Mini-projet 2021 - Module 6 Bioinformatique Intégrative

Magali Monnoye

2021-05-31

Contents

J'importe les librairies

Table 1: Loaded required libraries

libraries
knitr
FactoMineR
factoextra
gprofiler2
pheatmap
dplyr

Synopsis du projet

Données

Pavkovic, M., Pantano, L., Gerlach, C.V. et al. Multi omics analysis of fibrotic kidneys in two mouse models. Sci Data 6, 92 (2019)

Samples from two mouse models were collected. The first one is a reversible chemical-induced injury model (folic acid (FA) induced nephropathy). The second one is an irreversible surgically-induced fibrosis model (unilateral ureteral obstruction (UUO)). mRNA and small RNA sequencing, as well as 10-plex tandem mass tag (TMT) proteomics were performed with kidney samples from different time points over the course of fibrosis development.

Rappel sur les échantillons:

Deux modèles de fibrose rénale chez la souris sont étudiés:

1. Le premier est un modèle de néphropathie réversible induite par l'acide folique (folic acid (FA)). Les souris ont été sacrifiées avant le traitement (normal), puis à jour 1, 2, 7 et 14 (day1,. . .) après une seule injection d'acide folique.
2. Le second est un modèle irréversible induit chrirurgicalement (unilateral ureteral obstruction (UUO)). les souris ont été sacrifiées avant obstruction (day 0) et à 3, 7 et 14 jours après obstruction par ligation de l'uretère du rein gauche.

A partir de ces extraits de rein, l'ARN messager total et les petits ARNs ont été séquencés et les protéines caratérisées par spectrométrie de masse en tandem (TMT).

Supplementary material of the article with all the data tables (zip archive):

Les données se trouvent aussi dans le dépôt github

Les comptages du modèle FA sont dans tables/fa/results/counts/ pour la transcriptomique tables/pfa/results/counts/ pour la protéomique

Les comptages du modèle UUO sont dans tables/uuo/results/counts/ pour la transcriptomique tables/puuo/results/counts/ pour la protéomique

Nous travaillerons sur le modèle FA.

Analyses demandées

- Analyse d'expression différentielle pour les données de protéomique et transcriptomique => identifier les gènes/protéines significativement différentiellement exprimés dans le modèle FA en comparant Day 7 à Day 0.
- Analyse multi-omique (transcripto + protéo) avec, au choix, MOFA, mixOmics, mixKernel ou d'autres outils de factorisation multi-matrices. Vous pouvez soit vous focaliser sur un time point, soit intégrer les différents time points, en partant des données normalisées fournies dans le matériel supplémentaire du papier.
- Construction de réseau avec WGCNA à partir des données transcriptomiques.

Reconstruct the co-expression network from all the time points of the FA transcriptomics data. Propose to filter and remove all the zero expressed genes, the NAs and the less informative genes from the transcriptomics data. (I remove all the genes that are not expressed in at least 9 out of the 18 conditions (expression > 1 TPM in 9) and then filter with the coefficient of variation > 0.75).

Then apply the first part of the network reconstruction steps as we saw them on the WGCNA course until the module predictions.

Instead of using WGCNA's module prediction routines, apply a universal threshold of 0.5 on the adjacency matrix, and obtain an adjacency matrix that is reduced in size. This is the network. Import it to Cytoscape with aMatReader plugin.

Visualize, analyze the network and superimpose the proteomics data on it.

Colorez dans le réseau choisi les noeuds en fonction des données de protéomiques avec un gradient de couleur correspondant au fold-change des données de protéomique.

Remise du rapport

Vous fournirez un rapport au format pdf généré à partir d'un Rmd (déposez-nous impérativement les 2 fichiers, Rmd et pdf, avec comme nom de fichier "NOM-PRENOM_evaluation-m6-2021" + .Rmd ou .pdf dans le dossier /shared/projects/dubii2021//m6-bioinfo-integr/mini-projet/), avec une page d'introduction et deux figures maximum par analyse.

Vos travaux doivent être reproductibles, pensez à décrire et justifier les différentes étapes, seuils, extractions ... Bon courage ! Nous sommes disponibles sur Slack en cas de besoin.

Import données

Je télécharge les quatre fichiers dans un dossier local `~/Module6Projet`, et les charge dans les data.frames suivants:

- Données brutes de transcriptome: `fa_expr`
- Métadonnées transcriptome: `fa_meta`
- Données brutes de proteomique: `pfa_expr`
- Métadonnées transcriptome: `pfa_meta`

J'importe directement les données normalisées

Download the files

```
#' @title Download a file only if it is not yet here
#' @author Jacques van Helden email{Jacques.van-Helden@@france-bioinformatique.fr}
#' @param url_base base of the URL, that will be prepended to the file name
#' @param file_name name of the file (should not contain any path)
#' @param local_folder path of a local folder where the file should be stored
#' @return the function returns the path of the local file, built from local_folder and file_name
#' @export
downloadOnlyOnce <- function(url_base,
                             file_name,
                             local_folder) {

  ## Define the source URL
  url <- file.path(url_base, file_name)
  message("Source URL\n\t",  url)

  ## Define the local file
  local_file <- file.path(local_folder, file_name)

  ## Create the local data folder if it does not exist
  dir.create(local_folder, showWarnings = FALSE, recursive = TRUE)

  ## Download the file ONLY if it is not already there
  if (!file.exists(local_file)) {
    message("Downloading file from source URL to local file\n\t",
            local_file)
    download.file(url = url, destfile = local_file)
  } else {
    message("Local file already exists, no need to download\n\t",
            local_file)
  }

  return(local_file)
}
```

```
## Specify the basic parameters
pavkovic_base <- "https://github.com/DU-Bii/module-3-Stat-R/tree/master/stat-R_2021/data/pavkovic_2019"
pavkovic_folder <- "~/DUBii-m3_data/pavkovic_2019"

#### Dowload folic acid data and metadata ####

## Transcriptome data table
local_fa_file <- downloadOnlyOnce(
  url_base = pavkovic_base,
```

```
  file_name = "fa_raw_counts.tsv.gz",
  local_folder =  pavkovic_folder
)

## Transcriptome data table normalized
local_fa_file_norm <- downloadOnlyOnce(
  url_base = pavkovic_base,
  file_name = "fa_normalized_counts.tsv.gz",
  local_folder =  pavkovic_folder
)

## Transcriptome metadata
trans_metadata_file <- downloadOnlyOnce(
  url_base = pavkovic_base,
  file_name = "fa_transcriptome_metadata.tsv",
  local_folder =  pavkovic_folder
)

## Proteome data table
local_pfa_file <- downloadOnlyOnce(
  url_base = pavkovic_base,
  file_name = "pfa_model_counts.tsv.gz",
  local_folder =  pavkovic_folder
)

## Proteome data table normalized
local_pfa_file_norm <- downloadOnlyOnce(
  url_base = pavkovic_base,
  file_name = "pfa_model_log2_counts.tsv.gz",
  local_folder =  pavkovic_folder
)

## Proteome metadata
prot_metadata_file <- downloadOnlyOnce(
  url_base = pavkovic_base,
  file_name = "pfa_proteome_metadata.tsv",
  local_folder =  pavkovic_folder
)
```

Loading the files

```
#' @title Load a tab-separated value file and manually set row ames after having forced them to unique v
#' @author Jacques van Helden email{Jacques.van-Helden@@france-bioinformatique.fr}
#' @param file file path
#' @param header=1 Header is set to 1 by default
#' @param sep="\t" Column separator is set to tab by default
#' @param rownames.col=1 Column containing the row names
#' @param ... all other parameters are passed to read.delim()
#' @return a data frame with the loaded data
load_fix_row_names <- function(file,
                       header = 1,
                       sep = "\t",
```

```
                        rownames.col = 1,
                        ...) {
  x <- read.delim(file = file, ...)
  rownames(x) <- make.names(x[, rownames.col], unique = TRUE)
  x <- x[, -rownames.col]
  return(x)
}
```

```
## Load transcriptome data
fa <- read.delim(file = local_fa_file, sep = "\t", header = TRUE)

## Load same data with load_fix_row_names
fa_expr <- load_fix_row_names(file = local_fa_file, rownames.col = 1)
kable(head(fa_expr), caption = "Loaded with myEasyLad() fa")
```

Table 2: Loaded with myEasyLad() fa

	day1_d	ay1_d	ay1_d	ay14	day14	day14	day2_d	ay2_d	ay2_d	ay3_d	ay3_d	ay3_d	ay7_d	ay7_d	ay7_d	ormal	ormal	ormal	ormal
ENSMU																			
ENSMU																			
ENSMU																			
ENSMU																			
ENSMU																			
ENSMU																			

```
## Load transcriptome data normalized
fa_norm <- read.delim(file = local_fa_file_norm, sep = "\t", header = TRUE)

## Load same data with load_fix_row_names
fa_expr_norm <- load_fix_row_names(file = local_fa_file_norm, rownames.col = 1)
kable(head(fa_expr_norm), caption = "Loaded with myEasyLad() fa normalized")
```

Table 3: Loaded with myEasyLad() fa normalized

	day1_d	ay1_d	ay1_d	ay14	day14	day14	day2_d	ay2_d	ay2_d	ay3_d	ay3_d	ay3_d	ay7_d	ay7_d	ay7_d	ormal	ormal	ormal	ormal
ENSMU																			
ENSMU																			
ENSMU																			
ENSMU																			
ENSMU																			
ENSMU																			

```
## Load proteome data
pfa_expr <- load_fix_row_names(file = local_pfa_file, rownames.col = 1)
kable(head(pfa_expr), caption = "Loaded with myEasyLad() pfa")
```

Table 4: Loaded with myEasyLad() pfa

	normal_1	normal_2	day1_1	day1_2	day2_1	day2_2	day7_1	day7_2	day14_1	day14_2
ENSMUSG00000[illegible]	651.7200	335.5910	334.8460	197.1740	307.194	123.2060	272.6190	93.7247	196.1590	
ENSMUSG00000[illegible]	266.3590	175.4090	159.4190	234.8080	256.927	149.9380	315.0590	110.5880	126.3600	
ENSMUSG00000[illegible]	29.1331	57.7329	45.8475	81.6009	88.870	29.8560	44.5586	13.6292	27.6568	
ENSMUSG00000[illegible]	4784.0800	4064.4800	3917.2900	4599.0300	5957.030	2806.5200	6792.0900	2022.5100	3226.7500	
ENSMUSG00000[illegible]	1065.3900	914.2870	928.2760	1000.1000	1264.270	738.3520	1362.0700	466.4440	714.5870	
ENSMUSG00000[illegible]	89.8871	57.9041	76.3510	84.8474	105.245	72.8696	138.9810	40.8101	59.2117	

```
## Load proteome data normalized
pfa_expr_norm <- load_fix_row_names(file = local_pfa_file_norm, rownames.col = 1)
kable(head(pfa_expr_norm), caption = "Loaded with myEasyLad() pfa normalized")
```

Table 5: Loaded with myEasyLad() pfa normalized

	normal_1	normal_2	day1_1	day1_2	day2_1	day2_2	day7_1	day7_2	day14_1	day14_2
ENSMUSG00000[illegible]	5.037683	5.03274064.506491	4.492381	3.628542	3.932456	3.530692	3.685216	3.665305	4.026119	
ENSMUSG00000[illegible]	3.987831	3.74290663.571461	3.422865	3.880247	3.674968	3.813620	3.893704	3.903714	3.392438	
ENSMUSG00000[illegible]	0.784291	0.56459671.972258	1.630415	2.358439	2.146880	1.492104	1.081497	0.894504	1.208630	
ENSMUSG00000[illegible]	8.051796	7.90803308.103830	8.039705	8.170497	8.208356	8.038369	8.322345	8.095098	8.064739	
ENSMUSG00000[illegible]	5.834724	5.74151725.951775	5.962765	5.969607	5.972364	6.112220	6.004585	5.979016	5.890150	
ENSMUSG00000[illegible]	2.036202	2.17918891.976512	2.363160	2.414548	2.390031	2.774423	2.714986	2.468220	2.301407	

```
## Load transcriptome metadata
fa_meta <- read.delim(file = trans_metadata_file, sep = "\t", header = TRUE)
kable(fa_meta, caption = "Metadata for the transcriptome dataset fa")
```

Table 6: Metadata for the transcriptome dataset fa

	dataType	sampleName	condition	sampleNumber	color
1	transcriptome	day14_1	day14	1	#FF4400
2	transcriptome	day14_2	day14	2	#FF4400
3	transcriptome	day14_3	day14	3	#FF4400
4	transcriptome	day1_1	day1	1	#BBD7FF
5	transcriptome	day1_2	day1	2	#BBD7FF
6	transcriptome	day1_3	day1	3	#BBD7FF
7	transcriptome	day2_1	day1	1	#F0BBFF
8	transcriptome	day2_2	day1	2	#F0BBFF
9	transcriptome	day2_3	day1	3	#F0BBFF
10	transcriptome	day3_1	day3	1	#FFFFDD
11	transcriptome	day3_2	day3	2	#FFFFDD
12	transcriptome	day3_3	day3	3	#FFFFDD
13	transcriptome	day7_1	day7	1	#FFDD88
14	transcriptome	day7_2	day7	2	#FFDD88
15	transcriptome	day7_3	day7	3	#FFDD88
16	transcriptome	normal_1	normal	1	#BBFFBB
17	transcriptome	normal_2	normal	2	#BBFFBB
18	transcriptome	normal_3	normal	3	#BBFFBB

```
## Load proteome metadata
pfa_meta <- read.delim(file = prot_metadata_file, sep = "\t", header = TRUE)
kable(pfa_meta, caption = "Metadata for the proteome dataset pfa")
```

Table 7: Metadata for the proteome dataset pfa

dataType	sampleName	condition	sampleNumber	color
proteome	normal_1	normal	1	#BBFFBB
proteome	normal_2	normal	2	#BBFFBB
proteome	day1_1	day1	1	#FFFFDD
proteome	day1_2	day1	2	#FFFFDD
proteome	day2_1	day2	1	#BBD7FF
proteome	day2_2	day2	1	#BBD7FF
proteome	day3_1	day3	1	#F0BBFF
proteome	day3_2	day3	2	#F0BBFF
proteome	day7_1	day7	1	#FFDD88
proteome	day7_2	day7	2	#FFDD88
proteome	day14_1	day14	1	#FF4400
proteome	day14_2	day14	2	#FF4400

Structure de chaque dataframe.

```
str(fa_expr)
```

```
'data.frame':   46679 obs. of  18 variables:
 $ day1_1  : num  2278.8 0 36.3 13.2 0 ...
 $ day1_2  : num  1786.5 0 22.15 7.15 27.9 ...
 $ day1_3  : num  2368.62 0 39.48 1.12 6.9 ...
 $ day14_1 : num  627.758 0 14.471 0.867 5.692 ...
 $ day14_2 : num  559.2 0 10.2 0 1.9 ...
 $ day14_3 : num  611.434 0 31.691 0 0.655 ...
 $ day2_1  : num  2145.22 0 300.56 1.71 57.38 ...
 $ day2_2  : num  262.45 0 4.77 0 0 ...
 $ day2_3  : num  745.84 0 123.9 5.26 38.9 ...
 $ day3_1  : num  987.185 0 51.856 0.802 8.931 ...
 $ day3_2  : num  1077.65 0 8.43 0 6.97 ...
 $ day3_3  : num  1335.1 0 69.9 0 0 ...
 $ day7_1  : num  1096.08 0 6.67 0 7.94 ...
 $ day7_2  : num  1035.846 0 6.955 0.849 101.648 ...
 $ day7_3  : num  1090.04 0 42.58 1.71 0.65 ...
 $ normal_1: num  483.23 0 7.35 0.86 32.06 ...
 $ normal_2: num  1842.1 0 11.2 0 10.4 ...
 $ normal_3: num  475.7 0 1.03 0 0 ...
```

```
str(fa_expr_norm)
```

```
'data.frame':   46679 obs. of  18 variables:
 $ day1_1  : num  2711.2 0 42.8 15.5 0 ...
 $ day1_2  : num  1094.75 0 13.49 4.29 17.16 ...
 $ day1_3  : num  1331.588 0 21.921 0.562 3.935 ...
```

```
 $ day14_1 : num  728.67 0 16.24 1.16 6.96 ...
 $ day14_2 : num  603.85 0 10.8 0 2.16 ...
 $ day14_3 : num  646.12 0 33.84 0 1.06 ...
 $ day2_1  : num  1185.66 0 166.38 1.11 31.51 ...
 $ day2_2  : num  1239.5 0 23.7 0 0 ...
 $ day2_3  : num  829.6 0 137.9 5.56 43.37 ...
 $ day3_1  : num  928.11 0 48.9 0.94 8.46 ...
 $ day3_2  : num  1110.91 0 8.24 0 7.21 ...
 $ day3_3  : num  951.1 0 49.9 0 0 ...
 $ day7_1  : num  953.72 0 6.09 0 6.96 ...
 $ day7_2  : num  1008.027 0 6.811 0.973 99.246 ...
 $ day7_3  : num  1001.467 0 39.507 1.838 0.919 ...
 $ normal_1: num  719.03 0 10.42 1.49 47.64 ...
 $ normal_2: num  1247.3 0 7.45 0 6.77 ...
 $ normal_3: num  809.4 0 1.7 0 0 ...
```

```
str(fa_meta)
```

```
'data.frame':   18 obs. of  5 variables:
 $ dataType    : chr  "1 transcriptome" "2 transcriptome" "3 transcriptome" "4 transcriptome" ...
 $ sampleName  : chr  "day14_1" "day14_2" "day14_3" "day1_1" ...
 $ condition   : chr  "day14" "day14" "day14" "day1" ...
 $ sampleNumber: int  1 2 3 1 2 3 1 2 3 1 ...
 $ color       : chr  "#FF4400" "#FF4400" "#FF4400" "#BBD7FF" ...
```

```
str(pfa_expr)
```

```
'data.frame':   8044 obs. of  10 variables:
 $ normal_1: num  531.3 221.6 26.1 4363.1 879.3 ...
 $ normal_2: num  651.7 266.4 29.1 4784.1 1065.4 ...
 $ day1_1  : num  335.6 175.4 57.7 4064.5 914.3 ...
 $ day1_2  : num  334.8 159.4 45.8 3917.3 928.3 ...
 $ day2_1  : num  197.2 234.8 81.6 4599 1000.1 ...
 $ day2_2  : num  307.2 256.9 88.9 5957 1264.3 ...
 $ day7_1  : num  123.2 149.9 29.9 2806.5 738.4 ...
 $ day7_2  : num  272.6 315.1 44.6 6792.1 1362.1 ...
 $ day14_1 : num  93.7 110.6 13.6 2022.5 466.4 ...
 $ day14_2 : num  196.2 126.4 27.7 3226.8 714.6 ...
```

```
str(pfa_expr_norm)
```

```
'data.frame':   8044 obs. of  10 variables:
 $ normal_1: num  5.121 3.86 0.785 8.158 5.847 ...
 $ normal_2: num  5.033 3.743 0.565 7.908 5.742 ...
 $ day1_1  : num  4.51 3.57 1.97 8.1 5.95 ...
 $ day1_2  : num  4.49 3.42 1.63 8.04 5.96 ...
 $ day2_1  : num  3.63 3.88 2.36 8.17 5.97 ...
 $ day2_2  : num  3.93 3.67 2.15 8.21 5.97 ...
 $ day7_1  : num  3.53 3.81 1.49 8.04 6.11 ...
 $ day7_2  : num  3.69 3.89 1.08 8.32 6 ...
 $ day14_1 : num  3.665 3.904 0.895 8.095 5.979 ...
 $ day14_2 : num  4.03 3.39 1.21 8.06 5.89 ...
```

```
str(pfa_meta)
```

```
'data.frame':   12 obs. of  5 variables:
 $ dataType    : chr  "proteome" "proteome" "proteome" "proteome" ...
 $ sampleName  : chr  "normal_1" "normal_2" "day1_1" "day1_2" ...
 $ condition   : chr  "normal" "normal" "day1" "day1" ...
 $ sampleNumber: int  1 2 1 2 1 1 1 2 1 2 ...
 $ color       : chr  "#BBFFBB" "#BBFFBB" "#FFFFDD" "#FFFFDD" ...
```

