du package PepsNMR
# dir(dataGroup.path) # ce que contient le répertoire data.path
# group_HS <- read.csv(file.path(dataGroup.path, "Group_HS.csv"), header = TRUE, sep = ";")

group <- as.factor(substr(group_HS[,1],4,5)) # donneur comme groupe
table(group) # fréquences de chaque groupe
```

```
## group
## D1 D2 D3 D4
##  8  8  8  8
```

```
group_num <- group_HS[,2]  # Version numerique des noms de groupes
table(group_num) # fréquences de chaque groupe
```

```
## group_num
## 1 2 3 4
## 8 8 8 8
```

```
ls() # éléments dans l'environnement global
```

```
## [1] "data.path"        "dataname"         "dataset"
## [4] "Fid_info"         "group"            "group_HS"
## [7] "group_num"        "Re_Spectrum_data"
```

```
## C. Setup des paramètres de représentation graphique des paramètres de prétraitement
num.stacked <- 2 # nombre de graphes par sortie graphique


## D. paramètres des méthodes incluses dans le MIC à modifier
ncompPCA = 4 # nombre de composantes en PCA
nClust = 4 # nombre de clusters pour le clustering
nLVPLSDA = 4 # nombre de variables latentes en PLS-DA
```

Inertia

```
# INERTIA
Inertia.res = MBXUCL::Inertia(x = dataset, y = group_num, print = FALSE)
colnames(Inertia.res[["Between_within"]]) <- c("Inertia Between groups","Inertia Within groups", "Total
pander(Inertia.res[["Between_within"]])
```

	Inertia Between groups	Inertia Within groups	Total inertia
Value	6457	711.9	7169
Percentage	90.07	9.93	100

```
pander(Inertia.res[["Per_group"]])
```

	N	Inertia_group	Inertia_group100	Inertia_moy_group
Group 1	8	169.8	23.85	21.22
Group 2	8	256.2	36	32.03
Group 3	8	91.47	12.85	11.43
Group 4	8	194.4	27.3	24.3

	N	Inertia_group	Inertia_group100	Inertia_moy_group
Total	32	711.9	100	NA

PCA

```
# PCA
PCA.res = MBXUCL::SVDforPCA(dataset, ncomp=ncompPCA)
```

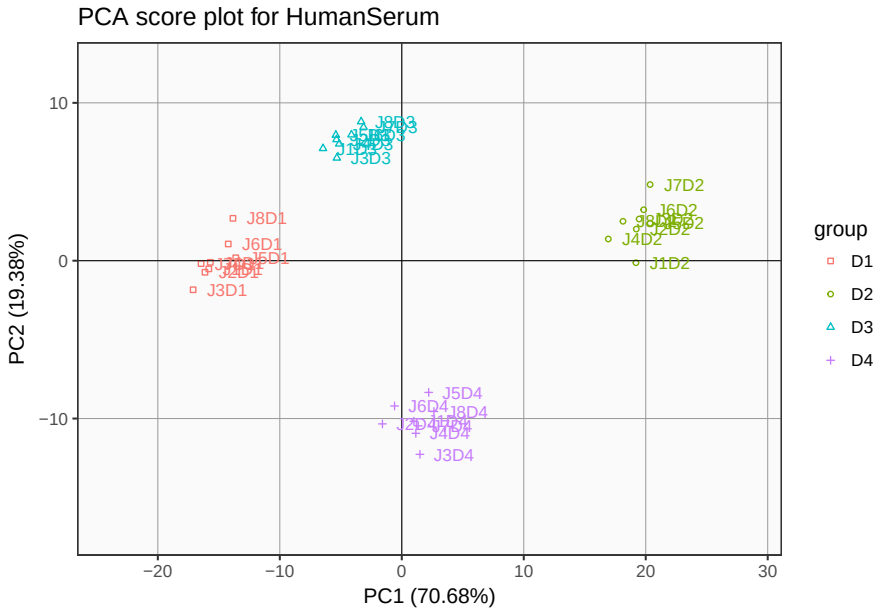
Eigenvalues:

```
eigval_mat <- rbind(PCA.res$eigval, PCA.res$var, PCA.res$cumvar)
rownames(eigval_mat) <- c("eigen values", "percentage of variance", "cumulated percentage of variance")
eigval_mat <- round(eigval_mat,2)

pander(eigval_mat[,1:4], caption = "valeurs propres PCA pour les 4 premières CP")
```

Table 3: valeurs propres PCA pour les 4 premières CP

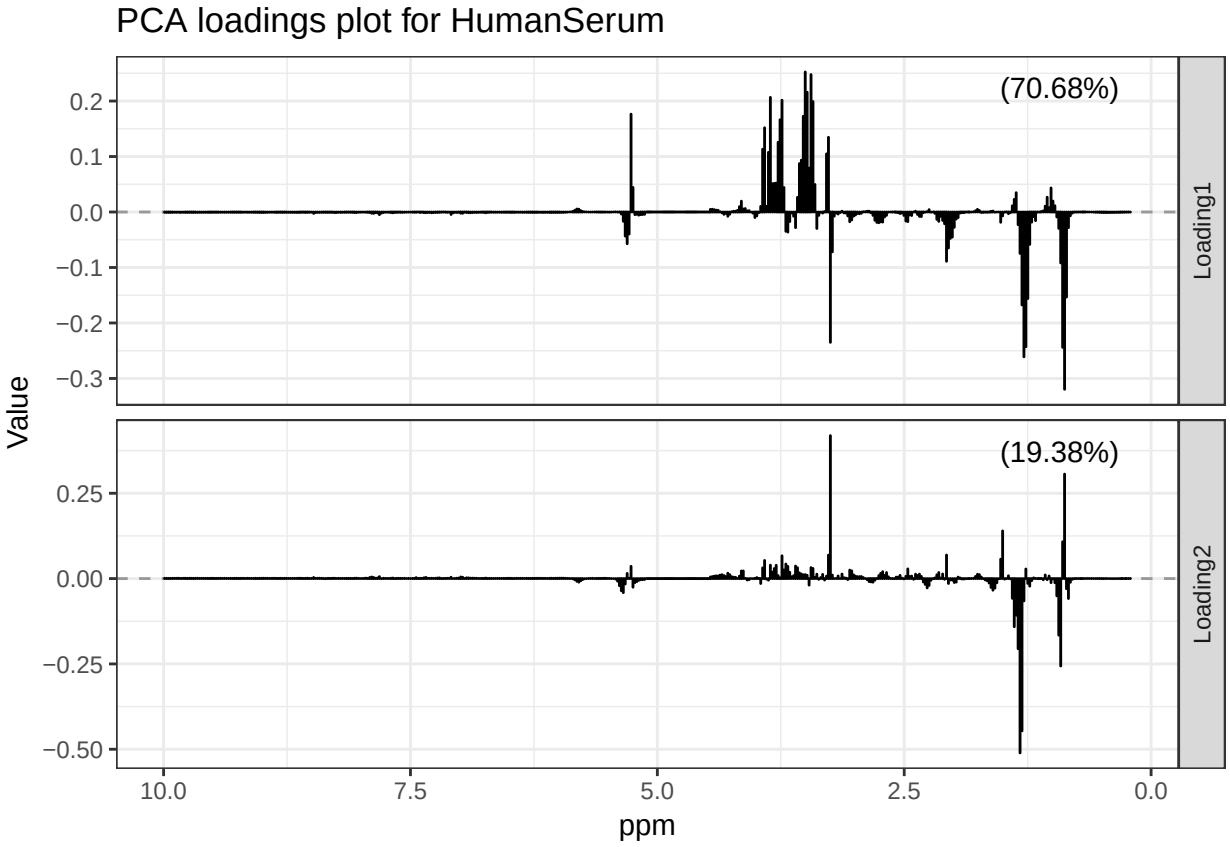
	PC1	PC2	PC3	PC4
eigen values	5067	1389	330.4	93.69
percentage of variance	70.68	19.38	4.61	1.31
cumulated percentage of variance	70.68	90.06	94.67	95.98

```
DrawScores(PCA.res, drawNames=TRUE, type.obj = "PCA",
        createWindow=FALSE, main = paste0("PCA score plot for ", dataname),
        color = group, pch = group,axes =c(1,2))
```




```
DrawLoadings(PCA.res,type.obj = "PCA",
        createWindow=FALSE, main = paste0("PCA loadings plot for ", dataname),
        axes = c(1:2),  loadingstype="s", num.stacked = num.stacked, xlab="ppm")
```

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



Unsupervised clustering

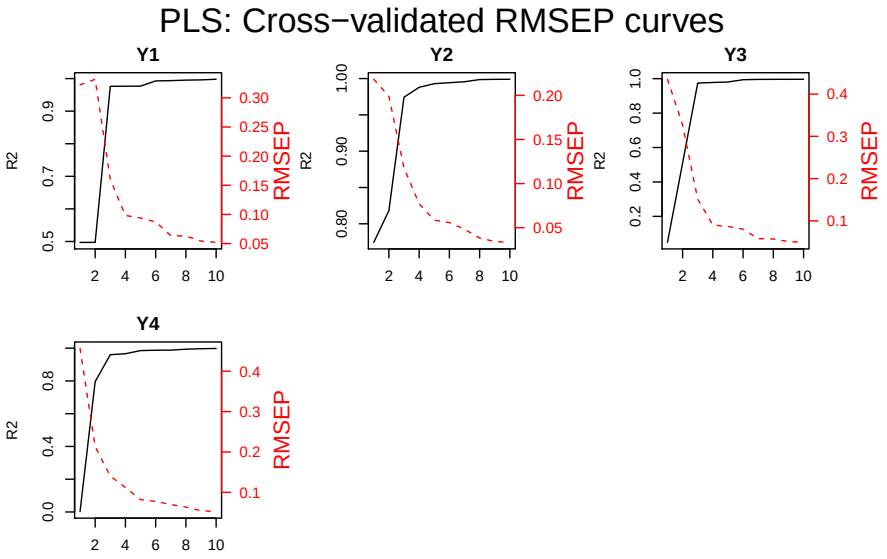
```
ClustMIC.res = MBXUCL::ClustMIC(Intensities = dataset, nClust = nClust,
                                Trcl = group_num, Dendr = FALSE)

ClustMIC.res <- ClustMIC.res[c(1,3,5,7)] # avec clustering hiérarchique
names(ClustMIC.res) <- c("Dunn index", "Davies-Bouldin index", "Rand index", "Adjusted Rand index")
pander(ClustMIC.res, caption ="Clustering hiérarchique")
```

Table 4: Clustering hiérarchique

Dunn index	Davies-Bouldin index	Rand index	Adjusted Rand index
0.8262	0.6712	1	1

PLS-DA

```
PLSDA.res = PLSDA(x = dataset, y = group_num, nLV = nLVPLSDA, drawRMSEP = TRUE)
```

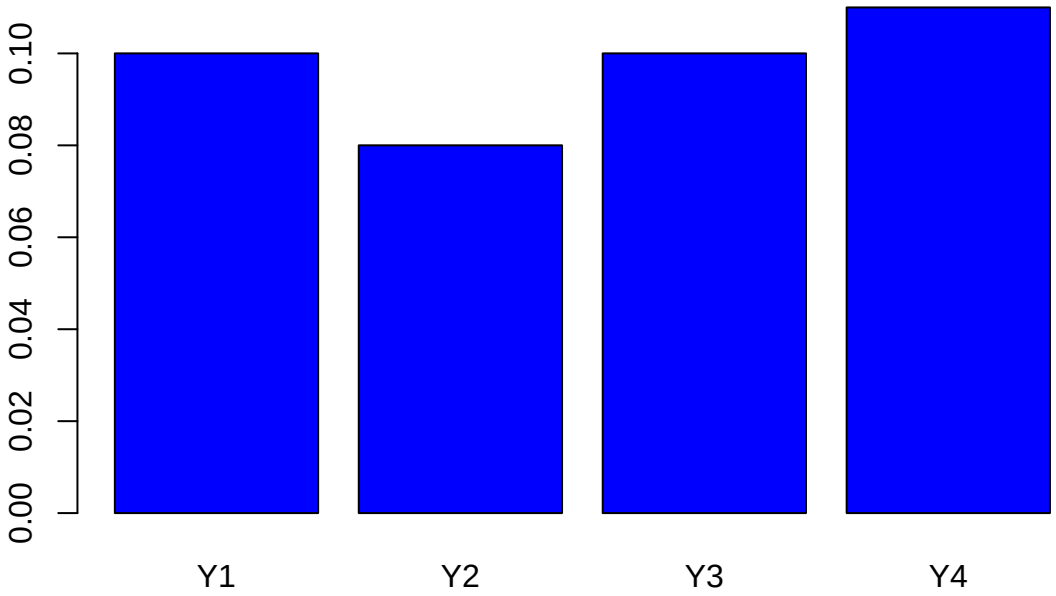



```
perf.plsda = PLSDA.res[4:6]

perf.plsda <- matrix(unlist(perf.plsda) , ncol=4, byrow = TRUE,
                     dimnames = list(names(perf.plsda), names(perf.plsda[[1]])))

perf.plsda <- round(perf.plsda,2)
pander(perf.plsda, caption = "Validation PLS-DA")
```

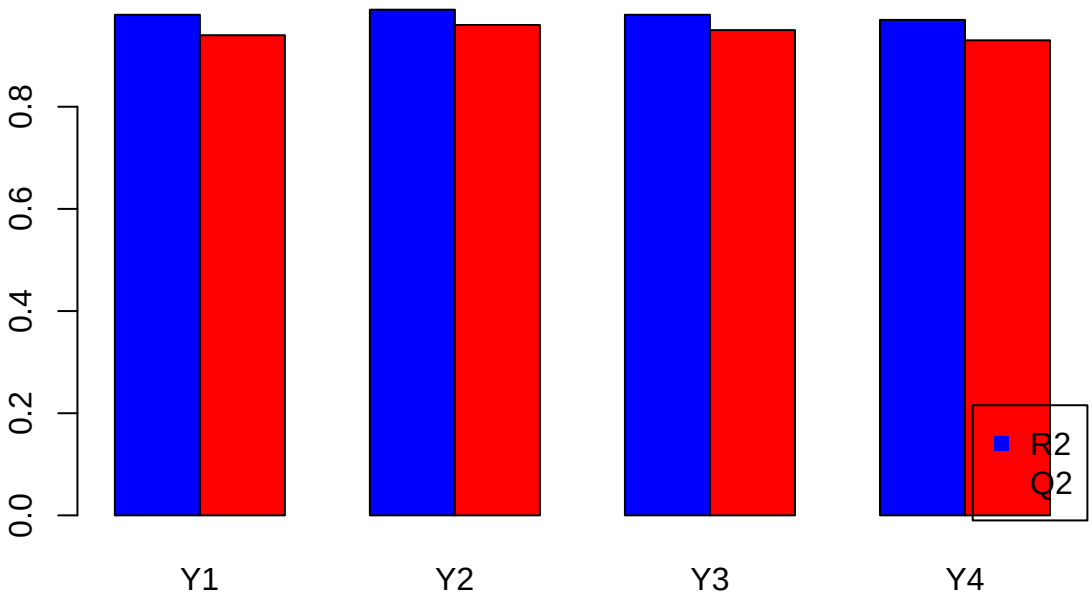
Table 5: Validation PLS-DA

	Y1	Y2	Y3	Y4
RMSEP	0.1	0.08	0.1	0.11
R2	0.98	0.99	0.98	0.97
Q2	0.94	0.96	0.95	0.93

```
par(mfrow=c(1,1),xpd=TRUE)
barplot(perf.plsda[1,], beside = TRUE, col="blue", main = "RMSEP")
```