Je supprime le groupe day3 du fichier metadata pfa

```
pfa_meta <- pfa_meta %>% filter(condition != "day3")
str(pfa_meta)
```

```
'data.frame':   10 obs. of  5 variables:
 $ dataType    : chr  "proteome" "proteome" "proteome" "proteome" ...
 $ sampleName  : chr  "normal_1" "normal_2" "day1_1" "day1_2" ...
 $ condition   : chr  "normal" "normal" "day1" "day1" ...
 $ sampleNumber: int  1 2 1 2 1 1 1 2 1 2
 $ color       : chr  "#BBFFBB" "#BBFFBB" "#FFFFDD" "#FFFFDD" ...
```

Les deux fichiers fa ne donnent pas les observations de l'échantillon dans le même ordre:

```
fa_meta$sampleName == names(fa_expr)
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TR
```

Donc je réorganise les échantillons dans l'ordre de l'expérience: condition normale, puis day 1 à 14 avec les 3 réplicats.

```
sample_order <- c(paste(rep(c("normal", "day1", "day2", "day3", "day7", "day14"), each = 3),
                        1:3, sep = "_"))

fa_expr <- fa_expr[,sample_order]
fa_meta <- fa_meta[match(sample_order, fa_meta$sampleName),]

fa_expr_norm <- fa_expr_norm[,sample_order]
fa_meta <- fa_meta[match(sample_order, fa_meta$sampleName),]
```

J'ai maintenant les deux jeux de données avec pour chaque un fichier metadata et une table de counts raw ou normalized:

- fa_expr
- fa_expr_norm
- fa_meta
- pfa_expr
- pfa_expr_norm
- pfa_meta

```
head(fa_expr)
```

```
                    normal_1   normal_2   normal_3     day1_1      day1_2      day1_3      day2_1
ENSMUSG00000000001 483.2298191 1842.14841 475.696800 2278.80022 1786.498848 2368.618959 2145.223455 262
ENSMUSG00000000003   0.0000000    0.00000   0.000000    0.00000    0.000000    0.000000    0.000000   0
ENSMUSG00000000028   7.3515305   11.19676   1.034465   36.27547   22.147861   39.484949  300.558779   4
ENSMUSG00000000031   0.8603067    0.00000   0.000000   13.18853    7.151932    1.115304    1.711944   0
ENSMUSG00000000037  32.0575944   10.42322   0.000000    0.00000   27.903214    6.897842   57.382077   0
ENSMUSG00000000049  27.7647071   38.42445  15.903837   30.86001    4.861367   51.466810   10.796186   2
```

```
head(fa_expr_norm)
```

```
                    normal_1    normal_2   normal_3     day1_1      day1_2       day1_3      day2_1
ENSMUSG00000000001 719.032852 1247.299971 809.375515 2711.22417 1094.745589 1331.5880610 1185.658698 123
ENSMUSG00000000003   0.000000    0.000000   0.000000    0.00000    0.000000    0.0000000    0.000000
ENSMUSG00000000028  10.420766    7.448588   1.700369   42.82759   13.485108   21.9214582  166.379146   1
ENSMUSG00000000031   1.488681    0.000000   0.000000   15.46552    4.290716    0.5620887    1.105509
ENSMUSG00000000037  47.637787    6.771444   0.000000    0.00000   17.162865    3.9346207   31.507014
ENSMUSG00000000049  41.683064   25.731487  27.205900   36.87931    3.064797   28.6665222    6.080301   1
```

```
fa_meta
```

```
           dataType sampleName condition sampleNumber   color
16 16 transcriptome   normal_1    normal            1 #BBFFBB
17 17 transcriptome   normal_2    normal            2 #BBFFBB
18 18 transcriptome   normal_3    normal            3 #BBFFBB
4   4 transcriptome     day1_1      day1            1 #BBD7FF
5   5 transcriptome     day1_2      day1            2 #BBD7FF
6   6 transcriptome     day1_3      day1            3 #BBD7FF
7   7 transcriptome     day2_1      day1            1 #F0BBFF
8   8 transcriptome     day2_2      day1            2 #F0BBFF
9   9 transcriptome     day2_3      day1            3 #F0BBFF
10 10 transcriptome     day3_1      day3            1 #FFFFDD
11 11 transcriptome     day3_2      day3            2 #FFFFDD
12 12 transcriptome     day3_3      day3            3 #FFFFDD
13 13 transcriptome     day7_1      day7            1 #FFDD88
14 14 transcriptome     day7_2      day7            2 #FFDD88
15 15 transcriptome     day7_3      day7            3 #FFDD88
1   1 transcriptome    day14_1     day14            1 #FF4400
2   2 transcriptome    day14_2     day14            2 #FF4400
3   3 transcriptome    day14_3     day14            3 #FF4400
```

```
head(pfa_expr)
```

```
                   normal_1  normal_2    day1_1    day1_2    day2_1    day2_2    day7_1    day7_2   day
ENSMUSG00000037686  531.2680  651.7200  335.5910  334.8460  197.1740  307.194  123.2060  272.6190   93.
ENSMUSG00000027831  221.6020  266.3590  175.4090  159.4190  234.8080  256.927  149.9380  315.0590  110.
ENSMUSG00000039201   26.0723   29.1331   57.7329   45.8475   81.6009   88.870   29.8560   44.5586   13.
ENSMUSG00000031095 4363.0500 4784.0800 4064.4800 3917.2900 4599.0300 5957.030 2806.5200 6792.0900 2022.
ENSMUSG00000034931  879.2790 1065.3900  914.2870  928.2760 1000.1000 1264.270  738.3520 1362.0700  466.
ENSMUSG00000038208   68.2225   89.8871   57.9041   76.3510   84.8474  105.245   72.8696  138.9810   40.
```

```
head(pfa_expr_norm)
```

```
                    normal_1  normal_2   day1_1   day1_2   day2_1   day2_2   day7_1   day7_2  day14_1  d
ENSMUSG00000037686 5.1206348 5.0327406 4.506491 4.492382 3.628542 3.932456 3.530692 3.685216 3.665305 4
ENSMUSG00000027831 3.8601167 3.7429066 3.571461 3.422865 3.880248 3.674968 3.813620 3.893704 3.903714 3
ENSMUSG00000039201 0.7849136 0.5645967 1.972258 1.630415 2.358439 2.146880 1.492104 1.081497 0.894504 1
ENSMUSG00000031095 8.1578622 7.9080330 8.103830 8.039705 8.170497 8.208356 8.038369 8.322345 8.095098 8
ENSMUSG00000034931 5.8472466 5.7415172 5.951775 5.962765 5.969607 5.972364 6.112220 6.004585 5.979016 5
ENSMUSG00000038208 2.1641269 2.1791889 1.976512 2.363160 2.414548 2.390031 2.774423 2.714986 2.468220 2
```

```
pfa_meta
```

```
   dataType sampleName condition sampleNumber   color
1  proteome   normal_1    normal            1 #BBFFBB
2  proteome   normal_2    normal            2 #BBFFBB
3  proteome     day1_1      day1            1 #FFFFDD
4  proteome     day1_2      day1            2 #FFFFDD
5  proteome     day2_1      day2            1 #BBD7FF
6  proteome     day2_2      day2            1 #BBD7FF
9  proteome     day7_1      day7            1 #FFDD88
10 proteome     day7_2      day7            2 #FFDD88
11 proteome    day14_1     day14            1 #FF4400
12 proteome    day14_2     day14            2 #FF4400
```

Analyses différentielles DESeq2

Modèle fa en comparant Day 7 à normal

```
#?DESeq
library("DESeq2")
```

Préparation objet **DESeq2**

```
# id <- pfa_expr [,1]
# pfa_expr <- pfa_expr[,-1]
# rownames(pfa_expr) <- make.names(id, unique = TRUE)

#any (fa_DataMatrix [,-1] < 0)

fa_DataMatrix <- as.matrix(fa_expr)

dds <- DESeqDataSetFromMatrix(countData = round(fa_DataMatrix), colData = fa_meta, design = ~ condition)
dds
```

```
class: DESeqDataSet
dim: 46679 18
metadata(1): version
assays(1): counts
rownames(46679): ENSMUSG00000000001 ENSMUSG00000000003 ... ENSMUSG00000109577 ENSMUSG00000109578
```

```
rowData names(0):
colnames(18): normal_1 normal_2 ... day14_2 day14_3
colData names(5): dataType sampleName condition sampleNumber color
```

Run fonction DESeq2

```
dds <- DESeq(dds)
```

Table de résultats du DESeq2 entre les groupes **normal** et **day7**

```
res <- results(dds, contrast=c("condition", "normal", "day7"))
head(res)
```

```
log2 fold change (MLE): condition normal vs day7
Wald test p-value: condition normal vs day7
DataFrame with 6 rows and 6 columns
                    baseMean log2FoldChange     lfcSE      stat    pvalue      padj
                   <numeric>      <numeric> <numeric> <numeric> <numeric> <numeric>
ENSMUSG00000000001 1061.10739     -0.0926767  0.371089 -0.249742  0.802787  0.936171
ENSMUSG00000000003    0.00000             NA        NA        NA        NA        NA
ENSMUSG00000000028   35.89103     -1.3988787  1.037433 -1.348404  0.177528  0.509671
ENSMUSG00000000031    1.85467     -1.0447828  2.258700 -0.462559  0.643680        NA
ENSMUSG00000000037   15.74259     -0.9804598  1.674099 -0.585664  0.558101  0.828730
ENSMUSG00000000049   20.97970      0.5186876  1.122010  0.462284  0.643877  0.872279
```

Combien de gènes sont **significatifs** $>$ **0.05**

```
table(res$padj < 0.05)
```

```
FALSE  TRUE
20701  1764
```

Je classe la table par **ordre décroissant de padj**

```
orderedRes <- res[ order(res$padj), ]
head(orderedRes)
```

```
log2 fold change (MLE): condition normal vs day7
Wald test p-value: condition normal vs day7
DataFrame with 6 rows and 6 columns
                    baseMean log2FoldChange     lfcSE      stat      pvalue        padj
                   <numeric>      <numeric> <numeric> <numeric>   <numeric>   <numeric>
ENSMUSG00000024164  14670.159       -8.19459  0.593794 -13.80039 2.53467e-43 5.69414e-39
ENSMUSG00000029304 224391.951       -5.14937  0.480753 -10.71105 9.03342e-27 9.62642e-23
ENSMUSG00000061947    621.939       -4.85093  0.454278 -10.67834 1.28552e-26 9.62642e-23
ENSMUSG00000027962   2261.581       -5.22527  0.503833 -10.37103 3.35873e-25 1.88635e-21
ENSMUSG00000003617   5729.876       -3.80553  0.383311  -9.92805 3.14337e-23 1.41232e-19
ENSMUSG00000066071   3575.949        2.88208  0.301427   9.56143 1.16138e-21 4.34841e-18
```

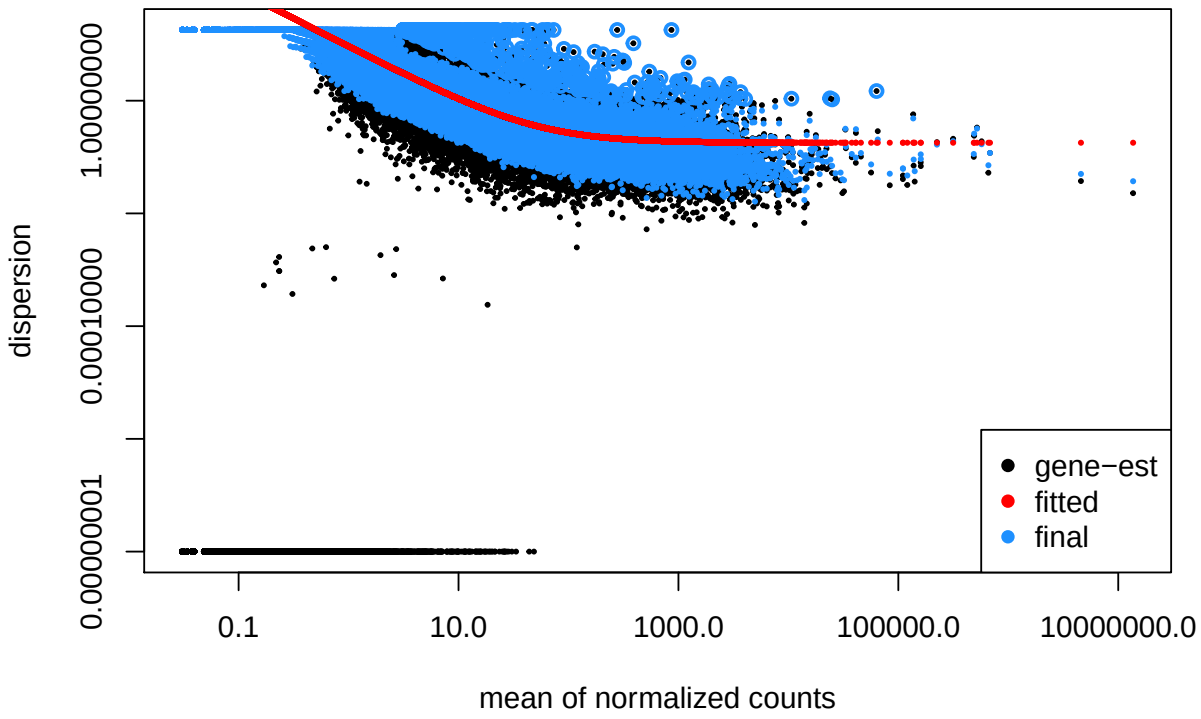
Je **normalise** la table

```
normCounts <- counts(dds, normalized = TRUE)
head(normCounts)
```

	normal_1	normal_2	normal_3	day1_1	day1_2	day1_3	day2_1	
ENSMUSG00000000001	719.032852	1247.299971	809.375515	2711.22417	1094.745589	1331.5880610	1185.658698	12
ENSMUSG00000000003	0.000000	0.000000	0.000000	0.00000	0.000000	0.0000000	0.000000	
ENSMUSG00000000028	10.420766	7.448588	1.700369	42.82759	13.485108	21.9214582	166.379146	
ENSMUSG00000000031	1.488681	0.000000	0.000000	15.46552	4.290716	0.5620887	1.105509	
ENSMUSG00000000037	47.637787	6.771444	0.000000	0.00000	17.162865	3.9346207	31.507014	
ENSMUSG00000000049	41.683064	25.731487	27.205900	36.87931	3.064797	28.6665222	6.080301	

Visualisation des **estimations de dispersion** de DESeq2

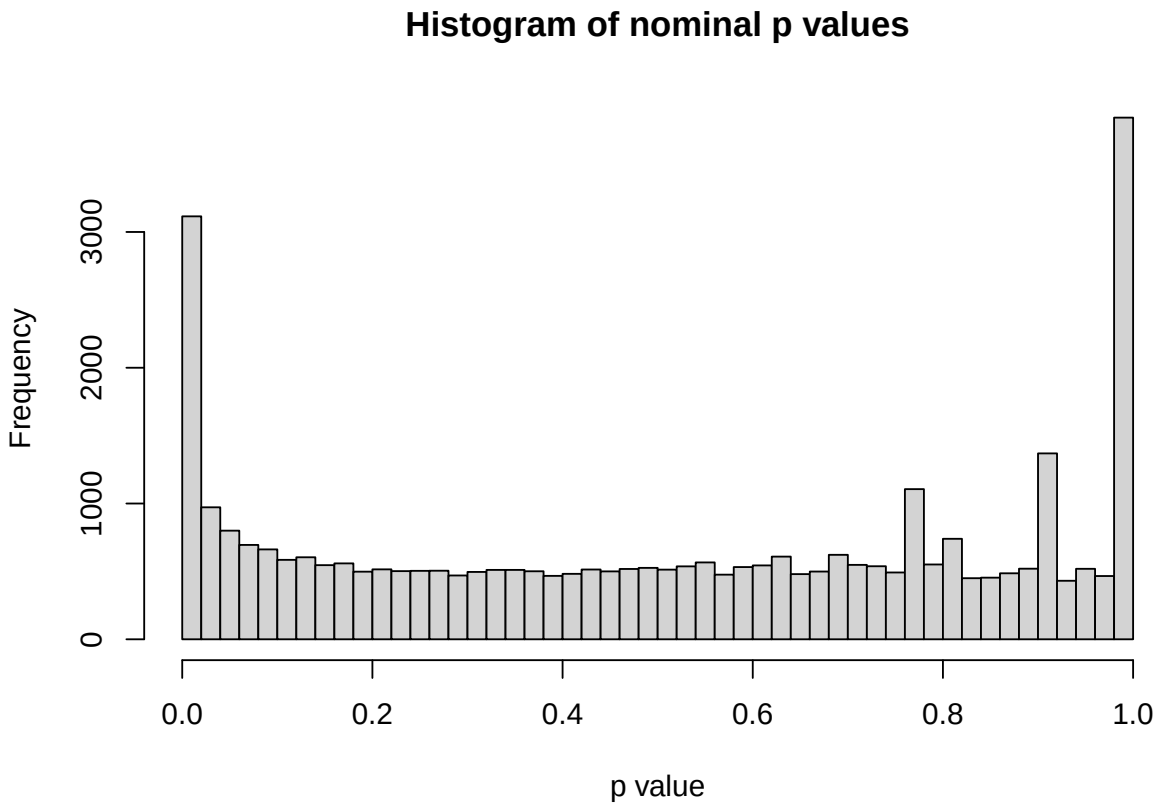
```
plotDispEsts(dds)
```



- Les points noirs sont la représentation “brute” des gènes (forte variabilité)
- La ligne de tendance rouge (dépendance des dispersions à la moyenne)
- Les points bleus représentent l’estimation de chaque gène vers la ligne rouge
- Les cercles bleus au-dessus du «nuage» principal représentent des gènes avec des valeurs aberrantes de dispersion.

Distribution des **pvalues**

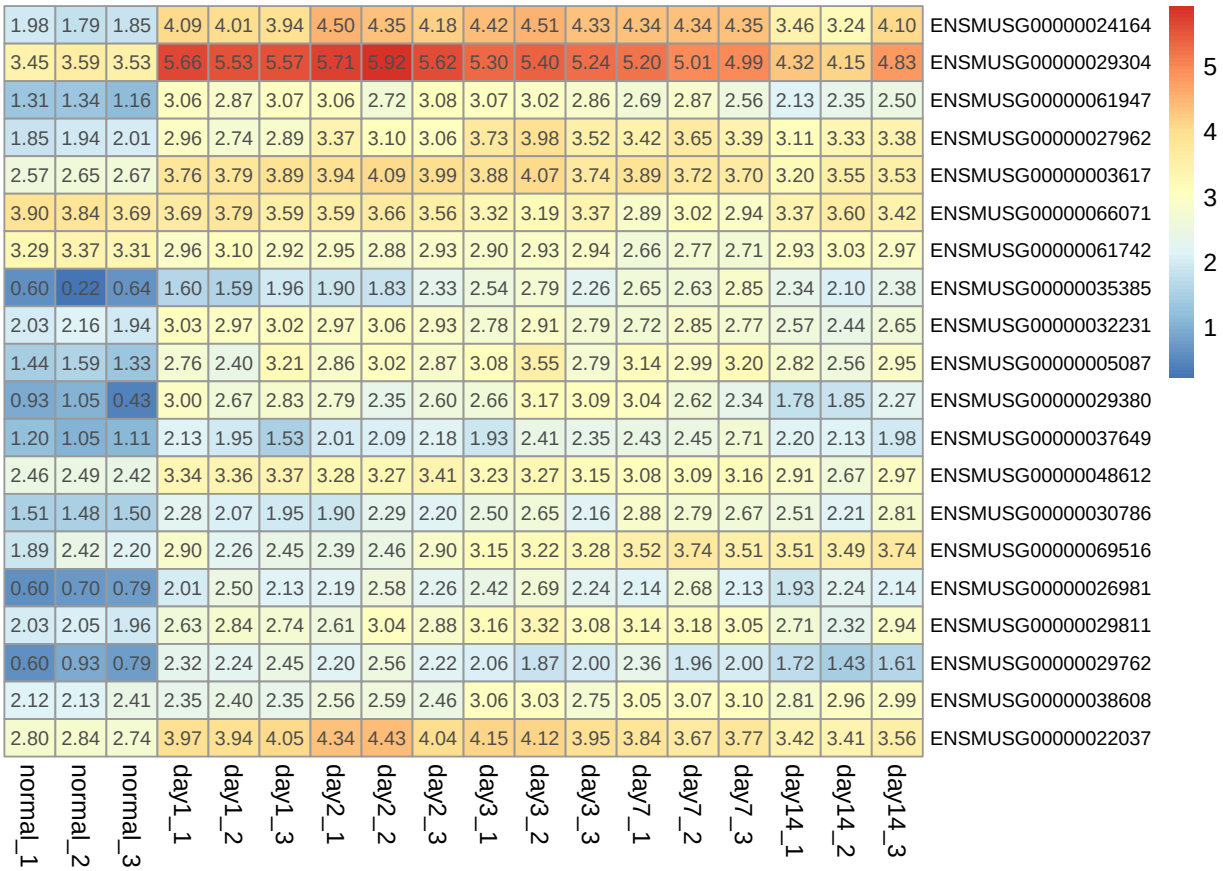
```
hist(orderedRes$pvalue, breaks=0:50/50, xlab="p value", main="Histogram of nominal p values")
```



Heatmap des **20 gènes les plus différentiellement exprimés**.

```
library(pheatmap)

# select the 20 most differentially expressed genes
select <- row.names(orderedRes[1:20, ])

# transform the counts to log10
log10_normCounts <- log10(normCounts + 1)

# get the values for the selected genes
values <- log10_normCounts[ select, ]

pheatmap(values,
         scale = "none",
         cluster_rows = FALSE,
         cluster_cols = FALSE,
         fontsize_row = 8,
         annotation_names_col = FALSE,
```

```
        #gaps_col = c(3,6),
        display_numbers = TRUE,
        number_format = "%.2f",
        height=12,
        width=6)
```



Modèle pfa en comparant Day 7 à normal

Préparation objet DESeq2

```
# id <- pfa_expr [,1]
# pfa_expr <- pfa_expr[,-1]
# rownames(pfa_expr) <- make.names(id, unique = TRUE)

#any (fa_DataMatrix [,-1] < 0)

pfa_DataMatrix <- as.matrix(pfa_expr)

dds <- DESeqDataSetFromMatrix(countData = round(pfa_DataMatrix), colData = pfa_meta, design = ~ conditi
dds
```

```
class: DESeqDataSet
```

```
dim: 8044 10
metadata(1): version
assays(1): counts
rownames(8044): ENSMUSG00000037686 ENSMUSG00000027831 ... ENSMUSG00000027523.1 ENSMUSG00000020741.1
rowData names(0):
colnames(10): 1 2 ... 11 12
colData names(5): dataType sampleName condition sampleNumber color
```

Run fonction DESeq2

```
dds <- DESeq(dds)
```

Table de résultats du DESeq2 entre les groupes **normal** et **day7**

```
res <- results(dds, contrast=c("condition", "normal", "day7"))
head(res)
```

```
log2 fold change (MLE): condition normal vs day7
Wald test p-value: condition normal vs day7
DataFrame with 6 rows and 6 columns
                    baseMean log2FoldChange     lfcSE      stat                 pvalue
                   <numeric>      <numeric> <numeric> <numeric>              <numeric>
ENSMUSG00000037686  283.1512      1.4928568  0.191451  7.797601 0.0000000000000630949
ENSMUSG00000027831  193.2636     -0.0228946  0.217834 -0.105101 0.9162955118721628489
ENSMUSG00000039201   42.2126     -0.5914299  0.411140 -1.438513 0.1502886183136471487
ENSMUSG00000031095 4050.7644     -0.1184521  0.129794 -0.912619 0.3614429221883942999
ENSMUSG00000034931  900.1919     -0.2292681  0.104342 -2.197286 0.0280000392543649085
ENSMUSG00000038208   76.4645     -0.5416567  0.276803 -1.956830 0.0503675109798810972
                                                                                  <num
ENSMUSG00000037686 0.000000000000063
ENSMUSG00000027831 0.937044312218241
ENSMUSG00000039201 0.215955992446405
ENSMUSG00000031095 0.450766955981929
ENSMUSG00000034931 0.050410057779230
ENSMUSG00000038208 0.083624552632174
```

Combien de gènes sont **significatifs** $>$ **0.05**

```
table(res$padj < 0.05)
```

```
FALSE  TRUE
 3582  4462
```

Je classe la table par **ordre décroissant de padj**

```
orderedRes <- res[ order(res$padj), ]
head(orderedRes)
```

```
log2 fold change (MLE): condition normal vs day7
Wald test p-value: condition normal vs day7
DataFrame with 6 rows and 6 columns
                    baseMean log2FoldChange     lfcSE      stat      pvalue        padj
                   <numeric>      <numeric> <numeric> <numeric>   <numeric>   <numeric>
ENSMUSG00000071715  1097.288       -2.96982  0.157649  -18.8382 3.67186e-79 1.47682e-75
ENSMUSG00000024757   963.642        2.91933  0.154778   18.8614 2.36703e-79 1.47682e-75
ENSMUSG00000053303   626.809        3.02825  0.163434   18.5290 1.20598e-76 3.23363e-73
ENSMUSG00000039519  2155.892        2.77257  0.153107   18.1087 2.72021e-73 5.47034e-70
ENSMUSG00000066097   932.324        2.61823  0.145835   17.9534 4.51718e-72 7.26724e-69
ENSMUSG00000032047 13304.389        1.93889  0.109767   17.6637 7.98453e-70 1.07046e-66
```

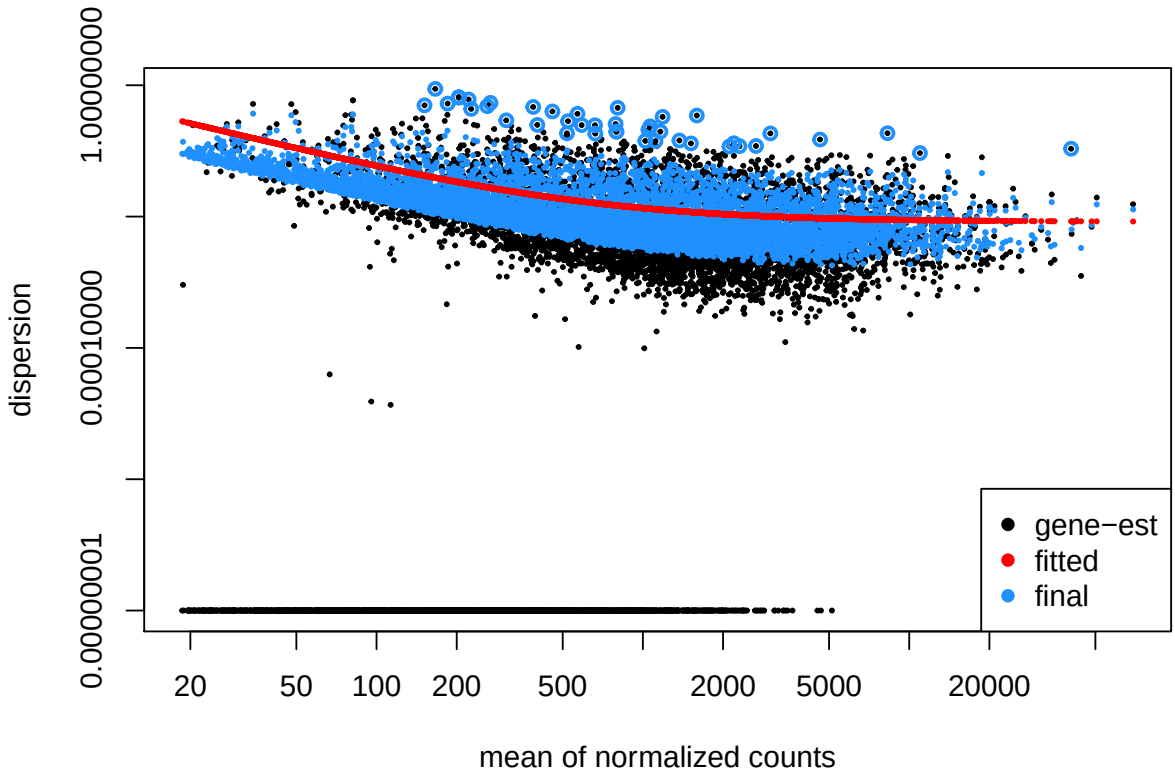
Je **normalise** la table

```
normCounts <- counts(dds, normalized = TRUE)
head(normCounts)
```

	1	2	3	4	5	6	9	
ENSMUSG00000037686	510.05841	481.36921	327.9377	325.17673	181.56837	226.98770	165.17223	185.040
ENSMUSG00000027831	213.24476	196.38683	170.8009	154.33761	216.59171	190.01902	201.42954	213.508
ENSMUSG00000039201	24.97461	21.41059	56.6083	44.65113	75.57668	65.80425	40.28591	30.501
ENSMUSG00000031095	4190.93192	3532.00967	3966.4851	3802.14103	4238.74590	4404.44867	3769.41820	4603.656
ENSMUSG00000034931	844.33398	786.28560	892.0687	900.78807	921.66686	934.56826	991.03336	923.171
ENSMUSG00000038208	65.31821	66.44667	56.6083	73.77144	78.34168	77.63423	98.02904	94.215

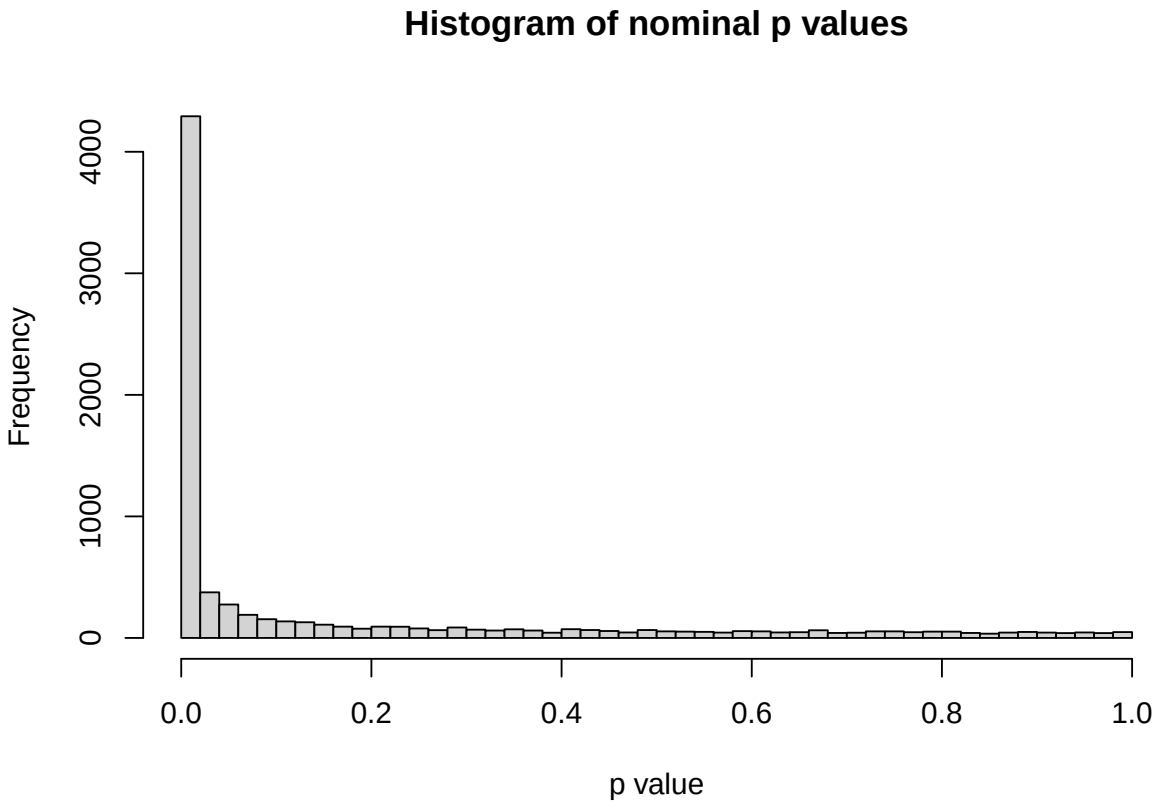
Visualisation des **estimations de dispersion** de DESeq2

```
plotDispEsts(dds)
```



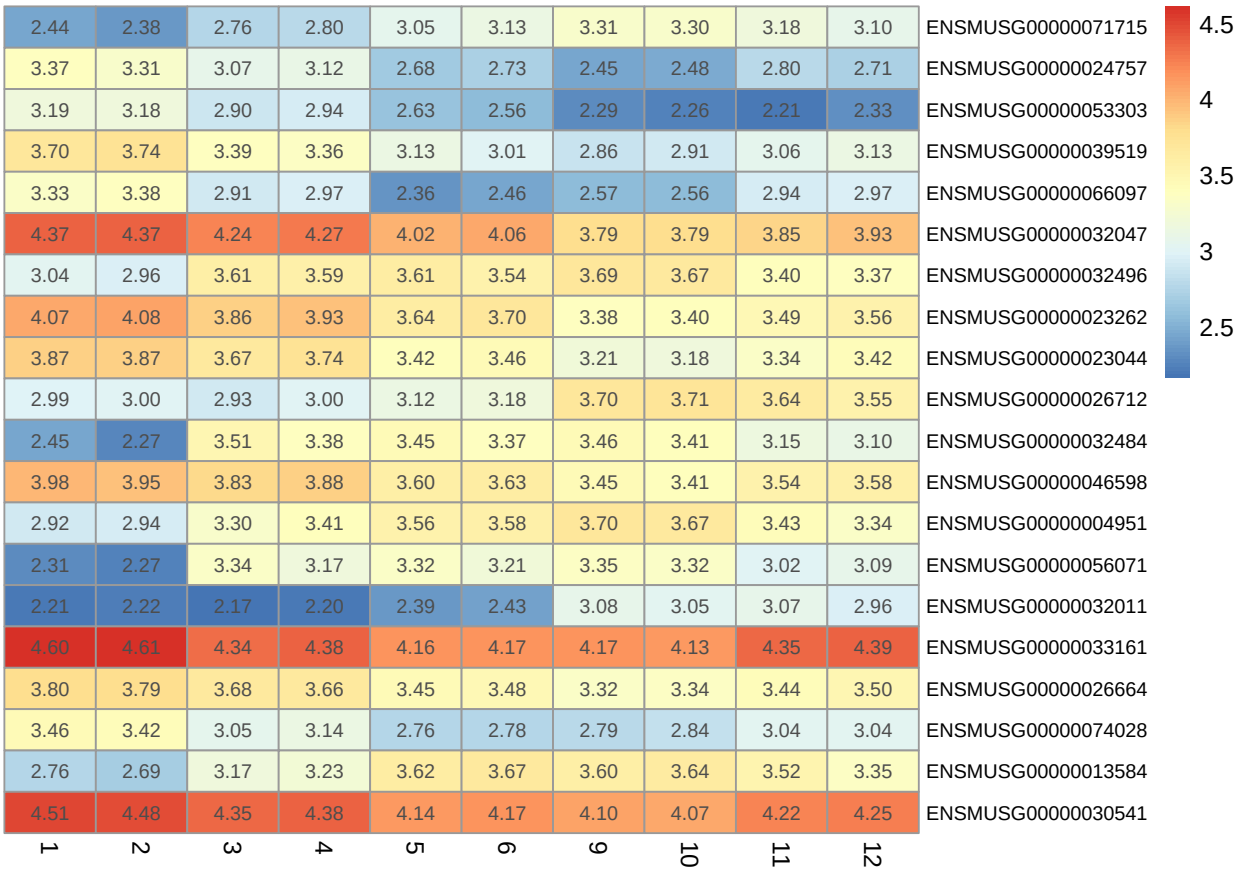
Distribution des **pvalues**

```
hist(orderedRes$pvalue, breaks=0:50/50, xlab="p value", main="Histogram of nominal p values")
```



Heatmap des **20 gènes les plus différentiellement exprimés**.

```
# select the 20 most differentially expressed genes
select <- row.names(orderedRes[1:20, ])

# transform the counts to log10
log10_normCounts <- log10(normCounts + 1)

# get the values for the selected genes
values <- log10_normCounts[ select, ]

pheatmap(values,
         scale = "none",
         cluster_rows = FALSE,
         cluster_cols = FALSE,
         fontsize_row = 8,
         annotation_names_col = FALSE,
         #gaps_col = c(3,6),
         display_numbers = TRUE,
         number_format = "%.2f",
         height=12,
         width=6)
```



MixOmics

Préparation des tables

Il y a 18 échantillons pour l'expérience fa et 10 pour l'expérience pfa, il faut donc supprimer les 6 échantillons supplémentaire de la table fa.