```
barplot(perf.plsda[2:3,], beside = TRUE, col=c("blue","red"), main = "R2 and Q2")
legend("bottomright", legend = c("R2", "Q2"), col = c("blue","red"),
       pch = 15)
```



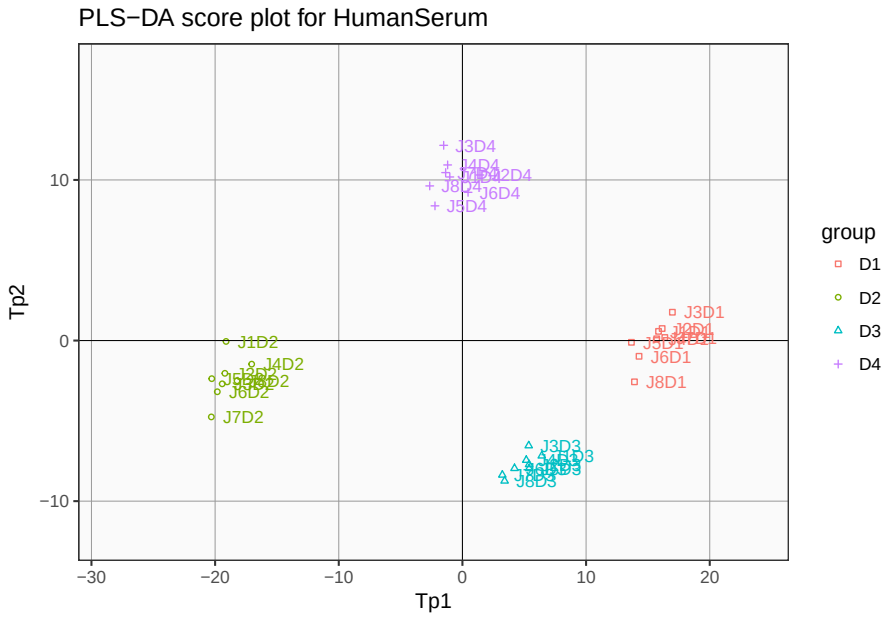

```
pander("Q2 cumulé pour toutes les réponses")
```

Q2 cumulé pour toutes les réponses

```
mean(perf.plsda["Q2",])
```

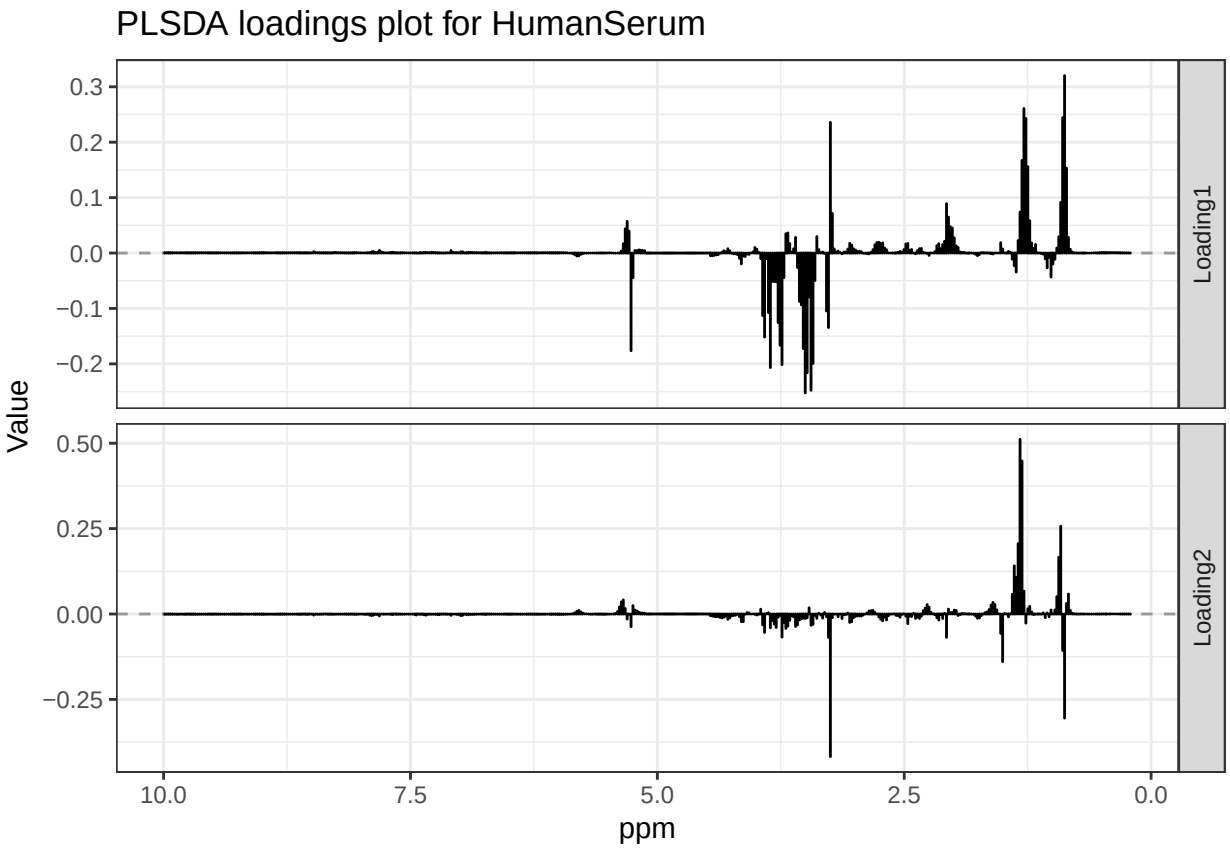
```
## [1] 0.945
```

```
DrawScores(PLSDA.res, drawNames=TRUE, type.obj = "PLSDA",
        createWindow=FALSE, main = paste0("PLS-DA score plot for ", dataname),
        color = group, pch = group,axes =c(1,2))
```




```
DrawLoadings(PLSDA.res,  type.obj = "PLSDA",
        createWindow=FALSE, main = paste0("PLSDA loadings plot for ", dataname),
        axes = c(1:2),  loadingstype="s", num.stacked = num.stacked, xlab = "ppm")
```

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



Evaluation de la répétabilité de différentes corrections de la Baseline

```
#============ PARAMETRES A MODIFIER quand vous utilisez MBXUCL
## A. Paramètres globaux liés aux noms et chemin d'accès des dataset

dataname <- "HumanSerum" # nom du jeu de données

## Mettre ici le nom du chemin d'accès du répertoire où sont les spectres prétraités

## Mettre ici le nom du chemin d'accès du répertoire où sont les FID
data.path <-  system.file("extdata", package = "PepsNMR") # données du package PepsNMR
dir(data.path) # ce que contient le répertoire data.path
```

```
## [1] "Group_HS.csv" "HumanSerum"
```

```
# ## B. Définition des groupes de spectres par sujet.
# group_HS <- read.csv(file.path(data.path, "Group_HS.csv"), header = TRUE, sep = ";")
# group <- as.factor(substr(group_HS[,1],4,5)) # donneur comme groupe
# table(group) # fréquences de chaque groupe
# group_num <- group_HS[,2]  # Version numerique des noms de groupes
# table(group_num) # fréquences de chaque groupe


## C. Setup des paramètres de représentation graphique des paramètres de prétraitement
```

```
num.stacked <- 2 # nombre de graphes par sortie graphique


## D. paramètres des méthodes incluses dans le MIC à modifier
ncompPCA = 4 # nombre de composantes en PCA
nClust = 4 # nombre de clusters pour le clustering
nLVPLSDA = 4 # nombre de variables latentes en PLS-DA
```

Fonction de prétraitement depuis la correction de la baseline

```
# Paramètres de la baseline qu'on veut faire varier
all_cond_BC <- expand.grid(lambda = c(1e5, 1e6,1e7,1e8), p = c(0.1,0.05, 0.01))
pander(all_cond_BC, caption = "Paramètres de correction de la baseline à faire varier")


BCconditions <- paste(paste("L:", all_cond_BC$lambda), paste("p:", all_cond_BC$p), sep=" ")


## PepsNMR prétraitement

# Lecture des Fids
fidList <- ReadFids(path = file.path(data.path, dataname), subdirs = FALSE)
# créer la matrice spectrale
Fid_data <- fidList[["Fid_data"]]
# créer la matrice d'infos sur les paramètres d'acquisition
Fid_info <- fidList[["Fid_info"]]


# réarranger les noms de ligne dans Fid_data et Fid_info pour les
# rendre plus lisibles
rownames(Fid_data) <- substr(rownames(Fid_data),1,5)
rownames(Fid_data) <- sub("-","", rownames(Fid_data))
rownames(Fid_info) <- rownames(Fid_data)