Il faut également uniformiser la table metadata en conséquence.

```
head(fa_expr_norm)
```

```
                    normal_1    normal_2   normal_3     day1_1      day1_2       day1_3      day2_1
ENSMUSG00000000001 719.032852 1247.299971 809.375515 2711.22417 1094.745589 1331.5880610 1185.658698 123
ENSMUSG00000000003   0.000000    0.000000   0.000000    0.00000    0.000000    0.0000000    0.000000
ENSMUSG00000000028  10.420766    7.448588   1.700369   42.82759   13.485108   21.9214582  166.379146   1
ENSMUSG00000000031   1.488681    0.000000   0.000000   15.46552    4.290716    0.5620887    1.105509
ENSMUSG00000000037  47.637787    6.771444   0.000000    0.00000   17.162865    3.9346207   31.507014
ENSMUSG00000000049  41.683064   25.731487  27.205900   36.87931    3.064797   28.6665222    6.080301   1
```

```
fa <- fa_expr_norm[,- c(3,6,9,10,11,12,15,18)]
head(fa)
```

```
                    normal_1    normal_2     day1_1      day1_2      day2_1     day2_2     day7_1
```

```
ENSMUSG00000000001 719.032852 1247.299971 2711.22417 1094.745589 1185.658698 1239.4695 953.722863 1008.
ENSMUSG00000000003   0.000000    0.000000    0.00000    0.000000    0.000000    0.0000   0.000000    0.
ENSMUSG00000000028  10.420766    7.448588   42.82759   13.485108  166.379146   23.6540   6.091296    6.
ENSMUSG00000000031   1.488681    0.000000   15.46552    4.290716    1.105509    0.0000   0.000000    0.
ENSMUSG00000000037  47.637787    6.771444    0.00000   17.162865   31.507014    0.0000   6.961481   99.
ENSMUSG00000000049  41.683064   25.731487   36.87931    3.064797    6.080301   14.1924  27.845923    6.
```

```
pfa <- pfa_expr_norm
head(pfa)
```

```
                    normal_1  normal_2   day1_1   day1_2   day2_1   day2_2   day7_1   day7_2  day14_1  d
ENSMUSG00000037686 5.1206348 5.0327406 4.506491 4.492382 3.628542 3.932456 3.530692 3.685216 3.665305 4
ENSMUSG00000027831 3.8601167 3.7429066 3.571461 3.422865 3.880248 3.674968 3.813620 3.893704 3.903714 3
ENSMUSG00000039201 0.7849136 0.5645967 1.972258 1.630415 2.358439 2.146880 1.492104 1.081497 0.894504 1
ENSMUSG00000031095 8.1578622 7.9080330 8.103830 8.039705 8.170497 8.208356 8.038369 8.322345 8.095098 8
ENSMUSG00000034931 5.8472466 5.7415172 5.951775 5.962765 5.969607 5.972364 6.112220 6.004585 5.979016 5
ENSMUSG00000038208 2.1641269 2.1791889 1.976512 2.363160 2.414548 2.390031 2.774423 2.714986 2.468220 2
```

```
metadata <- pfa_meta[,- 1]
metadata
```

```
   sampleName condition sampleNumber   color
1    normal_1    normal            1 #BBFFBB
2    normal_2    normal            2 #BBFFBB
3      day1_1      day1            1 #FFFFDD
4      day1_2      day1            2 #FFFFDD
5      day2_1      day2            1 #BBD7FF
6      day2_2      day2            1 #BBD7FF
9      day7_1      day7            1 #FFDD88
10     day7_2      day7            2 #FFDD88
11    day14_1     day14            1 #FF4400
12    day14_2     day14            2 #FF4400
```

```
#c(normal_1,normal_2,day1_1,day1_2,day2_1,day2_2,day7_1,day7_2,day14_1,day14_2)

## Classer suivant ordre colonne Name de metadata
#Je retire le rawnames original (les chiffres 1,2,3...)
rownames(metadata) <- NULL
metadata
```

```
   sampleName condition sampleNumber   color
1    normal_1    normal            1 #BBFFBB
2    normal_2    normal            2 #BBFFBB
3      day1_1      day1            1 #FFFFDD
4      day1_2      day1            2 #FFFFDD
5      day2_1      day2            1 #BBD7FF
6      day2_2      day2            1 #BBD7FF
7      day7_1      day7            1 #FFDD88
8      day7_2      day7            2 #FFDD88
9     day14_1     day14            1 #FF4400
10    day14_2     day14            2 #FF4400
```

```
# add the rownames as a proper column
library(tibble)
metadata <- column_to_rownames(metadata, var = "sampleName")


# ## Ordre des echantillons de data table idem metadata table
# fa <- fa[, row.names(metadata)]
# head(fa)
#
# pfa_expr_norm <- pfa_expr_norm[, row.names(metadata)]
# head(pfa_expr_norm)
#
# metadata
rownames(metadata)
```

```
 [1] "normal_1" "normal_2" "day1_1"   "day1_2"   "day2_1"   "day2_2"   "day7_1"   "day7_2"   "day14_1"
```

```
colnames(fa)
```

```
 [1] "normal_1" "normal_2" "day1_1"   "day1_2"   "day2_1"   "day2_2"   "day7_1"   "day7_2"   "day14_1"
```

```
colnames(pfa)
```

```
 [1] "normal_1" "normal_2" "day1_1"   "day1_2"   "day2_1"   "day2_2"   "day7_1"   "day7_2"   "day14_1"
```

Filtres

Je regarde combien il y a de lignes avec des zéros dans la table fa et je les supprime et je filtre les genes où il y au moins de 3 échantillons avec des counts supérieurs ou égaux à 5

```
dim(fa)
```

```
[1] 46679    10
```

```
#voir le nombre de lignes avec zero counts
rs <- rowSums(fa)
nbgenes_at_zeros <- length(which(rs==0))
nbgenes_at_zeros
```

```
[1] 13269
```

```
#Je supprime les lignes avec zero counts
fa <- fa[rowSums(fa[, -1])>0, ]
#Je ne garde que les genes où il y a moins de 3 échantillons avec des counts supérieurs ou égaux à 5.
fa <- fa[rowSums((fa[, -1])>=5)>= 3 , ]
dim(fa)
```

```
[1] 22459    10
```

Idem pour la table pfa

```
dim(pfa)
```

```
[1] 8044   10
```

```
rs <- rowSums(pfa)
nbgenes_at_zeros <- length(which(rs==0))
nbgenes_at_zeros
```

```
[1] 0
```

```
pfa <- pfa[rowSums((pfa[, -1])>=5)>= 3 , ]

dim(pfa)
```

```
[1] 5277   10
```

Il faut également transposer les counts tables

```
fa_t <- t(fa)
str(fa_t)
```

```
 num [1:10, 1:22459] 719 1247 2711 1095 1186 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:10] "normal_1" "normal_2" "day1_1" "day1_2" ...
  ..$ : chr [1:22459] "ENSMUSG00000000001" "ENSMUSG00000000028" "ENSMUSG00000000037" "ENSMUSG0000000004
```

```
dim(fa_t)
```

```
[1]    10 22459
```

```
class(fa_t)
```

```
[1] "matrix" "array"
```

```
pfa_t <- t(pfa_expr_norm)
str(pfa_t)
```

```
 num [1:10, 1:8044] 5.12 5.03 4.51 4.49 3.63 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:10] "normal_1" "normal_2" "day1_1" "day1_2" ...
  ..$ : chr [1:8044] "ENSMUSG00000037686" "ENSMUSG00000027831" "ENSMUSG00000039201" "ENSMUSG0000003109
```

```
dim(pfa_t)
```

```
[1]   10 8044
```

```
class(pfa_t)
```

```
[1] "matrix" "array"
```

```
dim(metadata)
```

```
[1] 10  3
```

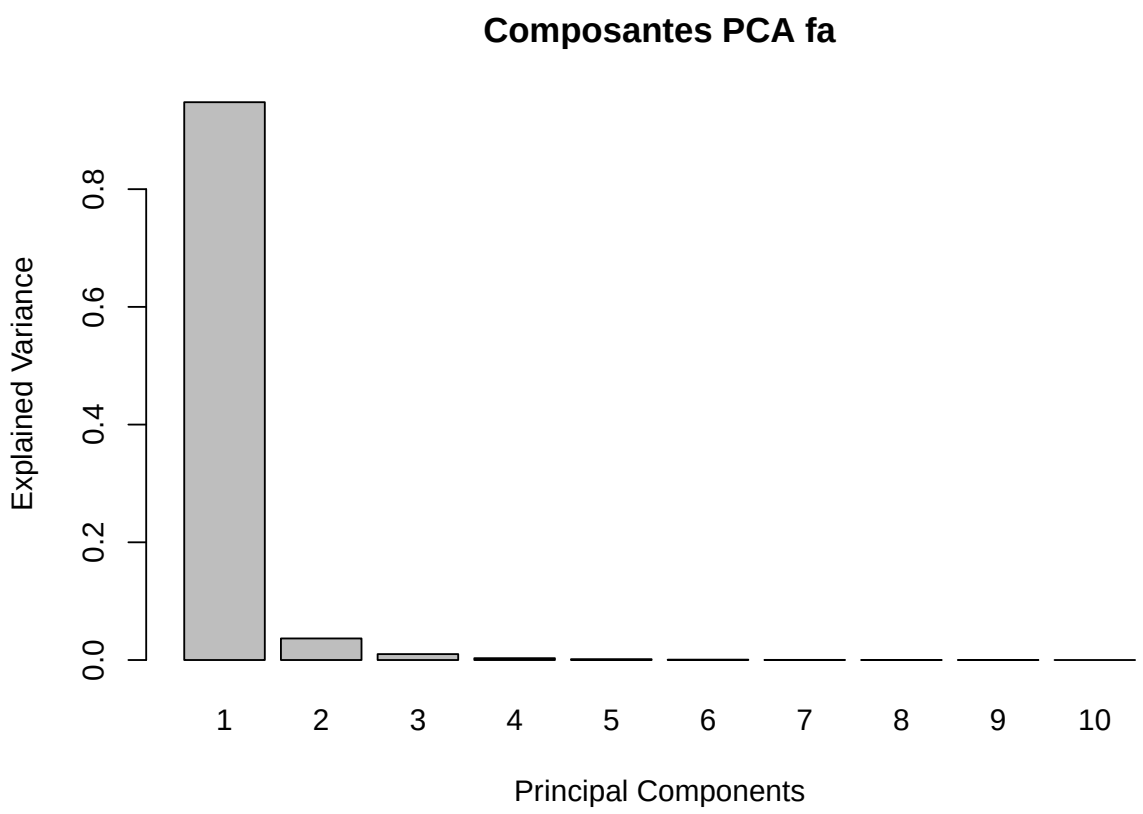
J'ai donc maintenant les deux counts tables avec 10 échantillons

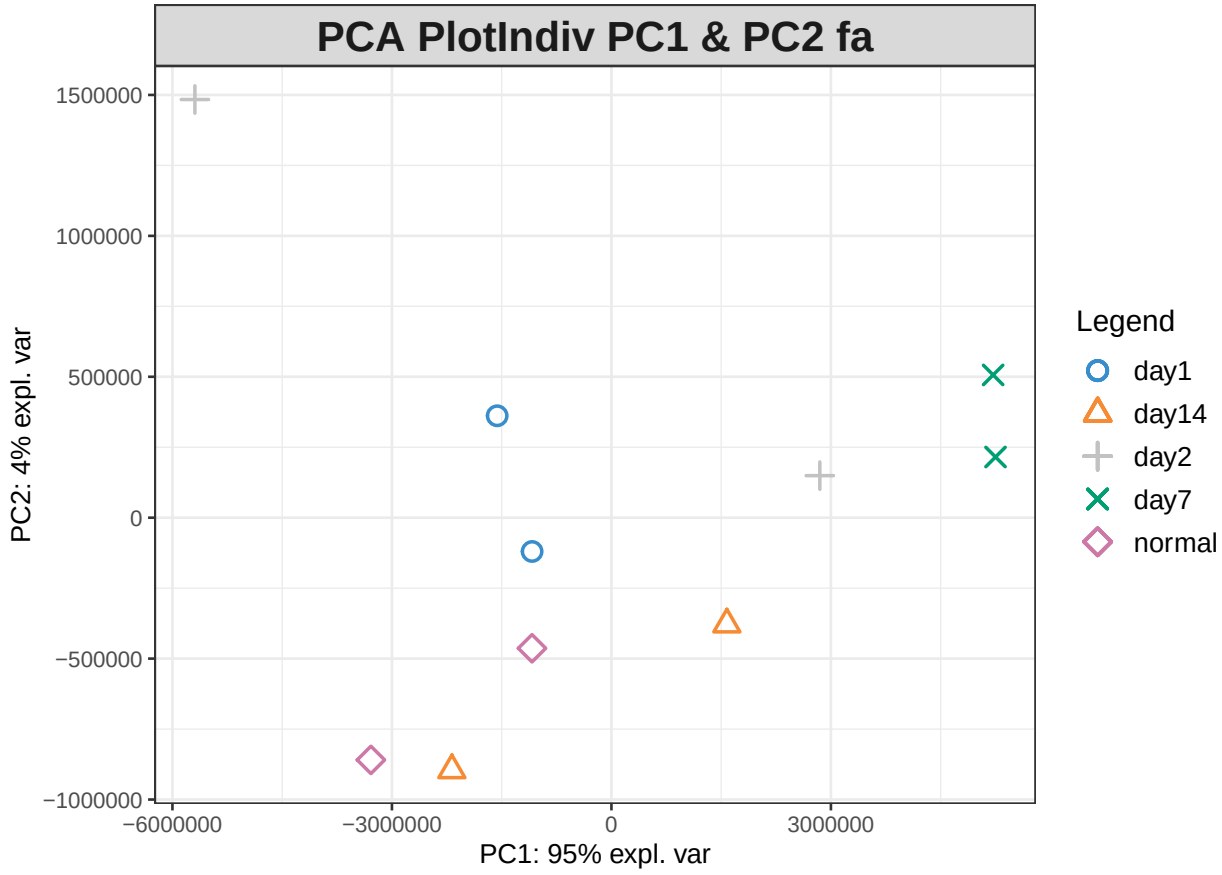
Import package

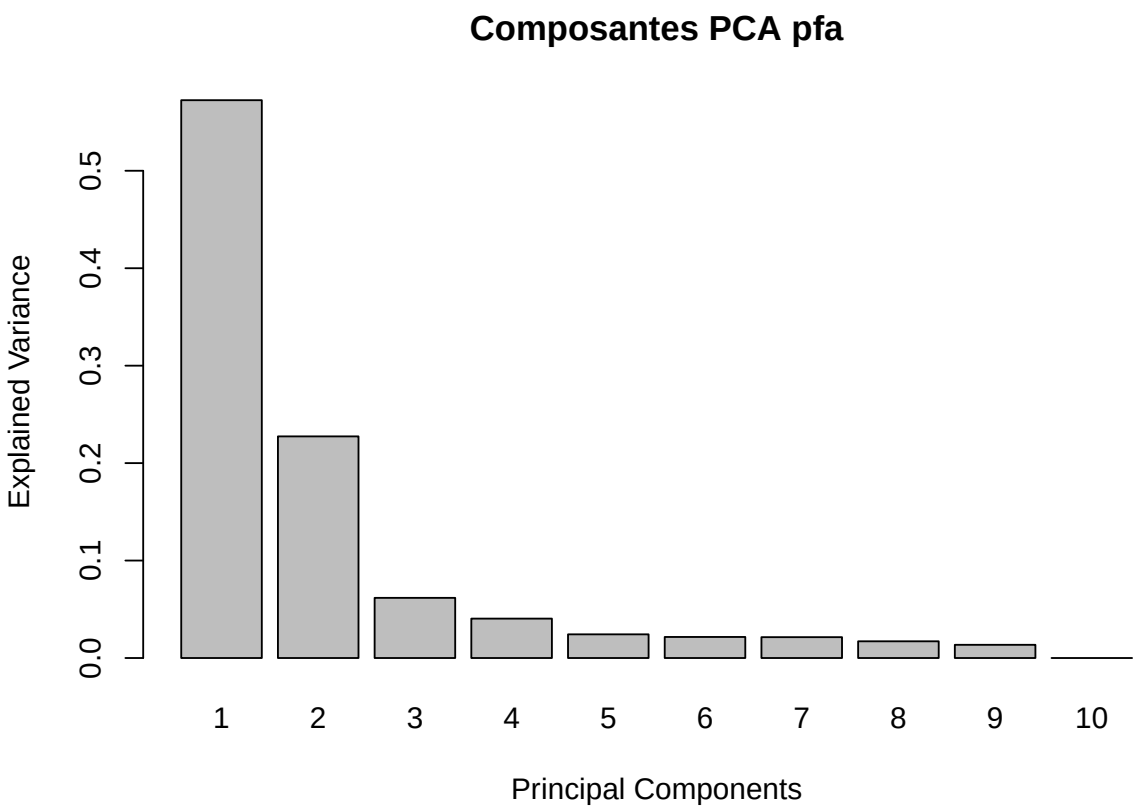
```
library(mixOmics)
```

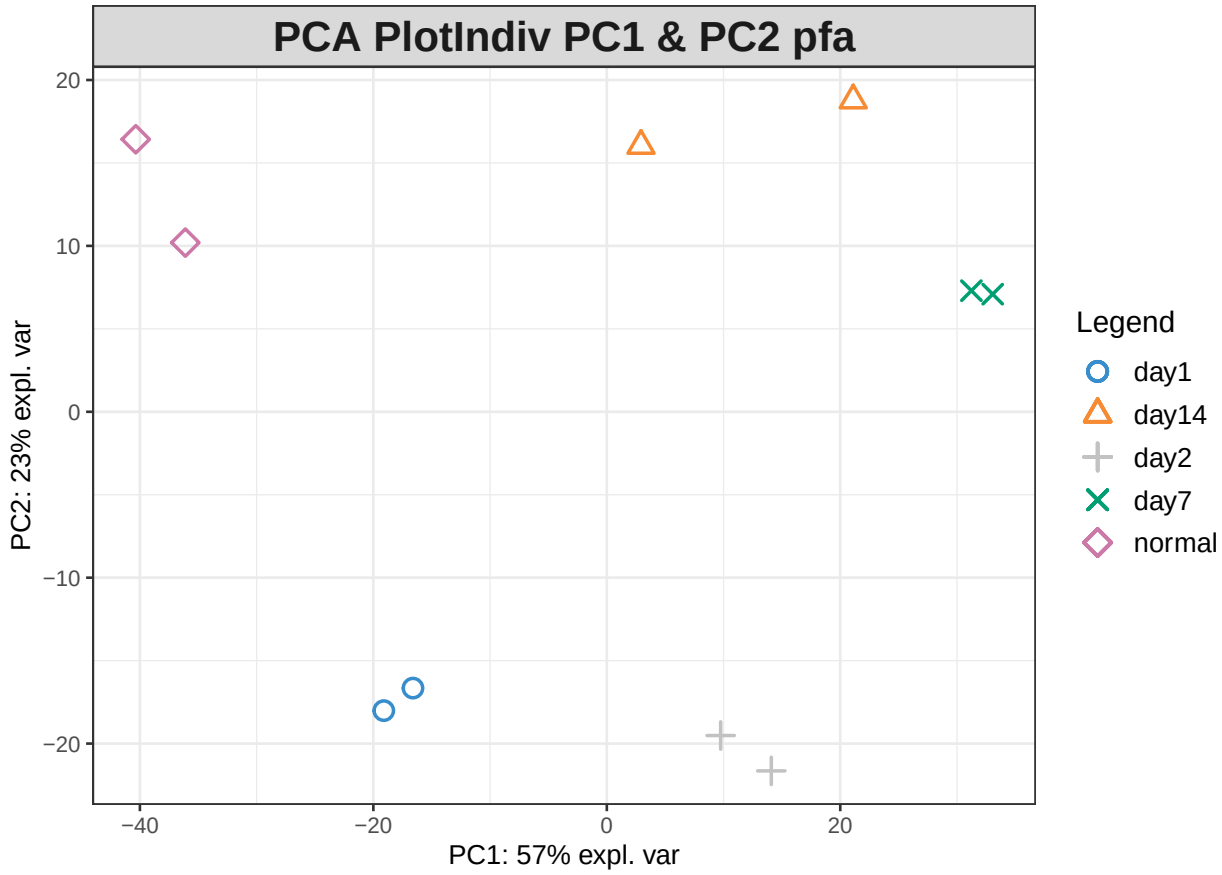
Analyses uni omic

PCA



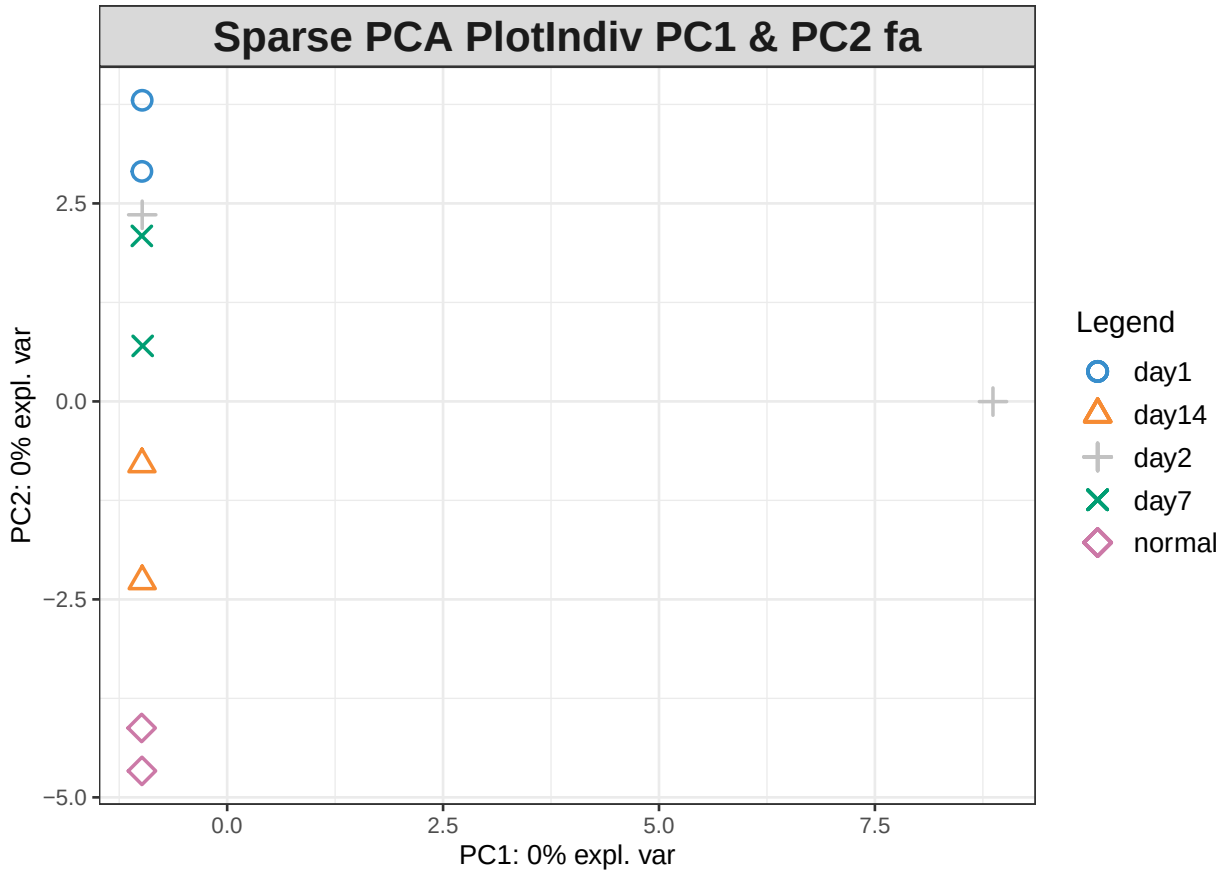

PCA PlotIndiv PC1 & PC2 fa
1500000
1000000
500000
0
−500000
−1000000
PC2: 4% expl. var
−6000000
−3000000
0
3000000
PC1: 95% expl. var
Legend
day1
day14
day2
day7
normal


Composantes PCA pfa
Explained Variance
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
6
7
8
9
10
Principal Components

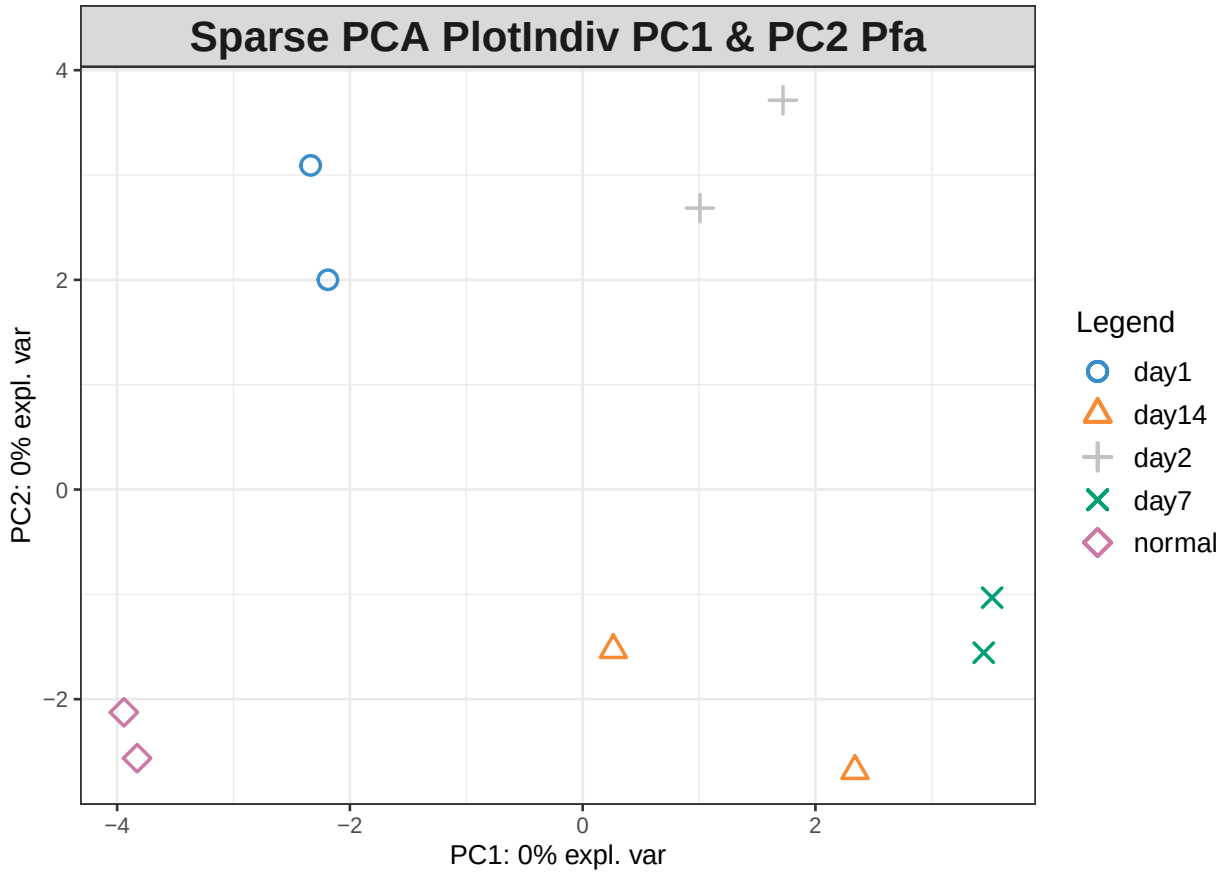


Sparse PCA

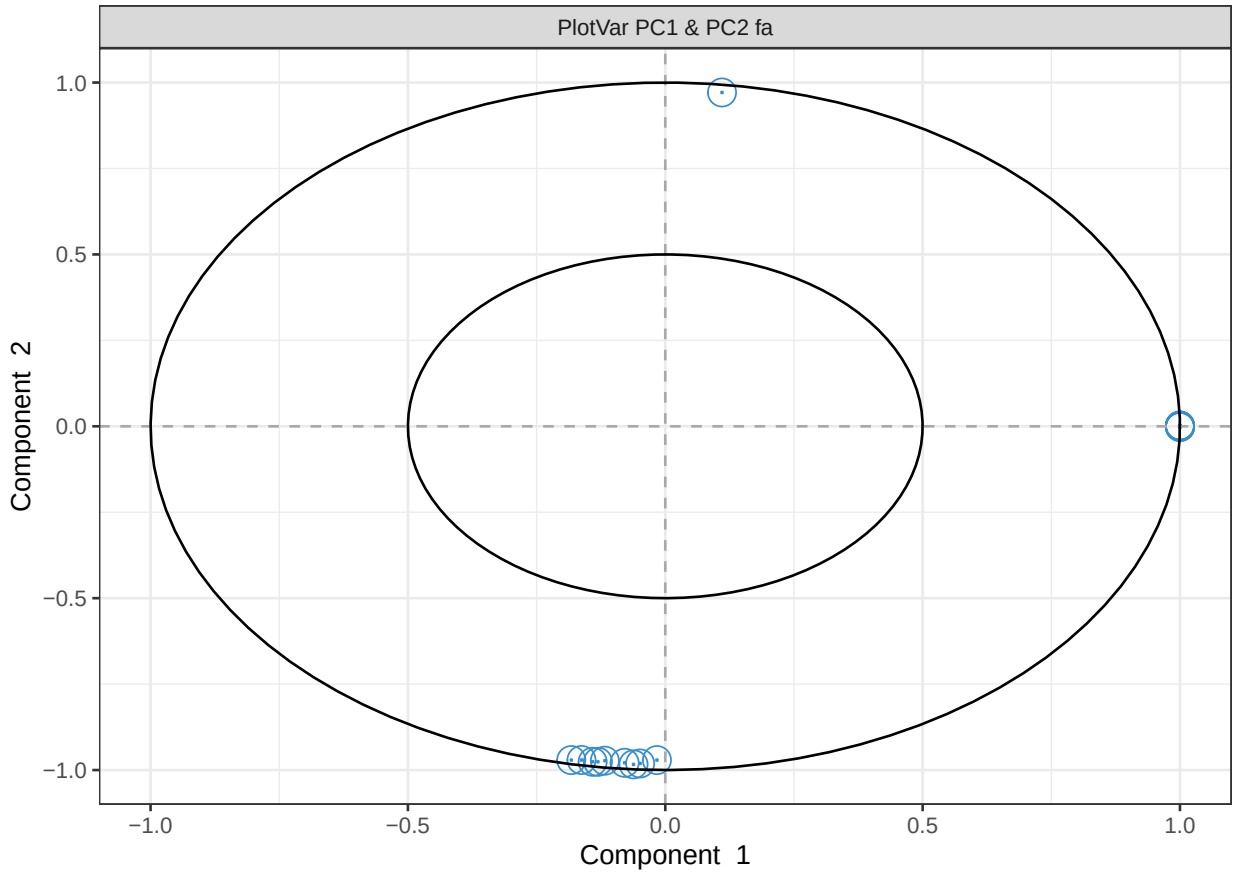
```
spca.fa_t <- spca(fa_t, ncomp=3, keepX=c(10,10,10))
spca.pfa_t <- spca(pfa_t, ncomp=3, keepX=c(10,10,10))
#?spca

plotIndiv(spca.fa_t, group = metadata$condition, comp= c(1,2), ind.names = FALSE, legend=TRUE,title = "
```



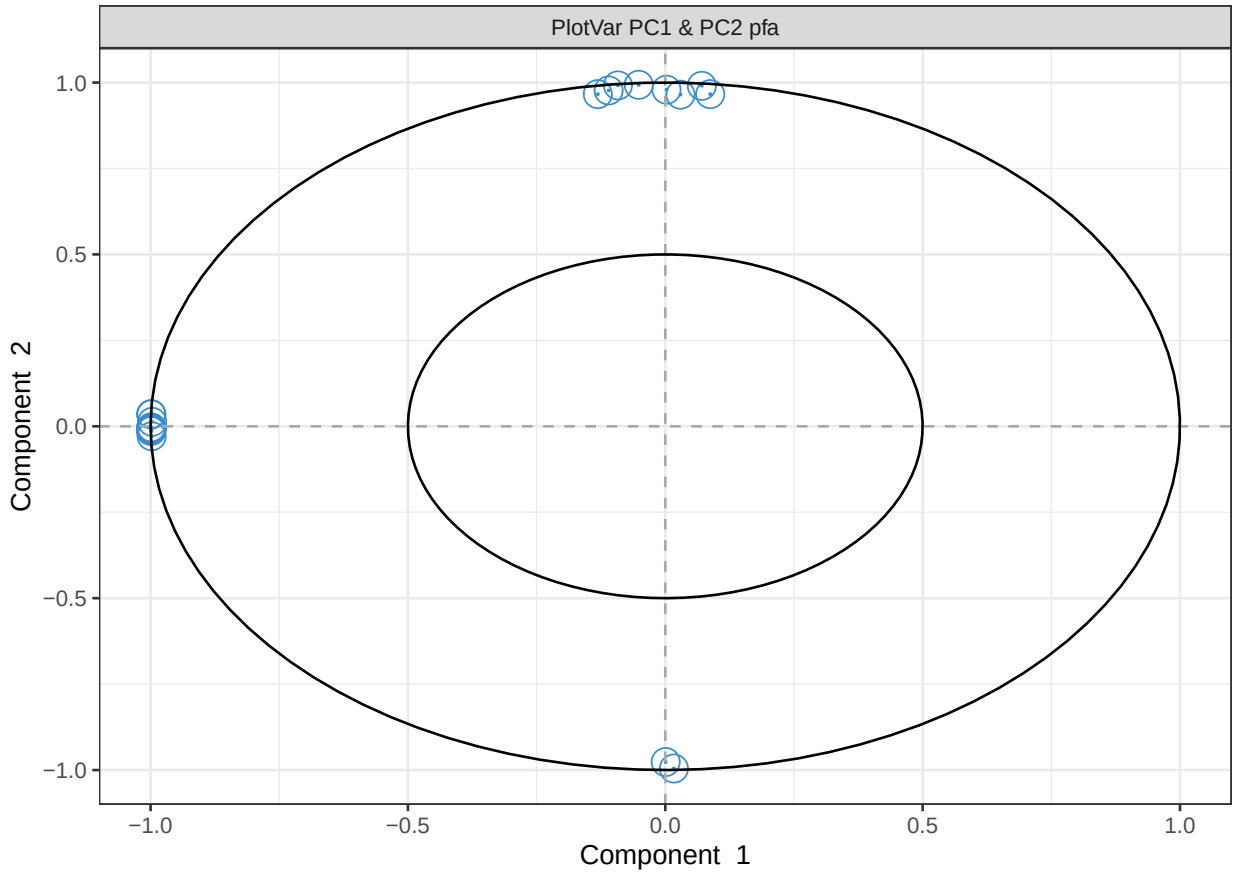

```
plotIndiv(spca.pfa_t, group = metadata$condition, comp= c(1,2), ind.names = FALSE, legend=TRUE,title =
```




```
plotVar(spca.fa_t, var.names = FALSE, comp= c(1,2),title = "PlotVar PC1 & PC2 fa")
```



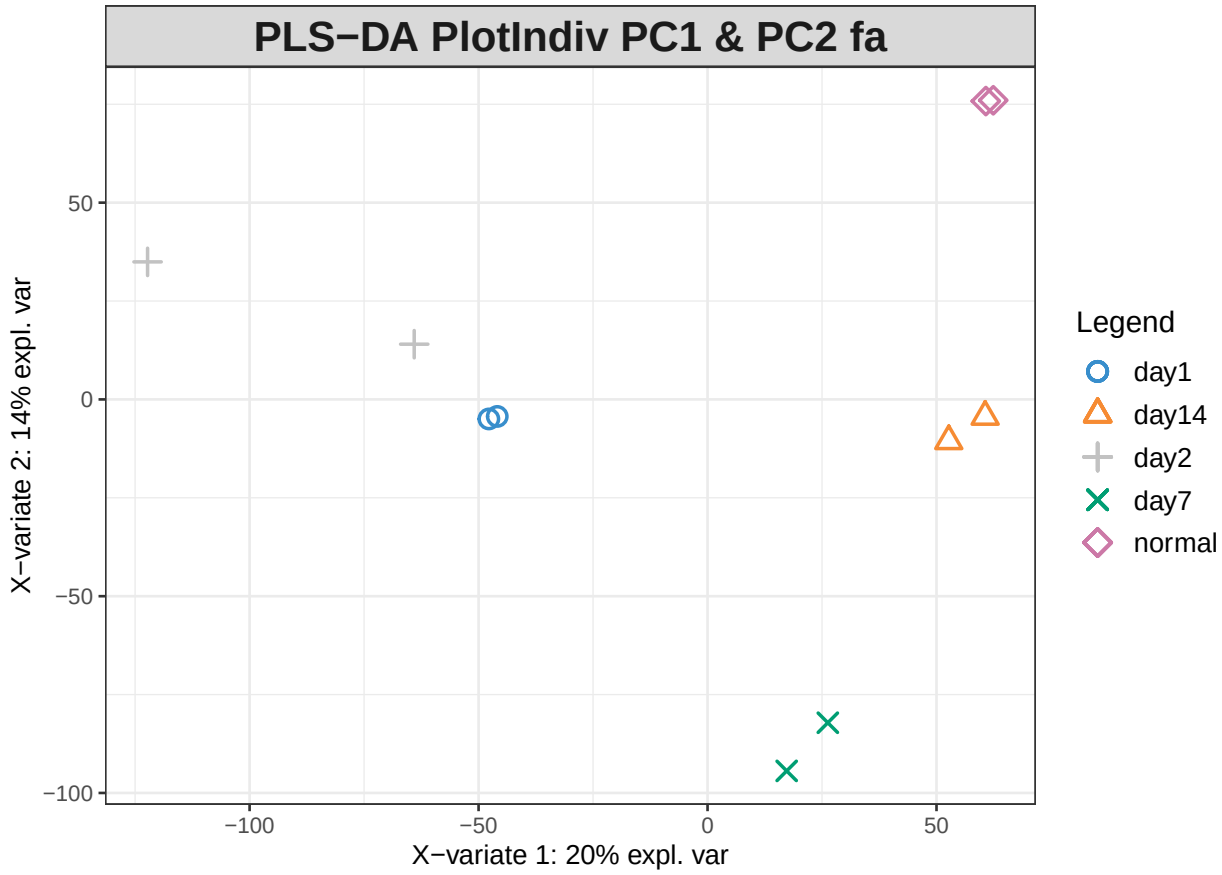

```
plotVar(spca.pfa_t, var.names = FALSE, comp= c(1,2),title = "PlotVar PC1 & PC2 pfa")
```



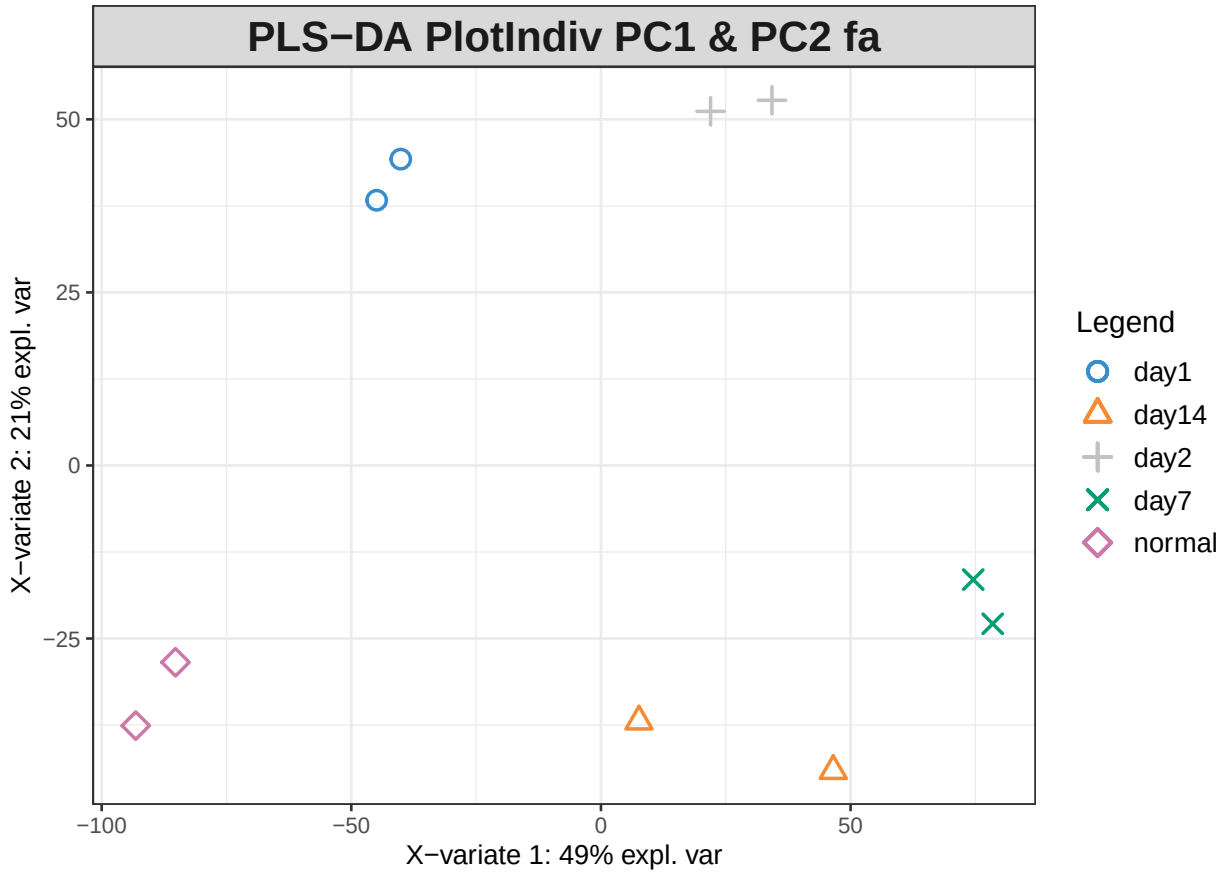
PLS-DA

Analyses multivariées supervisées

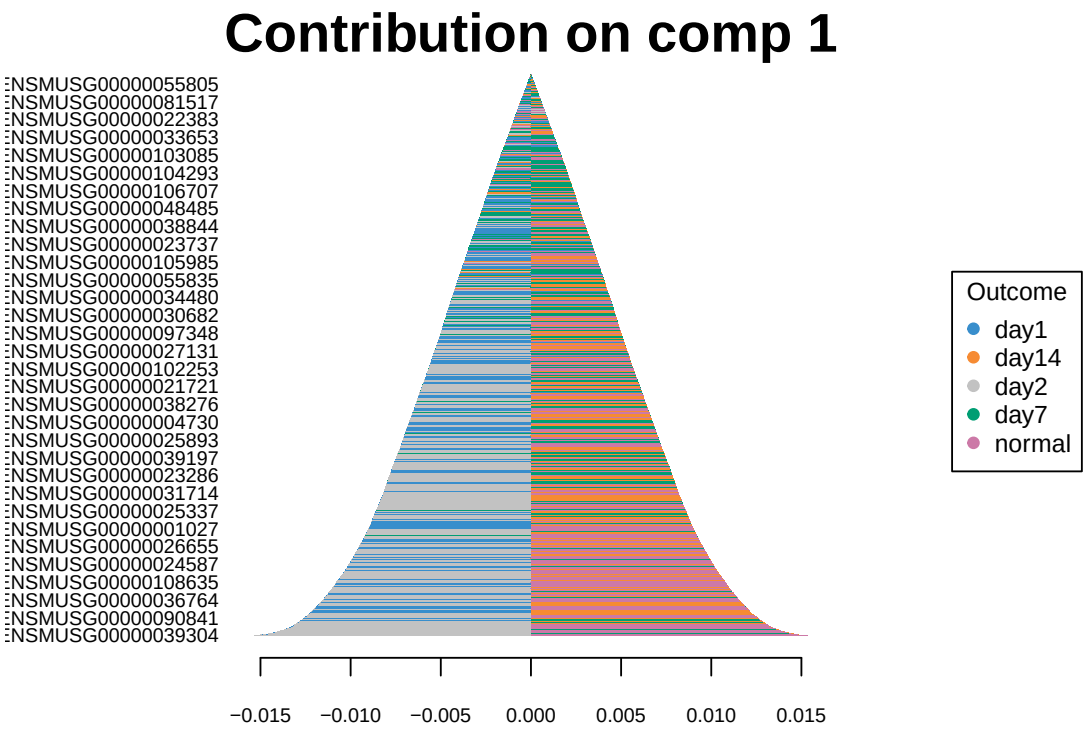
```
W <- metadata$condition

plsda.fa_t <- mixOmics::plsda(fa_t, W, ncomp = 3)
plsda.pfa_t <- mixOmics::plsda(pfa_t, W, ncomp = 3)

#Error: could not find function "plsda" donc rajouter mixOmics::
mixOmics::plotIndiv(plsda.fa_t, comp= c(1,2), ind.names=FALSE, legend=TRUE,title = "PLS-DA PlotIndiv PC
```