# GroupDelayCorrection
Fid_data_GDC <- GroupDelayCorrection(Fid_data, Fid_info)
# Solvent Suppression
Fid_data_SS <- SolventSuppression(Fid_data_GDC, returnSolvent = FALSE)
# Apodization
Fid_data_Apod <- Apodization(Fid_data_SS, Fid_info)
# FourierTransform
Spectrum_data_FT <- FourierTransform(Fid_data_Apod, Fid_info)
# ZeroOrderPhaseCorrection
Spectrum_data_ZOPC <- ZeroOrderPhaseCorrection(Spectrum_data_FT)
# InternalReferencing
Spectrum_data_IR <- InternalReferencing(Spectrum_data_ZOPC, Fid_info)


#============== After  Alignement pre-processing
AfterAlign_preprocess <- function(lambda.bc , p.bc) {

## Baseline correction avec les paramètres lambda.bc et p.bc à faire varier
```

```
Spectrum_data <- BaselineCorrection(Spectrum_data_IR, lambda.bc = lambda.bc,
                                    p.bc = p.bc)

## Suppression des valeurs négatives
Spectrum_data <- NegativeValuesZeroing(Spectrum_data)
## Alignement - Warping
Spectrum_data <- Warping(Spectrum_data,reference.choice = "before")
## WindowSelection
Spectrum_data <-  WindowSelection(Spectrum_data, from.ws = 10, to.ws = 0.2)
# Bucketing avec le Window selection intégré
Spectrum_data <- Bucketing(Spectrum_data, intmeth = "t", mb=500)
## Suppression des régions non informatives
Spectrum_data <- RegionRemoval(Spectrum_data, typeofspectra ="serum")
## Normalisation
Spectrum_data <- Normalization(Spectrum_data, type.norm="mean")

# Exportation des résultats
Re_Spectrum_data=Re(Spectrum_data)
return(Re_Spectrum_data)
}

# application de la fonction AfterAlign_preprocess sur les spectres
# prétraités jusqu'à InternalReferencing avec les différents parmaètres de
# la correction de la baseline
res_Preprocessing <- mapply(FUN = AfterAlign_preprocess, lambda.bc=all_cond_BC$lambda,
       p.bc=all_cond_BC$p, SIMPLIFY=FALSE)
```

Fonction EvalRepet

```
# Création de la fonction EvalRepet qui va appliquer les différents outils
# MIC sur les jeux de données générés juste au dessus

EvalRepet <- function(Re_Spectrum_data, group, groupN, nClust,
                      nLVPLSDA){
# Re_Spectrum_data : spectre prétraité
# group et groupN: classe et classe numérique des observations
# nClust : nombre de clusters pour le clustering
# nLVPLSDA: nombre de variables latentes pour le PLS-DA



# Inertia
Inertia.res = MBXUCL::Inertia(x = Re_Spectrum_data, y = groupN, print = FALSE)

# PCA
res.pca <-  SVDforPCA(Re_Spectrum_data)
scores12 <- DrawScores(res.pca, type.obj = "PCA",  color = group_num,
                       axes = c(1:2), main = "PCA scores")

scores34 <-DrawScores(res.pca, type.obj = "PCA",  color = group, axes = c(3:4))

# clustering
```

```r
ClustMIC.res = MBXUCL::ClustMIC(Intensities = Re_Spectrum_data, nClust = nClust,
                                Trcl = groupN, Dendr = FALSE)

# PLS-DA
PLSDA.res = PLSDA(x = Re_Spectrum_data, y = groupN,
                  nLV = nLVPLSDA, drawRMSEP = FALSE)

scoresPLS12 <- DrawScores(PLSDA.res, drawNames=TRUE, type.obj = "PLSDA",
        createWindow=FALSE, main = paste0("PLSDA score plot for ", dataname),
        color = group, pch = group,axes =c(1,2))

loadingsPLS12 <- DrawLoadings(PLSDA.res,  type.obj = "PLSDA",
        createWindow=FALSE, main = paste0("PLSDA loadings plot for", dataname),
        axes = c(1:2),  loadingstype="l", num.stacked = 2)


return(list(scores12=scores12,scores34=scores34,
            Inertia.res=Inertia.res,
            ClustMIC.res=ClustMIC.res,
            PLSDA.res=PLSDA.res,
            scoresPLS12=scoresPLS12,
            loadingsPLS12=loadingsPLS12))

}

# Application de la fonction EvalRepet à toutes les matrices spectrales
# contenues dans res_Preprocessing et prétraitées avec des paramètres
# de correction de la baseline différents ----------------------

res.EvalRepet <- lapply(res_Preprocessing, FUN=EvalRepet, group = group,
                        groupN = group_num, nClust=4, nLVPLSDA=4)

names(res.EvalRepet) <- BCconditions


# Arrangement des sorties de la fonction EvalRepet ---------------------

# inertia
res.inertia <- sapply(res.EvalRepet, function(x) x[["Inertia.res"]][["Between_within"]]["Percentage",])
colnames(res.inertia) <- BCconditions
rownames(res.inertia) <- c("Inertia Between groups","Inertia Within groups", "Total inertia")
res.inertia <- round(res.inertia,2)

# PCA
PCAscores12 <- lapply(res.EvalRepet, function(x) x[["scores12"]])
names(PCAscores12) <- BCconditions

# clustering
res.clustering <- sapply(res.EvalRepet, function(x) x[["ClustMIC.res"]])
colnames(res.clustering) <- BCconditions
dimnam <- dimnames(res.clustering)
res.clustering <- matrix(data = unlist(res.clustering),
                         ncol = length(BCconditions), dimnames = dimnam,
```

```
                          byrow = FALSE)
res.clustering <- res.clustering[c(1,3,5,7),]
rownames(res.clustering) <- c("Dunn index", "Davies-Bouldin index", "Rand index", "Adjusted Rand index")
res.clustering <- round(res.clustering,2)

# PLS-DA
Q2_PLS <- sapply(res.EvalRepet, function(x) x[["PLSDA.res"]][["Q2"]])
Q2cum_PLS <- round(colMeans(Q2_PLS),2)

PLSscores12 <- lapply(res.EvalRepet, function(x) x[["scoresPLS12"]])
names(PLSscores12) <- BCconditions
```

Paramètres de la correction de la baseline λ et p

```
pander("lambda.bc (L) and p.bc (p) values for Baseline Correction")
```

lambda.bc (L) and p.bc (p) values for Baseline Correction

```
pander(all_cond_BC)
```

lambda	p
1e+05	0.1
1e+06	0.1
1e+07	0.1
1e+08	0.1
1e+05	0.05
1e+06	0.05
1e+07	0.05
1e+08	0.05
1e+05	0.01
1e+06	0.01
1e+07	0.01
1e+08	0.01

Inertia