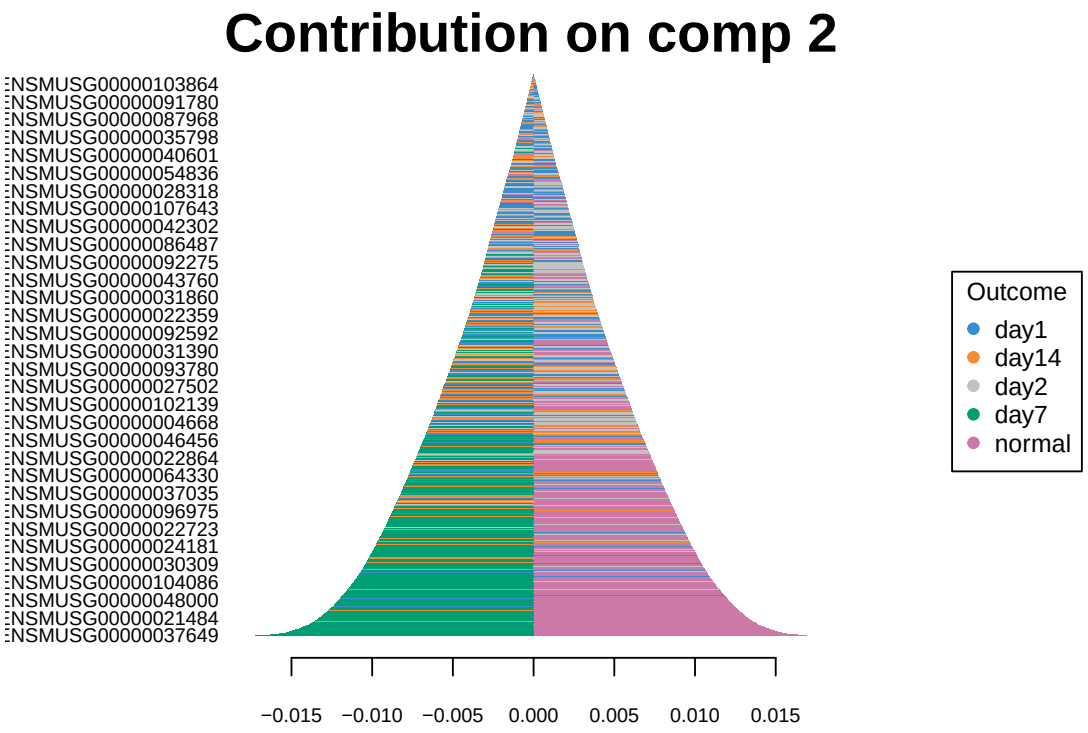

```
mixOmics::plotIndiv(plsda.pfa_t, comp= c(1,2), ind.names=FALSE, legend=TRUE,title = "PLS-DA PlotIndiv PC
```



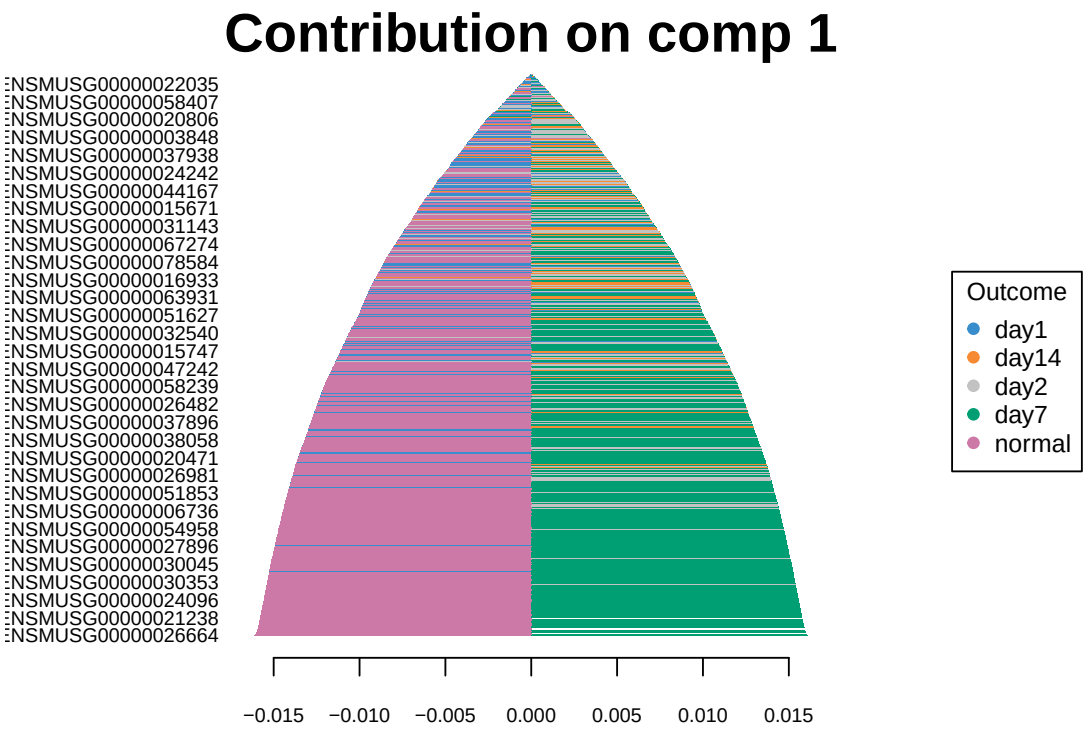

```
mixOmics::plotLoadings(plsda.fa_t, comp=1, contrib = "max")
```



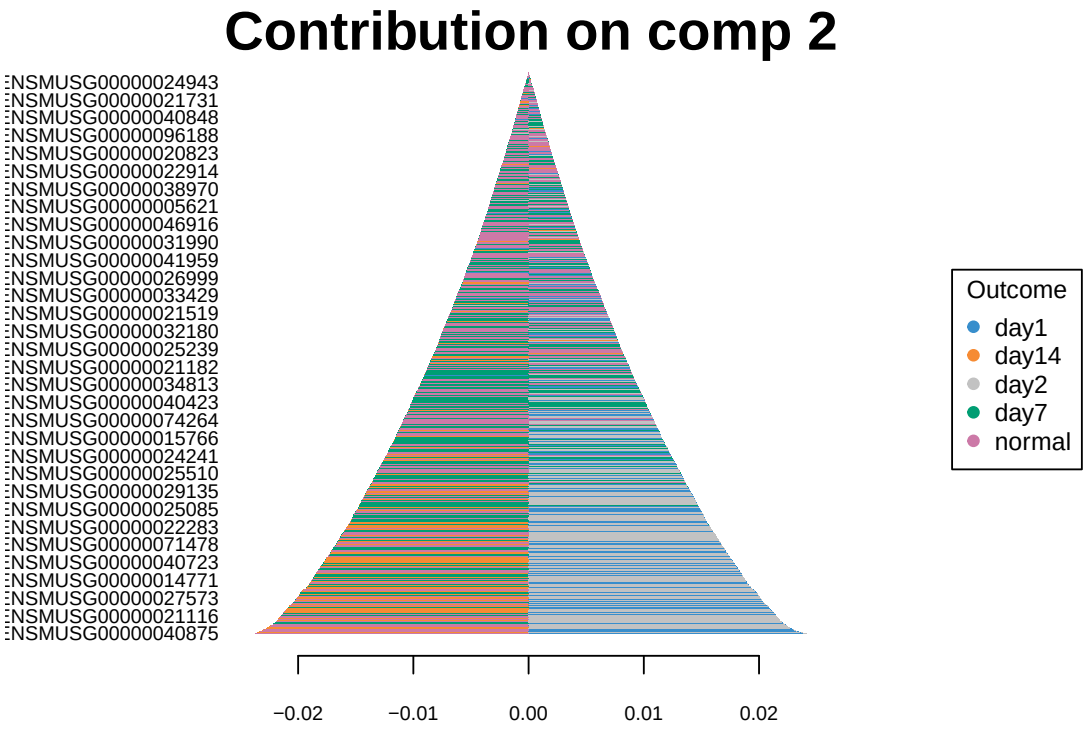

```
mixOmics::plotLoadings(plsda.fa_t, comp=2, contrib = "max")
```




```
mixOmics::plotLoadings(plsda.pfa_t, comp=1, contrib = "max")
```

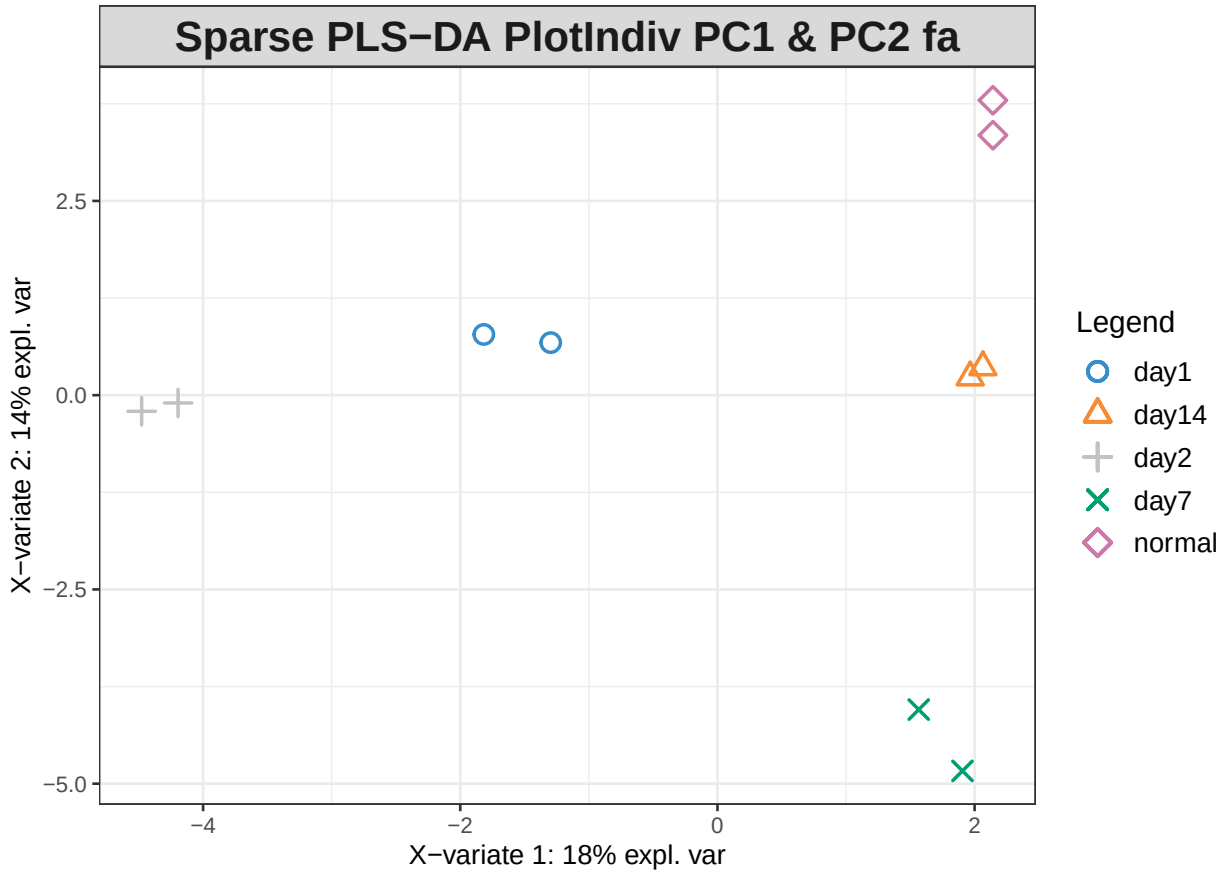



```
mixOmics::plotLoadings(plsda.pfa_t, comp=2, contrib = "max")
```

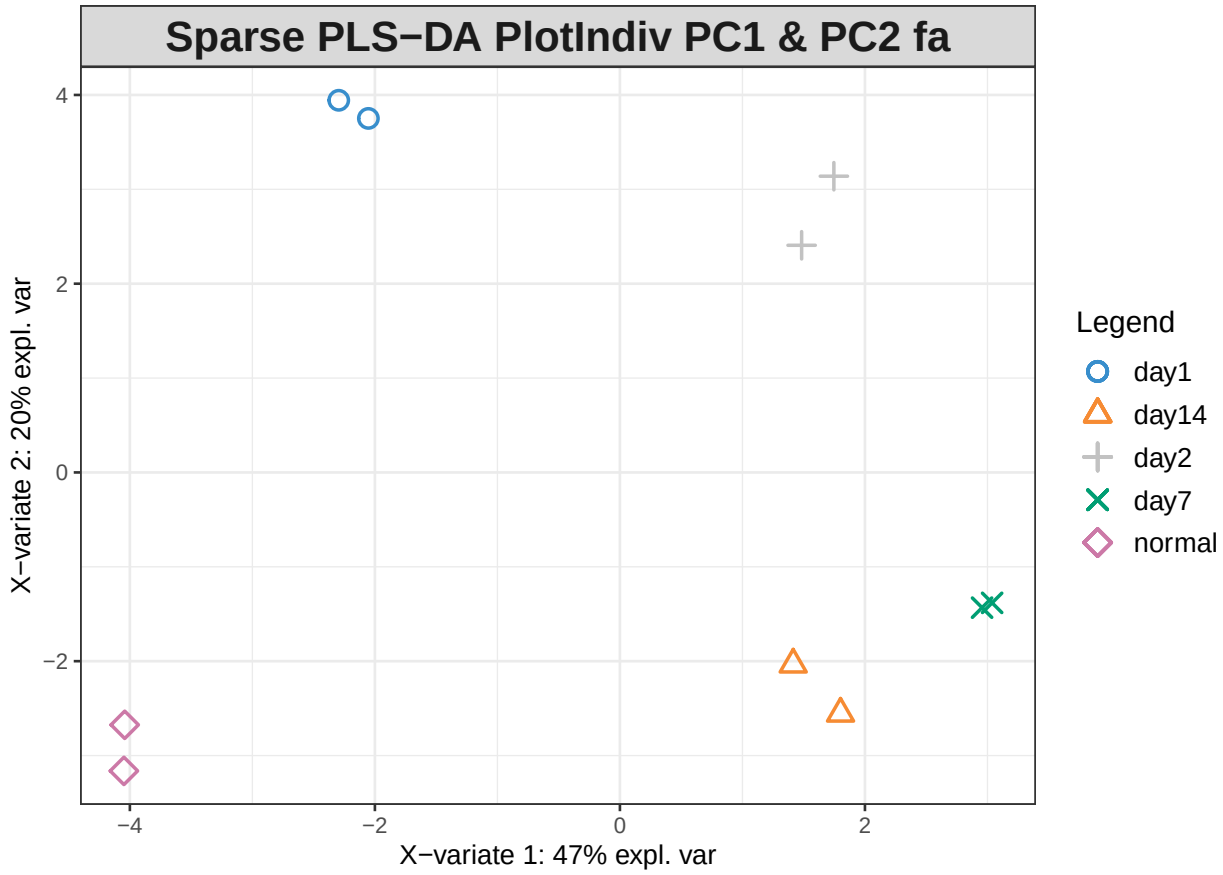


Sparse PLS-DA

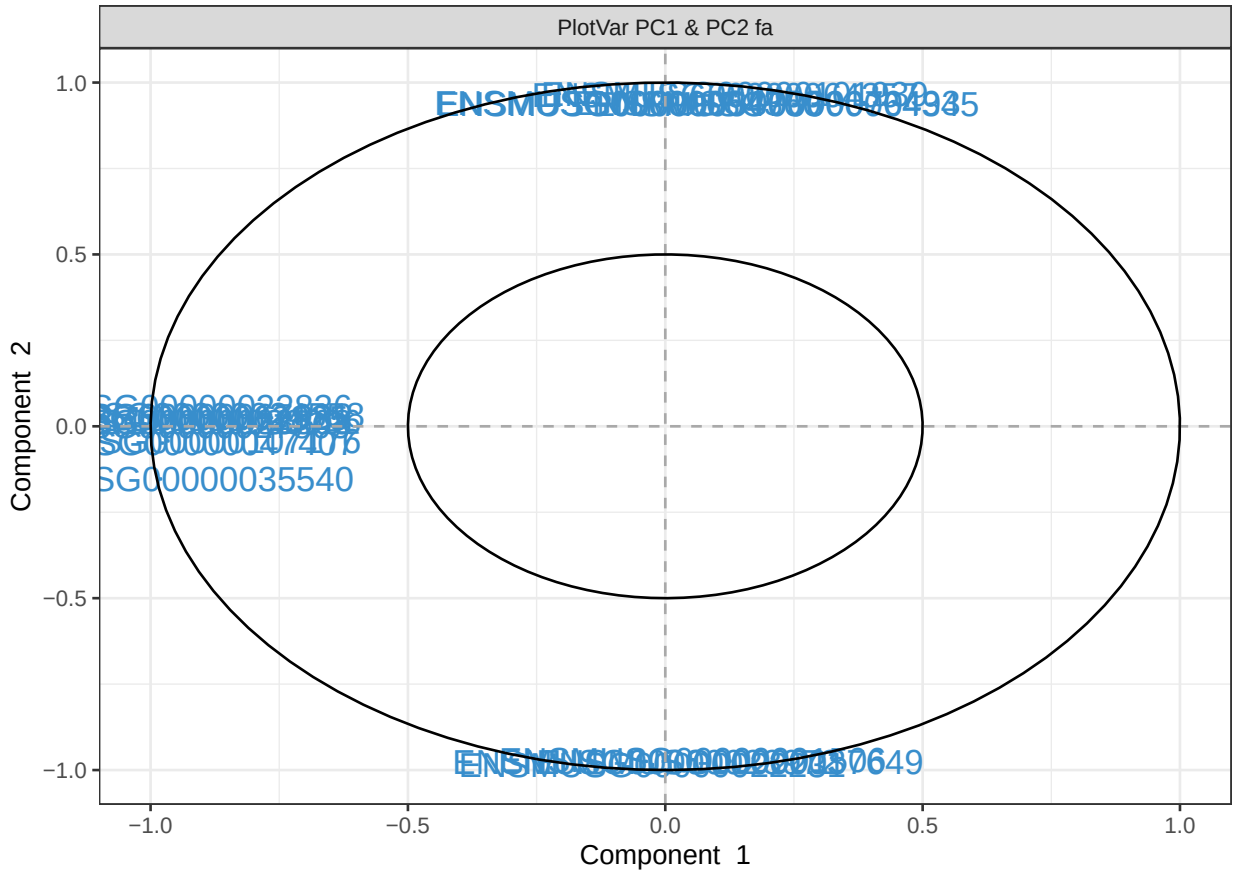
```
splsda.fa_t <- splsda(fa_t, W, ncomp = 3, keepX = c(10,10,10))
splsda.pfa_t <- splsda(pfa_t, W, ncomp = 3, keepX = c(10,10,10))

plotIndiv(splsda.fa_t, ind.names=FALSE, legend=TRUE,comp = c(1,2),title = "Sparse PLS-DA PlotIndiv PC1 &
```



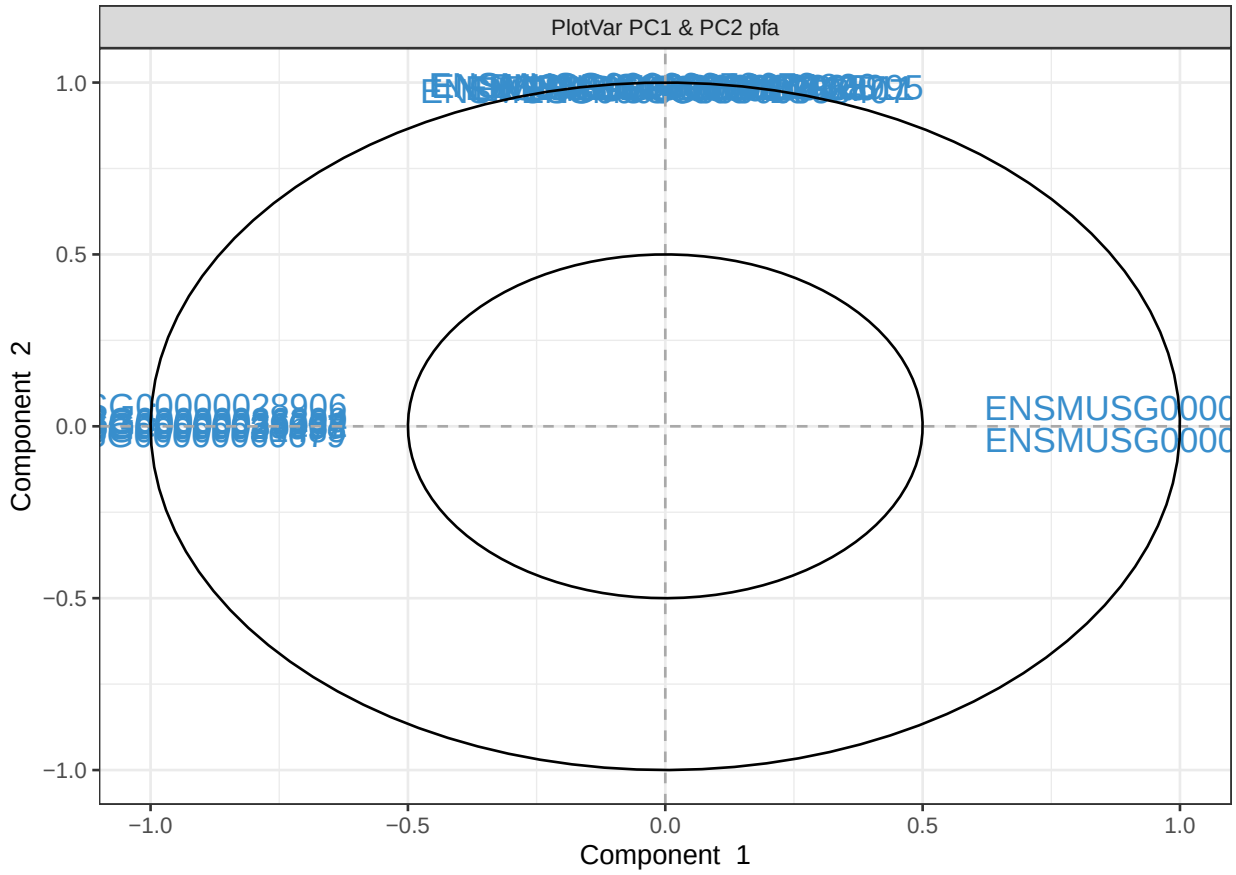

```
plotIndiv(splsda.pfa_t, ind.names=FALSE, legend=TRUE,comp = c(1,2),title = "Sparse PLS-DA PlotIndiv PC1
```



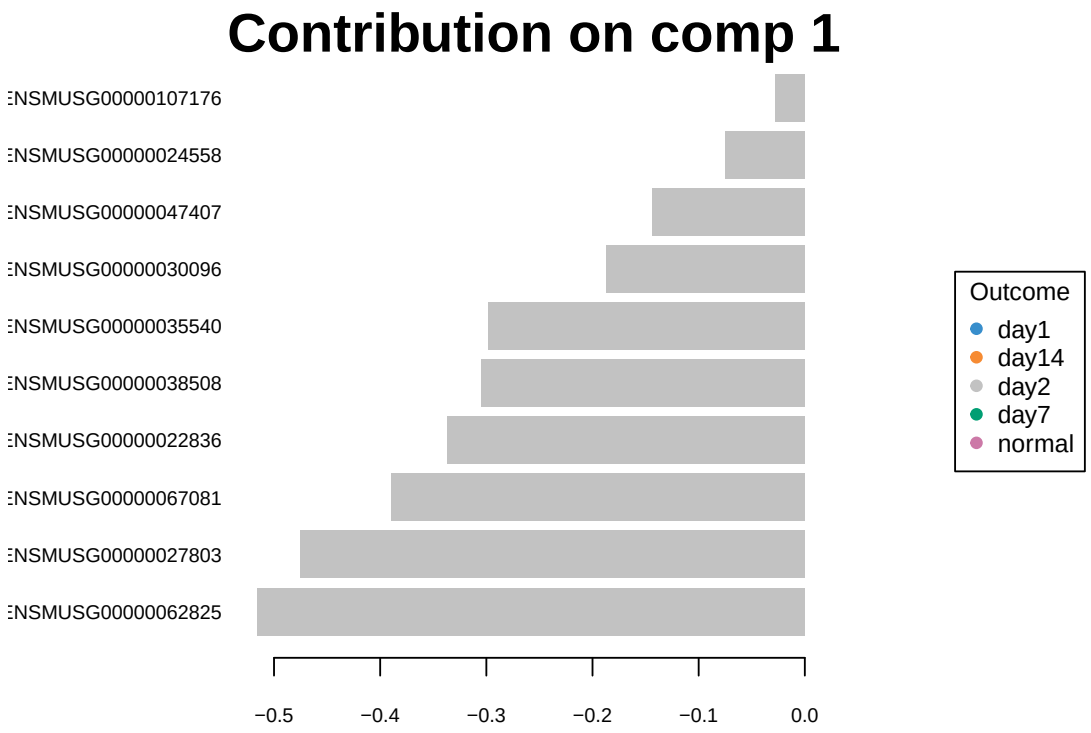

```
plotVar(splsda.fa_t, var.names = TRUE, comp= c(1,2),title = "PlotVar PC1 & PC2 fa")
```



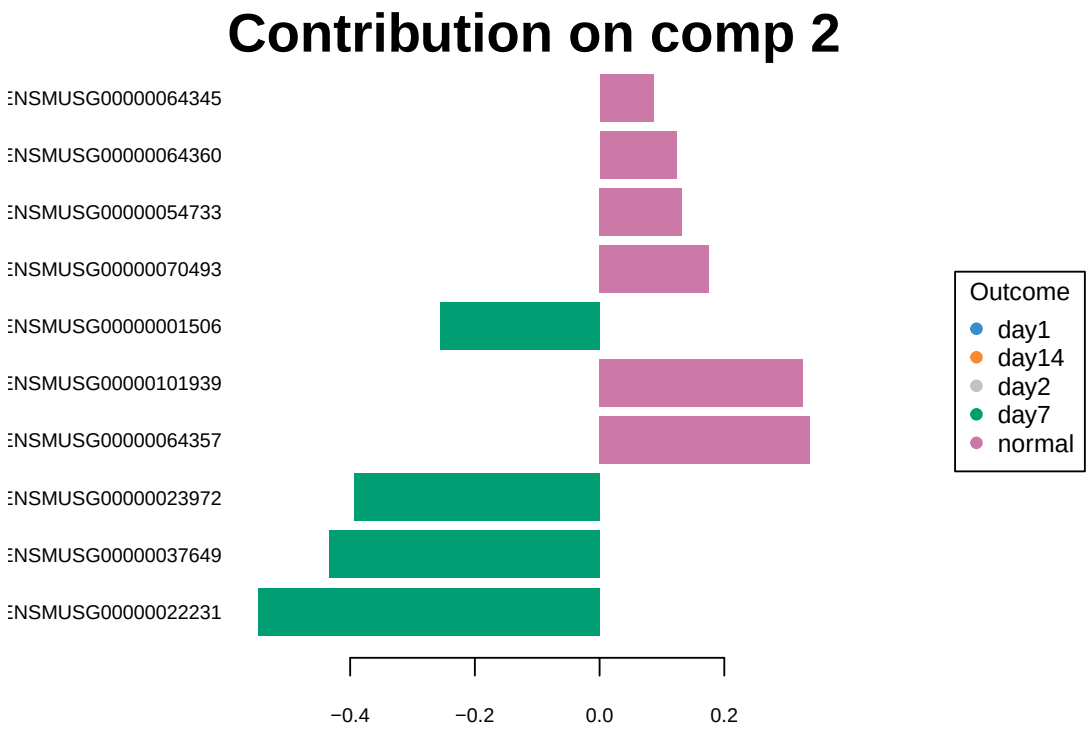

```
plotVar(splsda.pfa_t, var.names = TRUE, comp= c(1,2),title = "PlotVar PC1 & PC2 pfa")
```



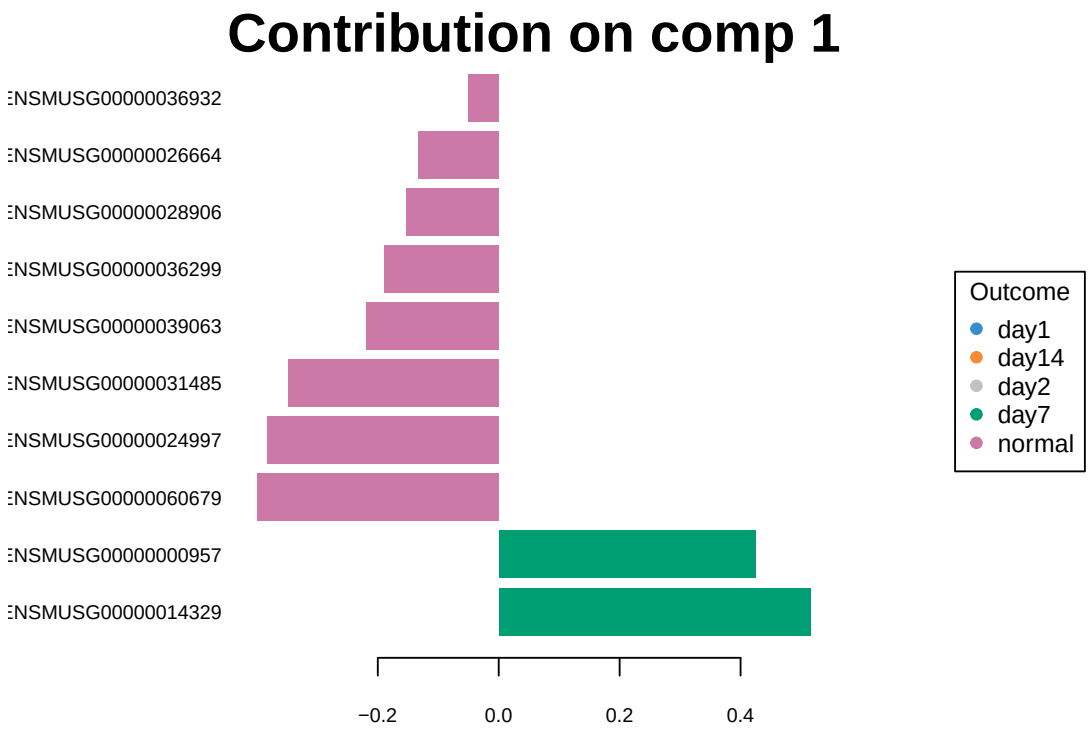

```
plotLoadings(splsda.fa_t, comp=1, contrib = 'max')
```



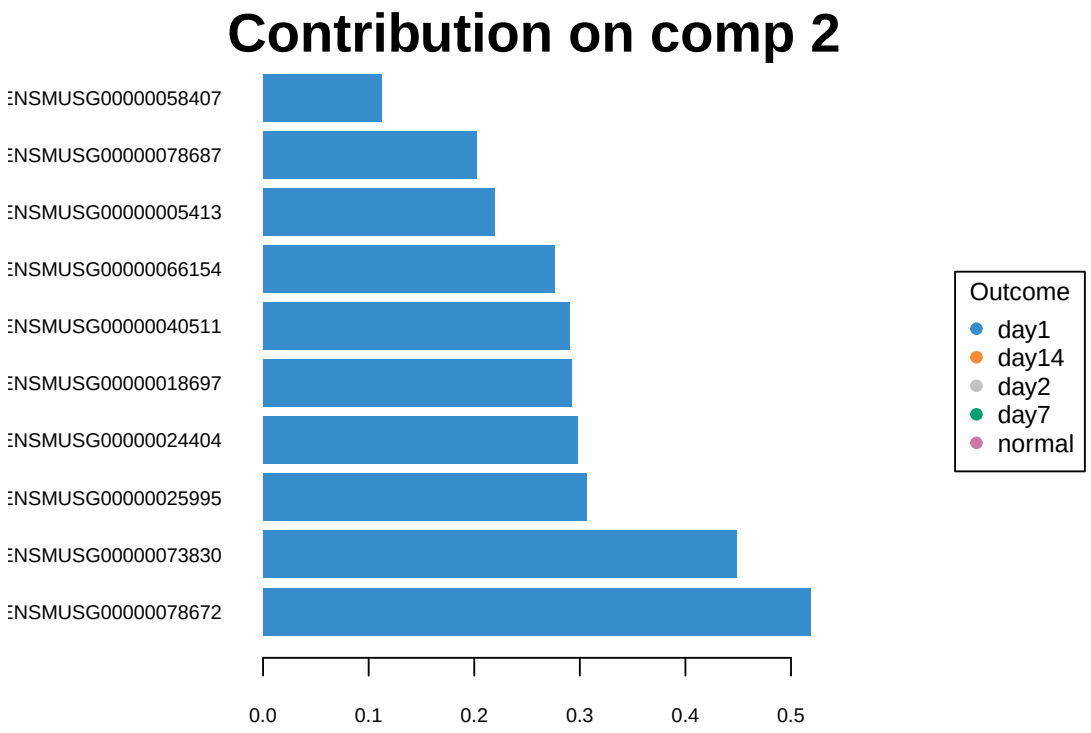

```
plotLoadings(splsda.fa_t, comp=2, contrib = 'max')
```




```
plotLoadings(splsda.pfa_t, comp=1, contrib = 'max')
```



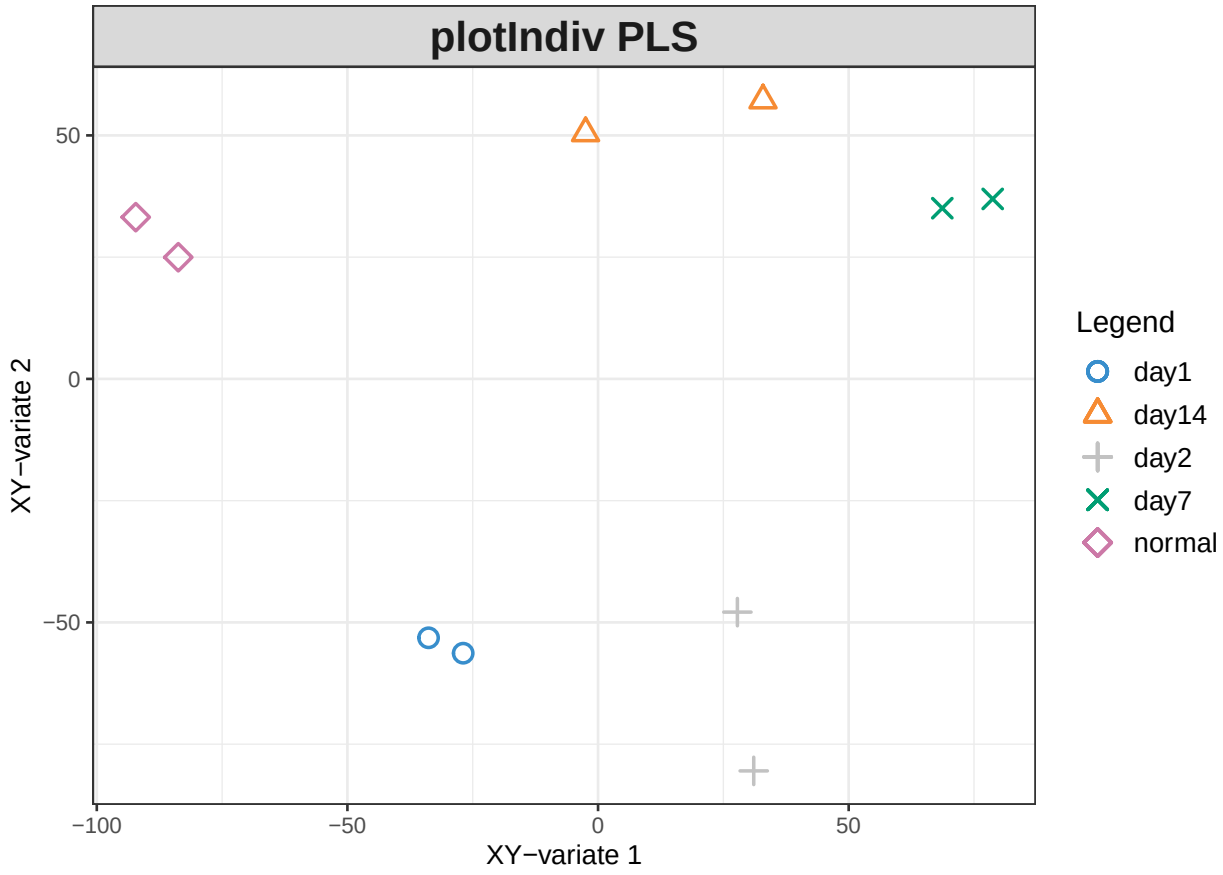

```
plotLoadings(splsda.pfa_t, comp=2, contrib = 'max')
```