```
pander(res.inertia)
```

Table 7: Table continues below

	L: 1e+05 p: 0.1	L: 1e+06 p: 0.1
Inertia Between groups	84.76	86.22
Inertia Within groups	15.24	13.78
Total inertia	100	100

Table 8: Table continues below

	L: 1e+07 p: 0.1	L: 1e+08 p: 0.1
Inertia Between groups	87.49	89.2

	L: 1e+07 p: 0.1	L: 1e+08 p: 0.1
Inertia Within groups	12.51	10.8
Total inertia	100	100

Table 9: Table continues below

	L: 1e+05 p: 0.05	L: 1e+06 p: 0.05
Inertia Between groups	85.16	86.72
Inertia Within groups	14.84	13.28
Total inertia	100	100

Table 10: Table continues below

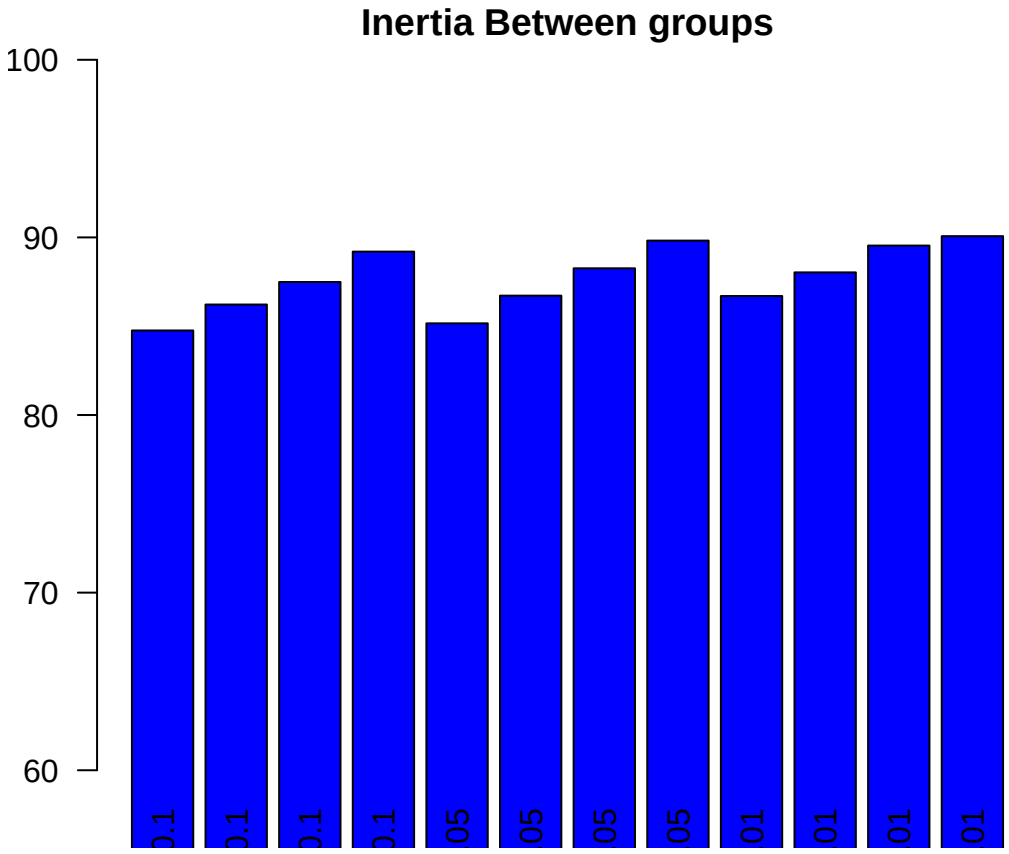
	L: 1e+07 p: 0.05	L: 1e+08 p: 0.05
Inertia Between groups	88.26	89.82
Inertia Within groups	11.74	10.18
Total inertia	100	100

Table 11: Table continues below

	L: 1e+05 p: 0.01	L: 1e+06 p: 0.01
Inertia Between groups	86.7	88.03
Inertia Within groups	13.3	11.97
Total inertia	100	100

	L: 1e+07 p: 0.01	L: 1e+08 p: 0.01
Inertia Between groups	89.54	90.07
Inertia Within groups	10.46	9.93
Total inertia	100	100

```
par(mar=c(2,6,2,2))
barplot(res.inertia["Inertia Between groups",],
     main= "Inertia Between groups", col="blue", ylim = c(60,100), las=2)
```



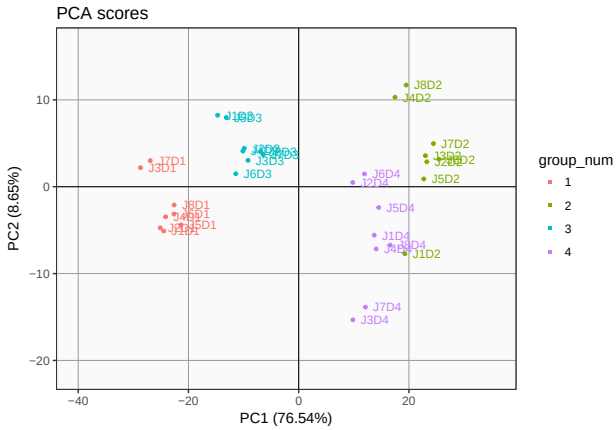
PCA scores plots

```
pander("PCA scores plots")
```

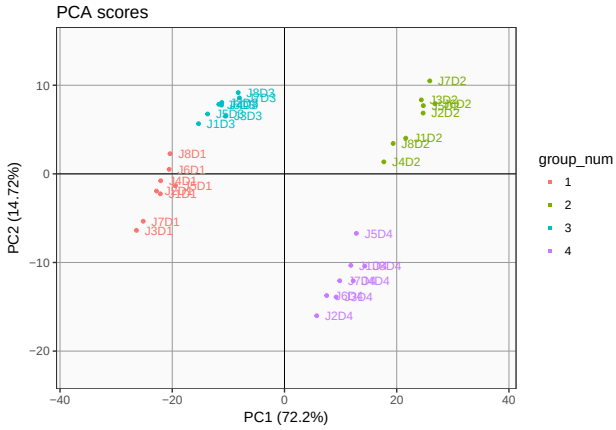
PCA scores plots

```
PCAscores12
```

```
## $`L: 1e+05 p: 0.1`
```

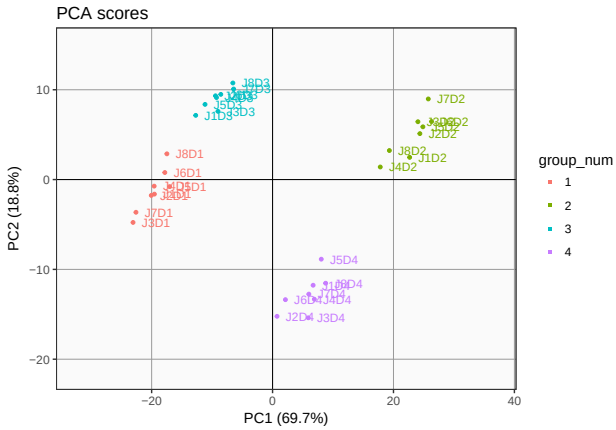



```
##
## $`L: 1e+06 p: 0.1`
```



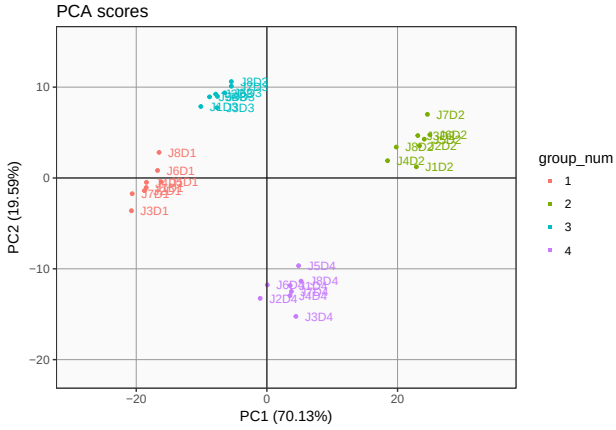
##

$`L: 1e+07 p: 0.1`



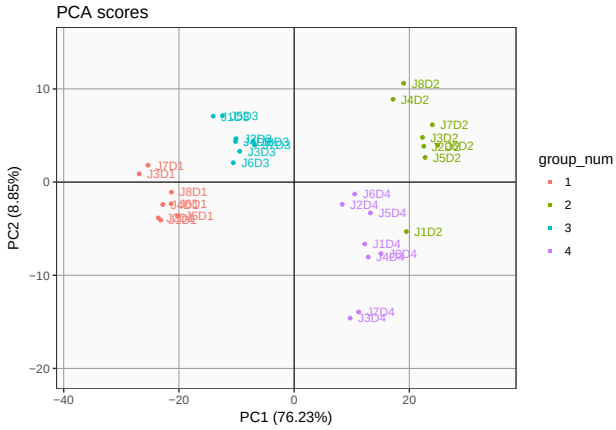
##

$`L: 1e+08 p: 0.1`



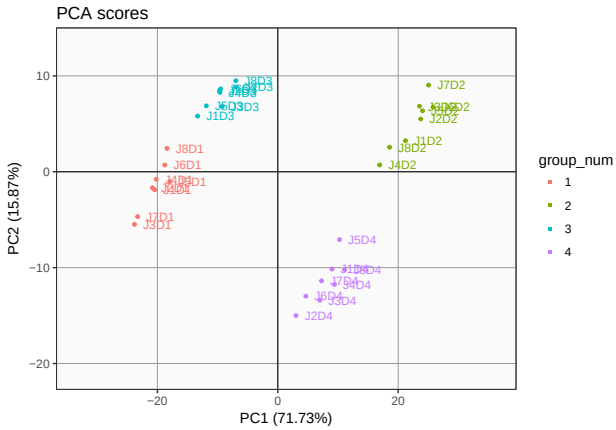
##

$`L: 1e+05 p: 0.05`



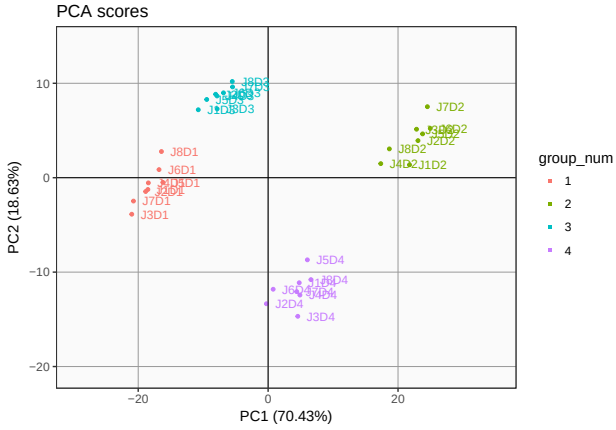
##

$`L: 1e+06 p: 0.05`



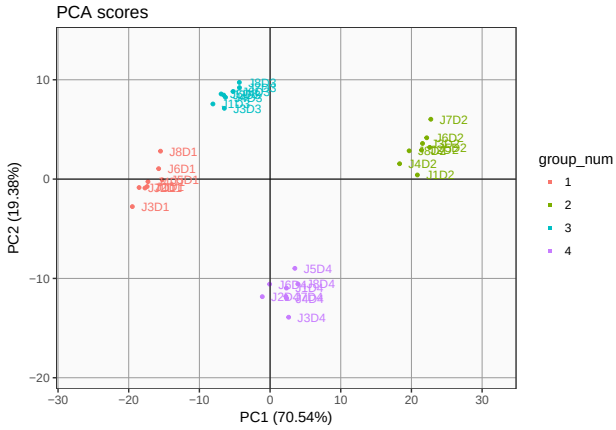
##

$`L: 1e+07 p: 0.05`



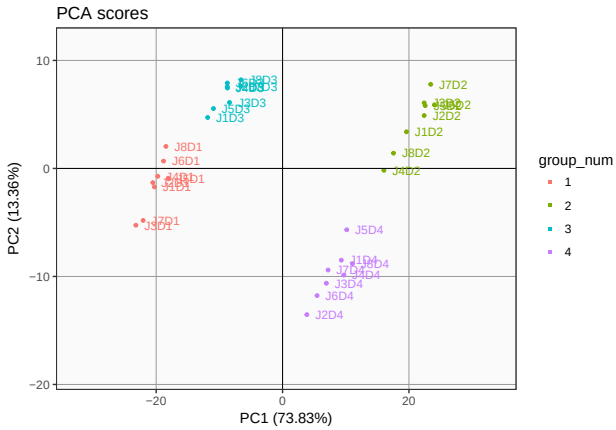
##

$`L: 1e+08 p: 0.05`



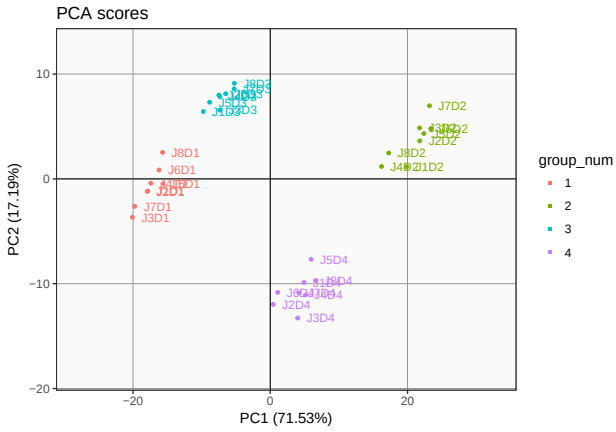
##

$`L: 1e+05 p: 0.01`



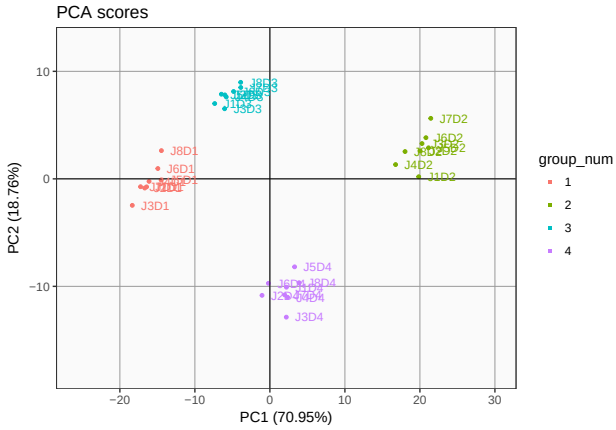
##

$`L: 1e+06 p: 0.01`



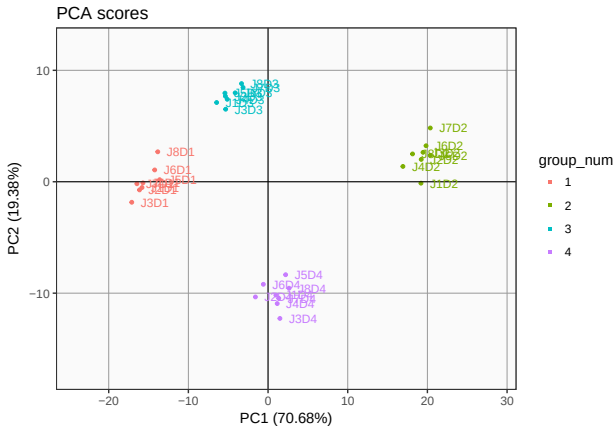
##

$`L: 1e+07 p: 0.01`



##

$`L: 1e+08 p: 0.01`



Clustering

```
pander("clustering")
```

clustering

```
pander(res.clustering)
```

Table 13: Table continues below

	L: 1e+05 p: 0.1	L: 1e+06 p: 0.1	L: 1e+07 p: 0.1
Dunn index	0.5	0.59	0.64
Davies-Bouldin index	1.01	0.92	0.8
Rand index	0.97	1	1
Adjusted Rand index	0.91	1	1

Table 14: Table continues below

	L: 1e+08 p: 0.1	L: 1e+05 p: 0.05
Dunn index	0.73	0.49
Davies-Bouldin index	0.64	0.96

	L: 1e+08 p: 0.1	L: 1e+05 p: 0.05
Rand index	1	0.97
Adjusted Rand index	1	0.91

Table 15: Table continues below

	L: 1e+06 p: 0.05	L: 1e+07 p: 0.05
Dunn index	0.64	0.68
Davies-Bouldin index	0.84	0.76
Rand index	1	1
Adjusted Rand index	1	1

Table 16: Table continues below

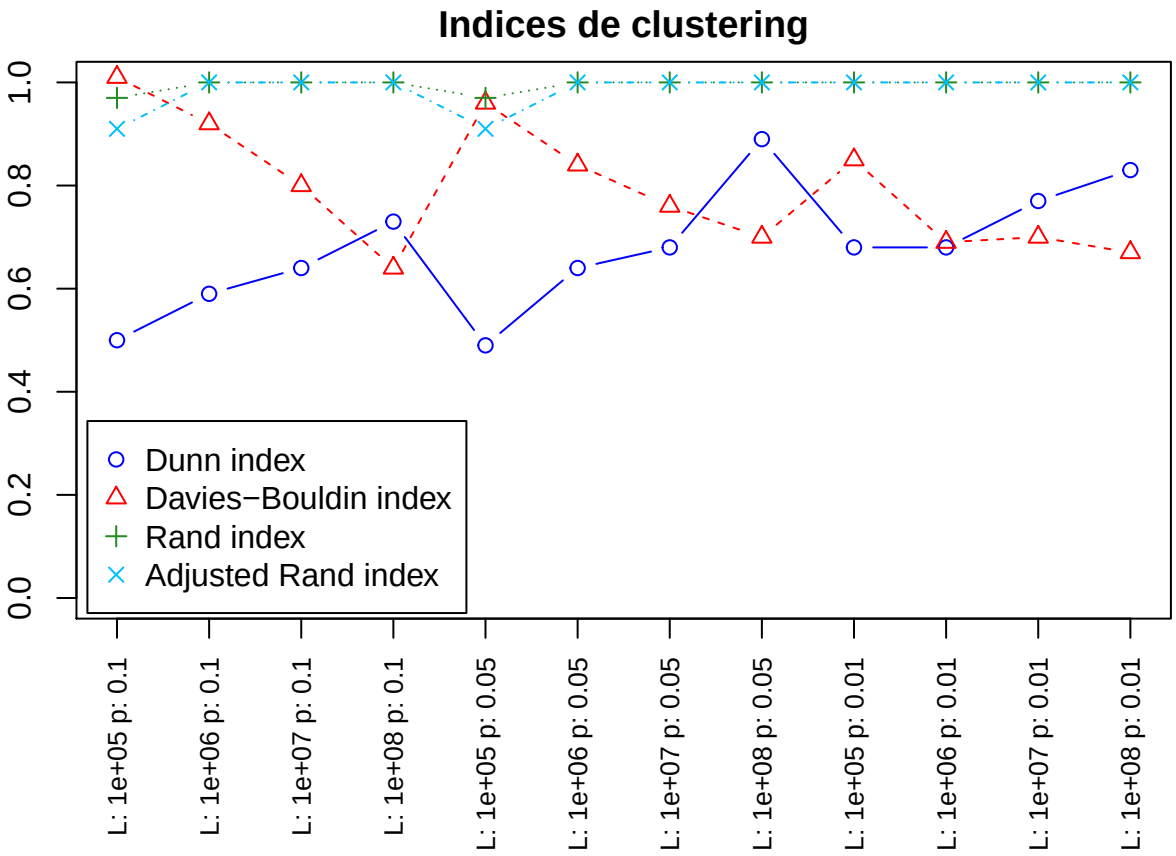
	L: 1e+08 p: 0.05	L: 1e+05 p: 0.01
Dunn index	0.89	0.68
Davies-Bouldin index	0.7	0.85
Rand index	1	1
Adjusted Rand index	1	1

Table 17: Table continues below

	L: 1e+06 p: 0.01	L: 1e+07 p: 0.01
Dunn index	0.68	0.77
Davies-Bouldin index	0.69	0.7
Rand index	1	1
Adjusted Rand index	1	1

	L: 1e+08 p: 0.01
Dunn index	0.83
Davies-Bouldin index	0.67
Rand index	1
Adjusted Rand index	1

```
par(mar=c(6,2,2,2))
matplot(1:length(BCconditions),t(res.clustering), ylim=c(0,1),type="b",col=c("blue", "red", "forestgreen
        pch=1:4, xaxt = "n", main = "Indices de clustering", xlab="")
legend("bottomleft", inset=0.01, legend=rownames(res.clustering),
       pch=1:4, col=c("blue", "red","forestgreen", "deepskyblue"))
axis(side = 1, at = 1:length(BCconditions), labels = BCconditions,
     las=2,cex.axis=0.8)
```



PLS-DA

```
pander("PLS-DA Q2cum")
```

PLS-DA Q2cum

```
pander(Q2cum_PLS)
```

Table 19: Table continues below

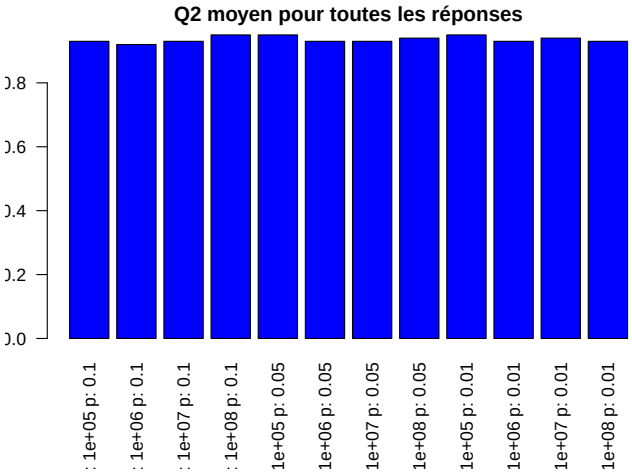
L: 1e+05 p: 0.1	L: 1e+06 p: 0.1	L: 1e+07 p: 0.1	L: 1e+08 p: 0.1
0.93	0.92	0.93	0.95

Table 20: Table continues below

L: 1e+05 p: 0.05	L: 1e+06 p: 0.05	L: 1e+07 p: 0.05	L: 1e+08 p: 0.05
0.95	0.93	0.93	0.94

L: 1e+05 p: 0.01	L: 1e+06 p: 0.01	L: 1e+07 p: 0.01	L: 1e+08 p: 0.01
0.95	0.93	0.94	0.93

```
par(mar=c(6,2,2,2))
barplot(Q2cum_PLS, main = "Q2 moyen pour toutes les réponses", col="blue", las=2)
```

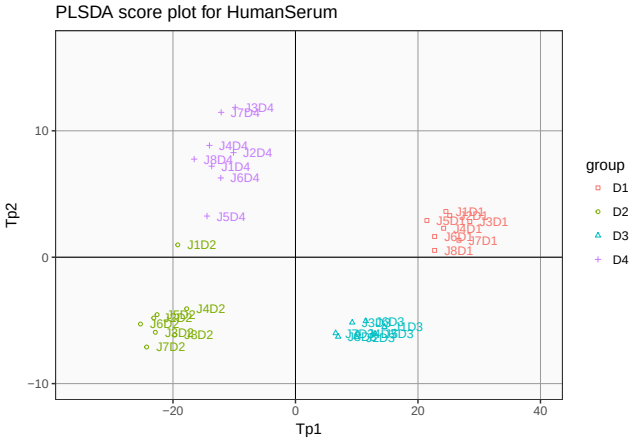



```
pander("PLS-DA scores plots")
```

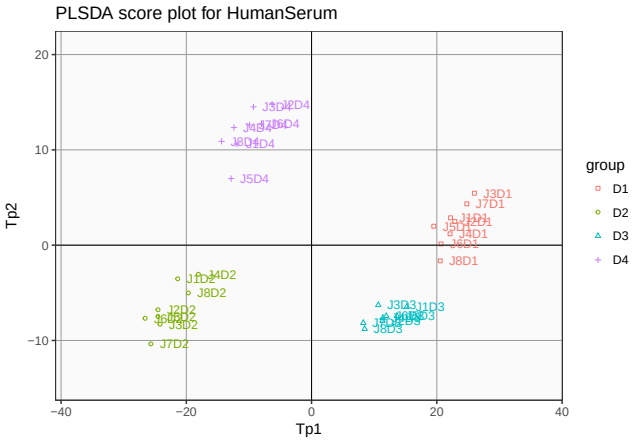
PLS-DA scores plots

```
PLSscores12
```

```
## $`L: 1e+05 p: 0.1`
```

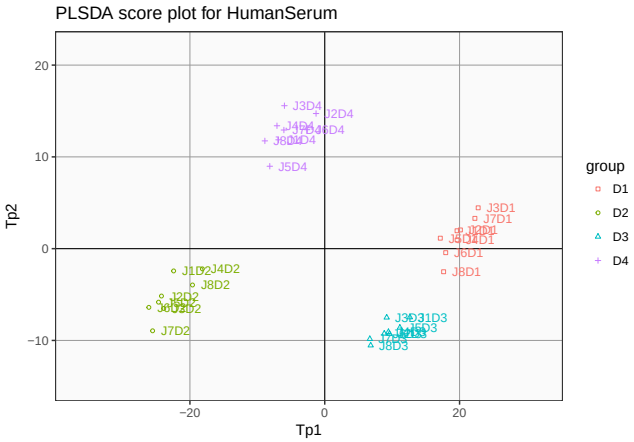



```
##
## $`L: 1e+06 p: 0.1`
```



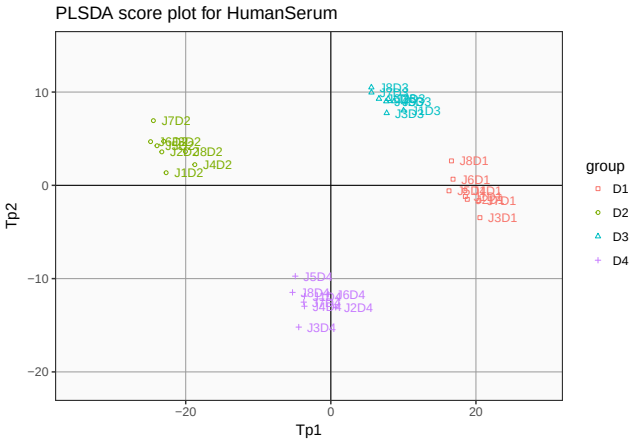
##

$`L: 1e+07 p: 0.1`



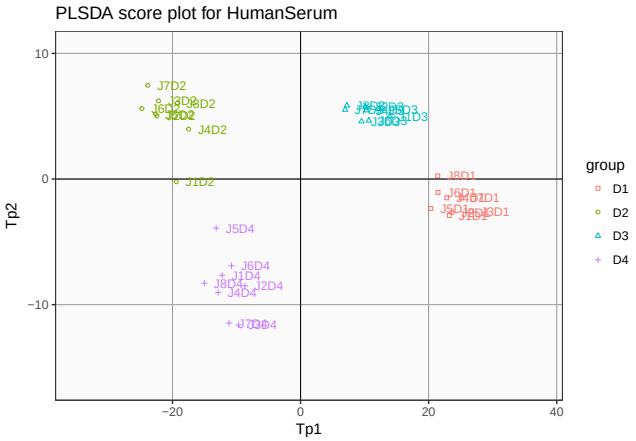
##

$`L: 1e+08 p: 0.1`



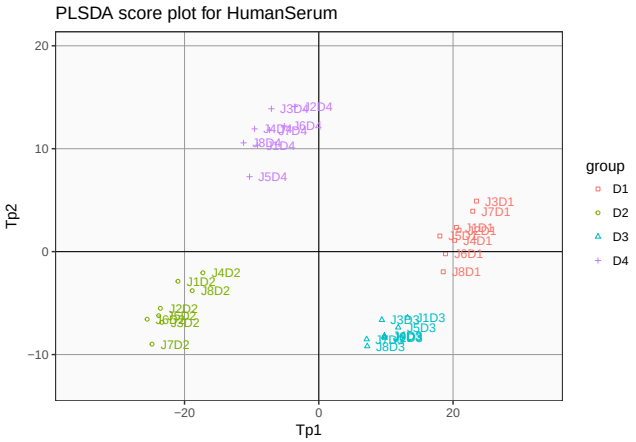
##

$`L: 1e+05 p: 0.05`



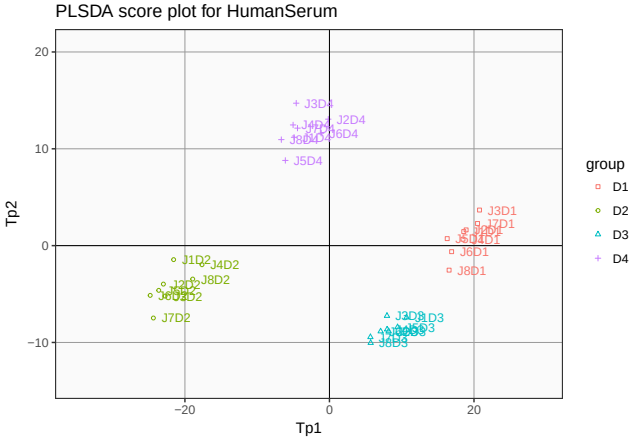
##

$`L: 1e+06 p: 0.05`



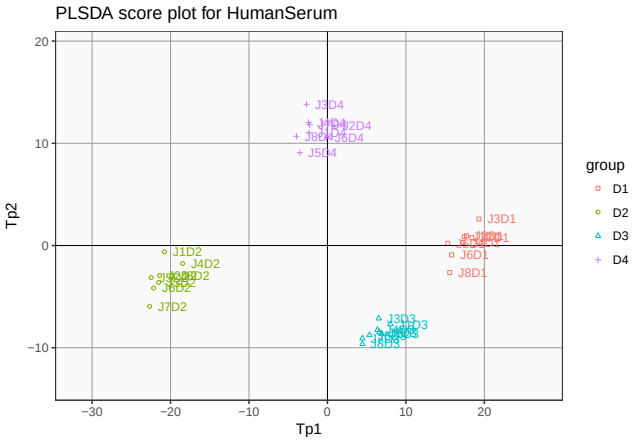
##

$`L: 1e+07 p: 0.05`



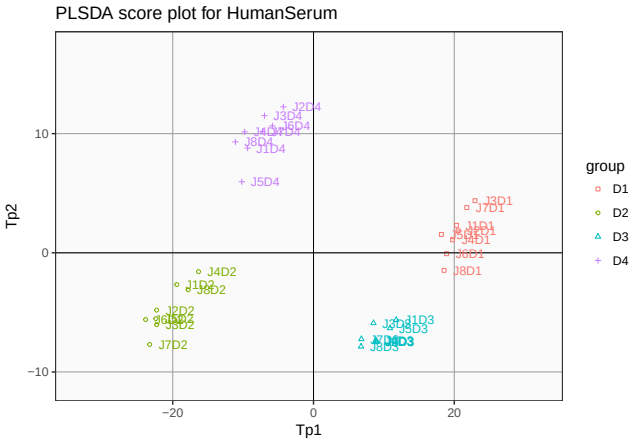
##

$`L: 1e+08 p: 0.05`



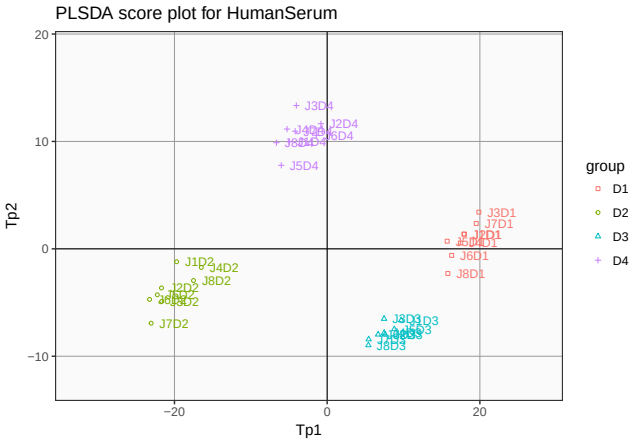
##

$`L: 1e+05 p: 0.01`



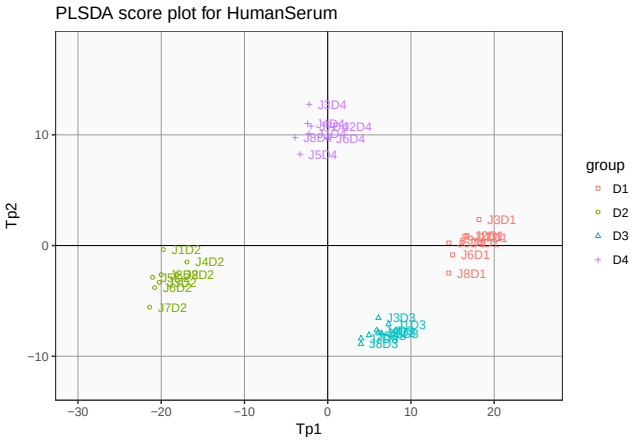
##

$`L: 1e+06 p: 0.01`



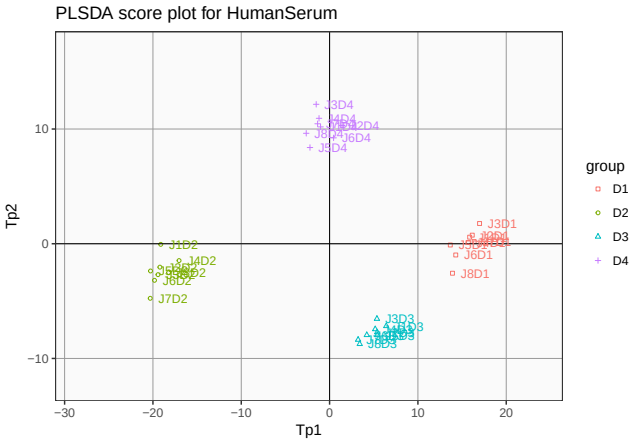
##

$`L: 1e+07 p: 0.01`


PLSDA score plot for HumanSerum
Tp2
Tp1
group
D1
D2
D3
D4

##

$`L: 1e+08 p: 0.01`


PLSDA score plot for HumanSerum
Tp2
Tp1
group
D1
D2
D3
D4