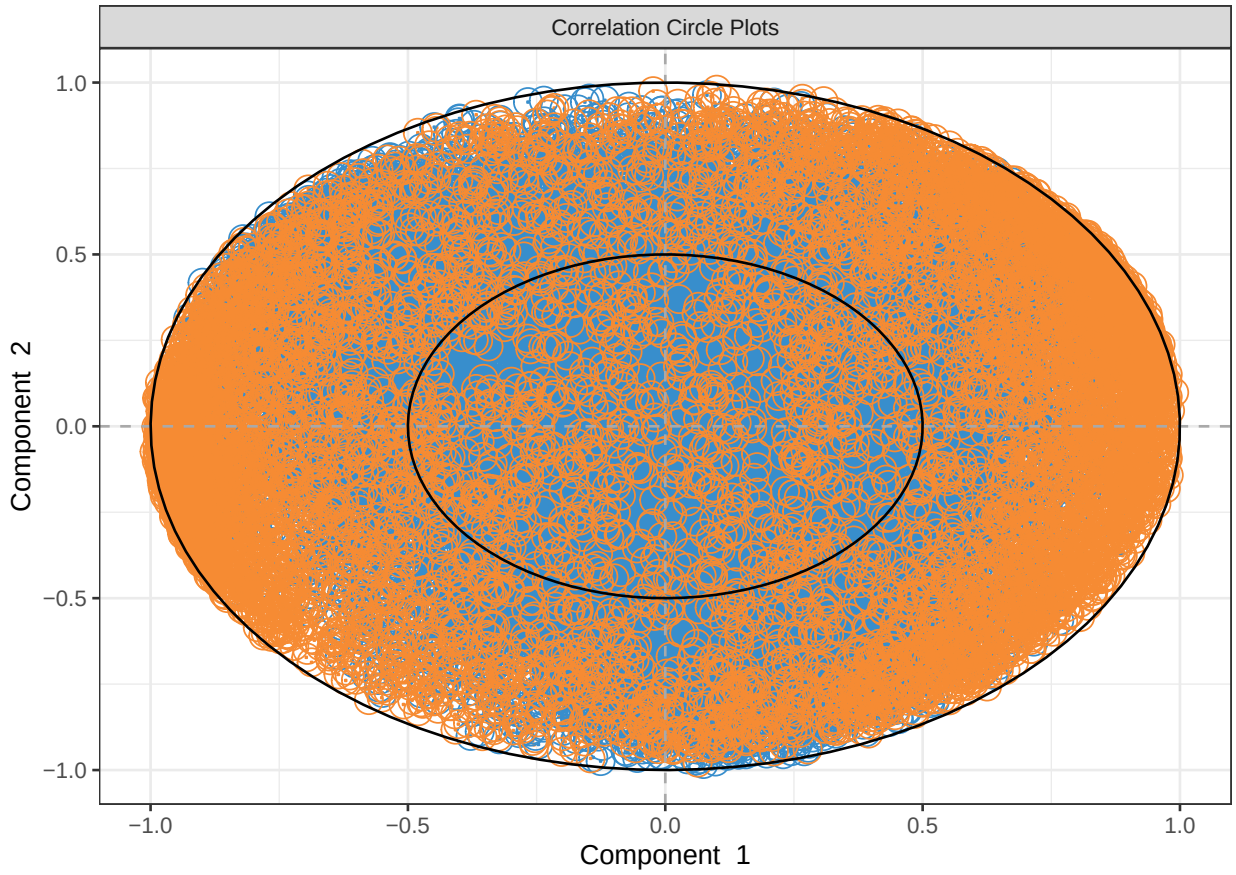
Analyses multi omics

PLS

```
pls.fa.pfa <- pls(fa_t,pfa_t)

plotIndiv(pls.fa.pfa,rep.space="XY-variate",
          title = "plotIndiv PLS",
          ind.names=FALSE,
          group=metadata$condition,
          pch = as.numeric(factor(metadata$condition)),
          pch.levels =metadata$condition,
          legend = TRUE)
```

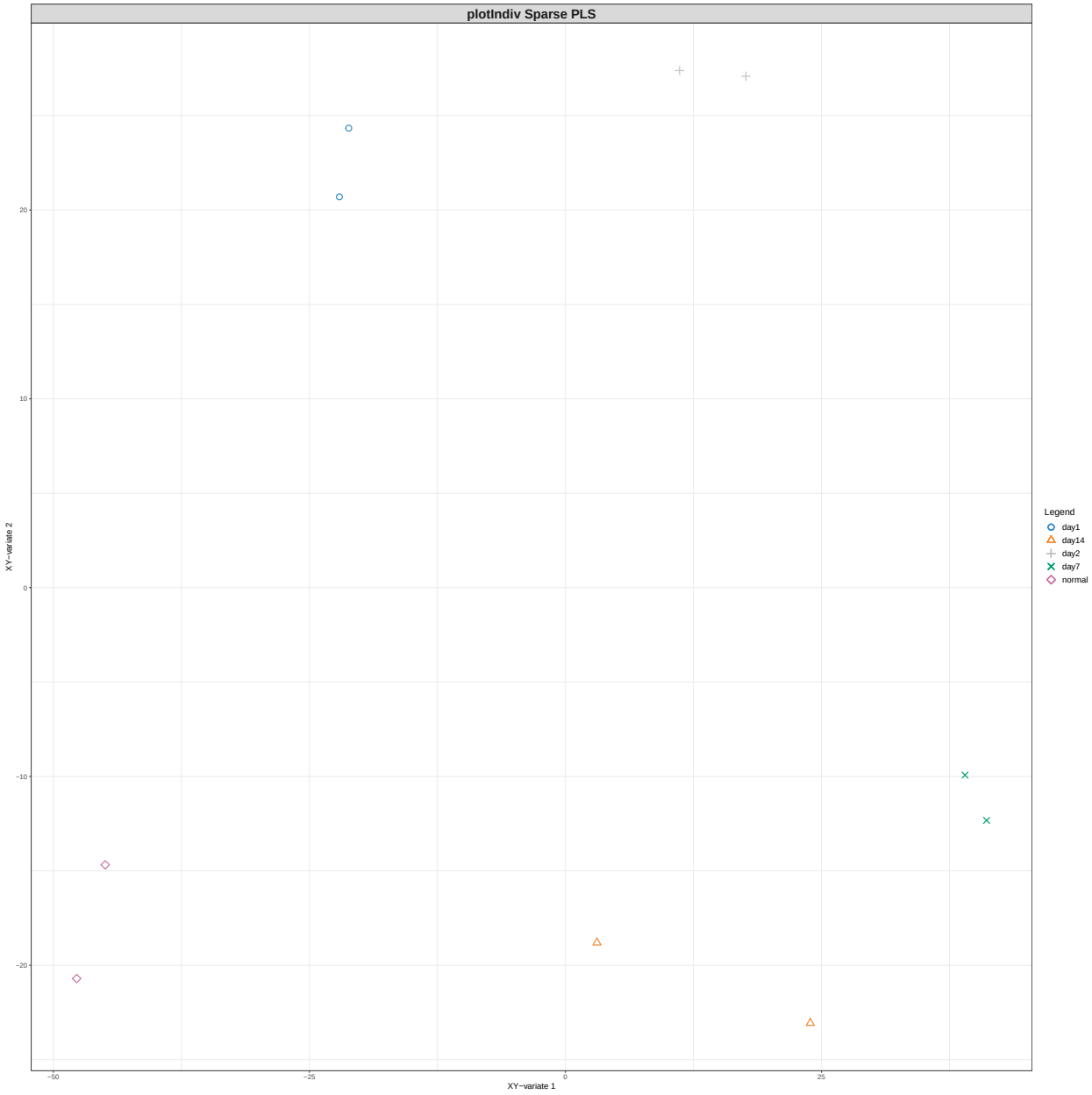



```
plotVar(pls.fa.pfa, var.names = c(FALSE, FALSE))
```

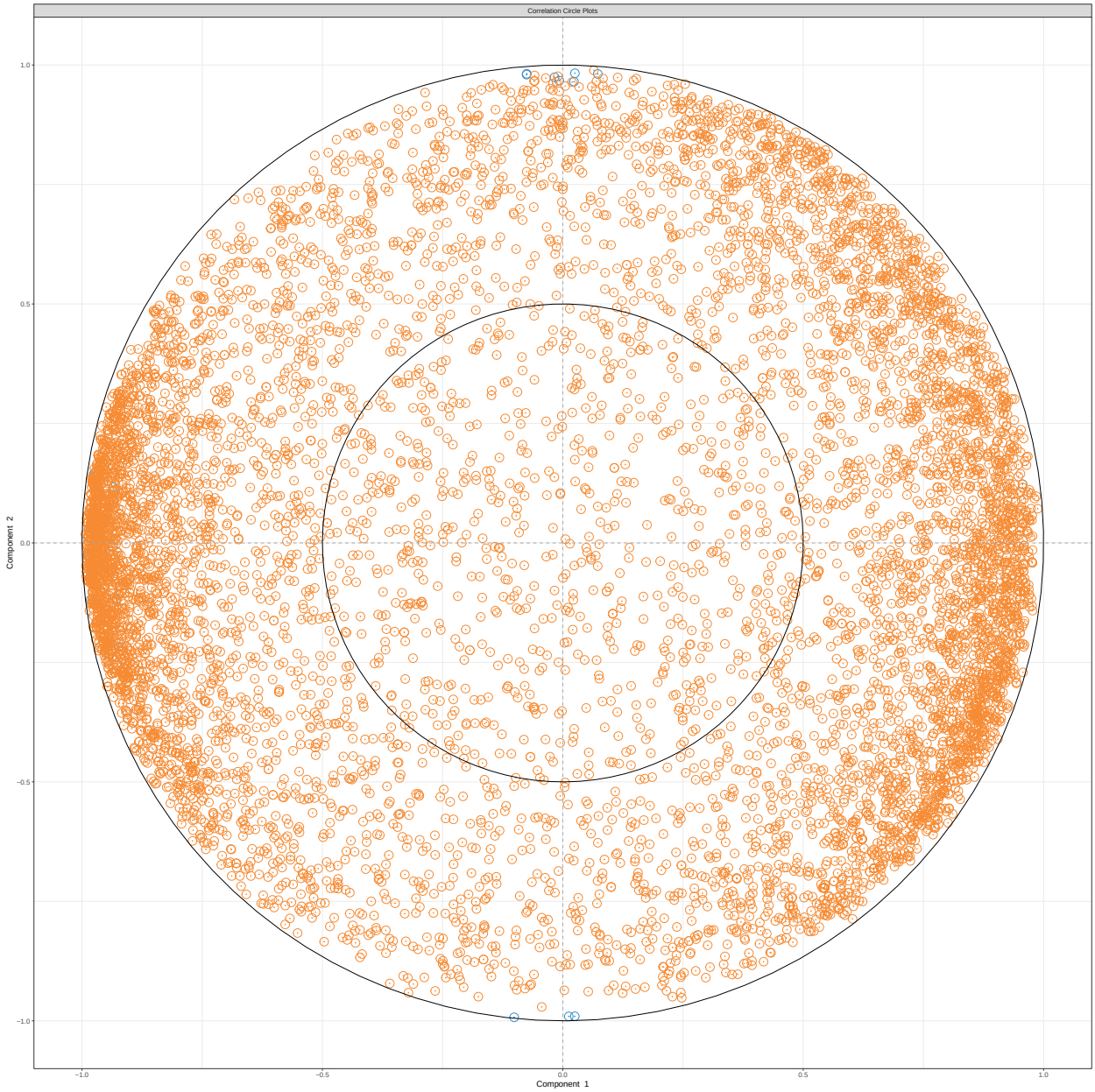


S-PLS

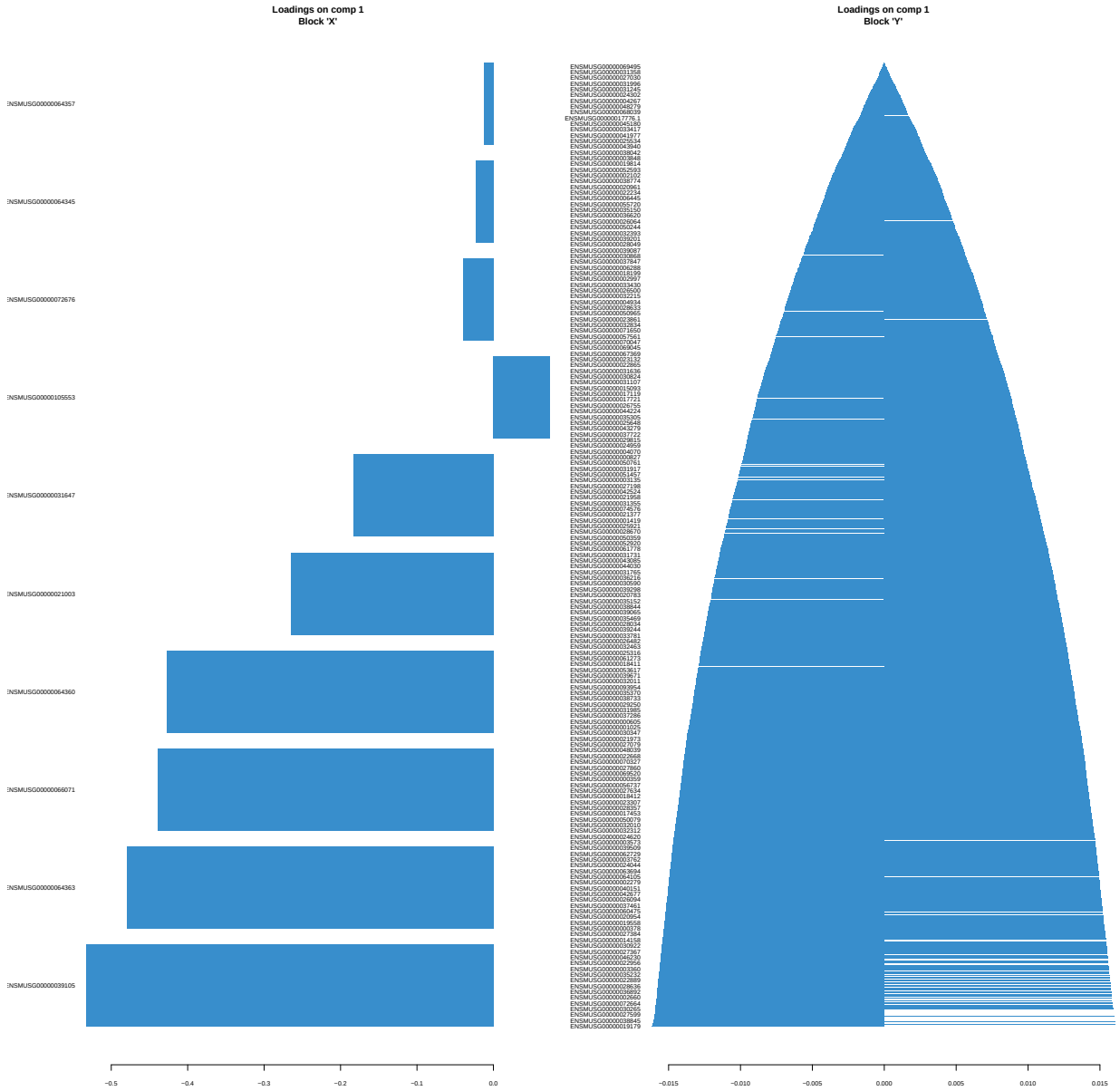
```
spls.fa.pfa <- spls(fa_t,pfa_t, ncomp=10, keepX = c(10,10,10))

plotIndiv(spls.fa.pfa,rep.space="XY-variate",
          title = "plotIndiv Sparse PLS",
          ind.names=FALSE,
          group=metadata$condition,
          pch = as.numeric(factor(metadata$condition)),
          pch.levels =metadata$condition,
          legend = TRUE)
```



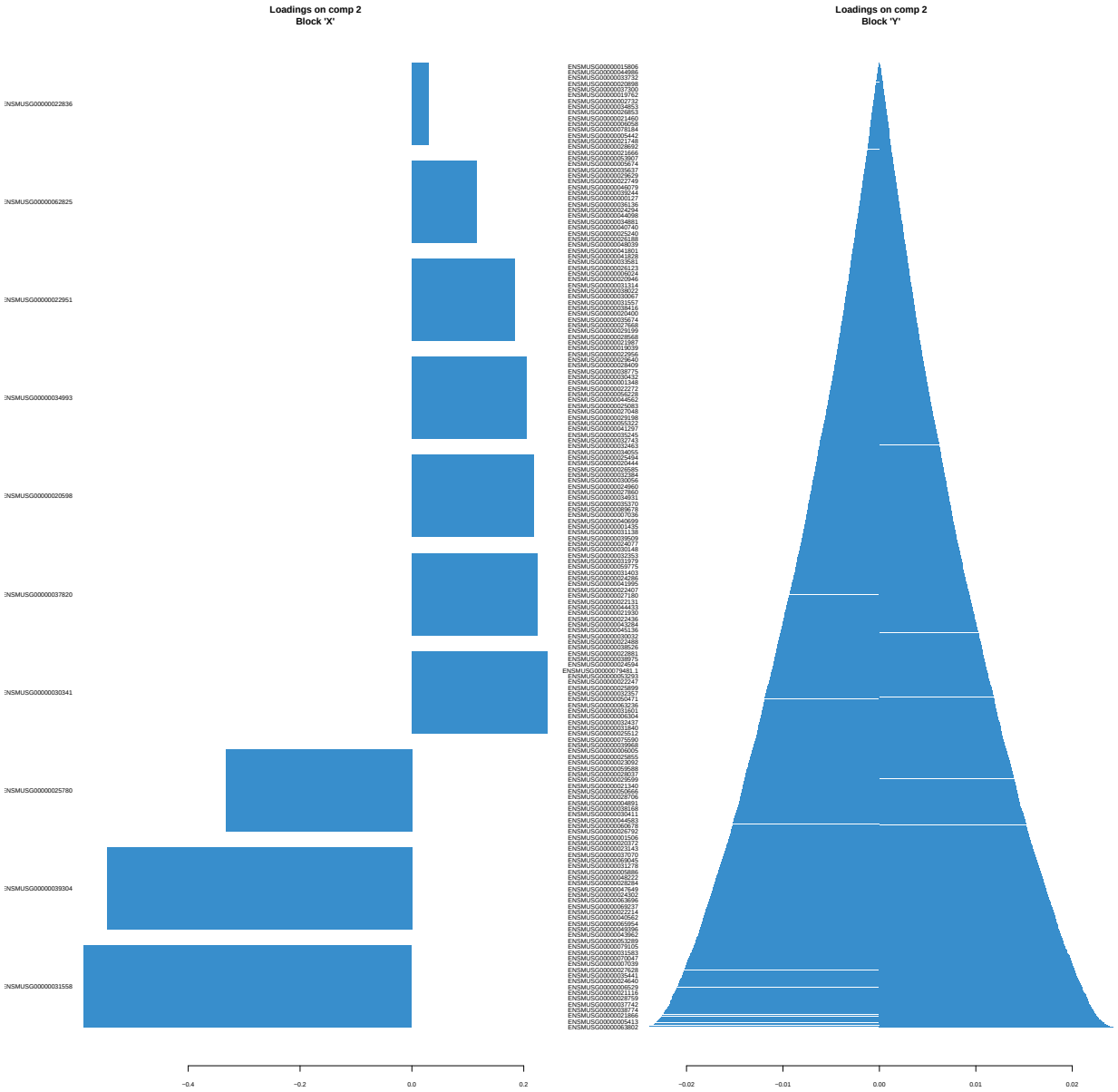

```
plotVar(spls.fa.pfa, var.names = c(FALSE, FALSE))
```




```
plotLoadings(spls.fa.pfa, comp=1, size.title = 1,name.var = NULL, max.name.length = 50)
```

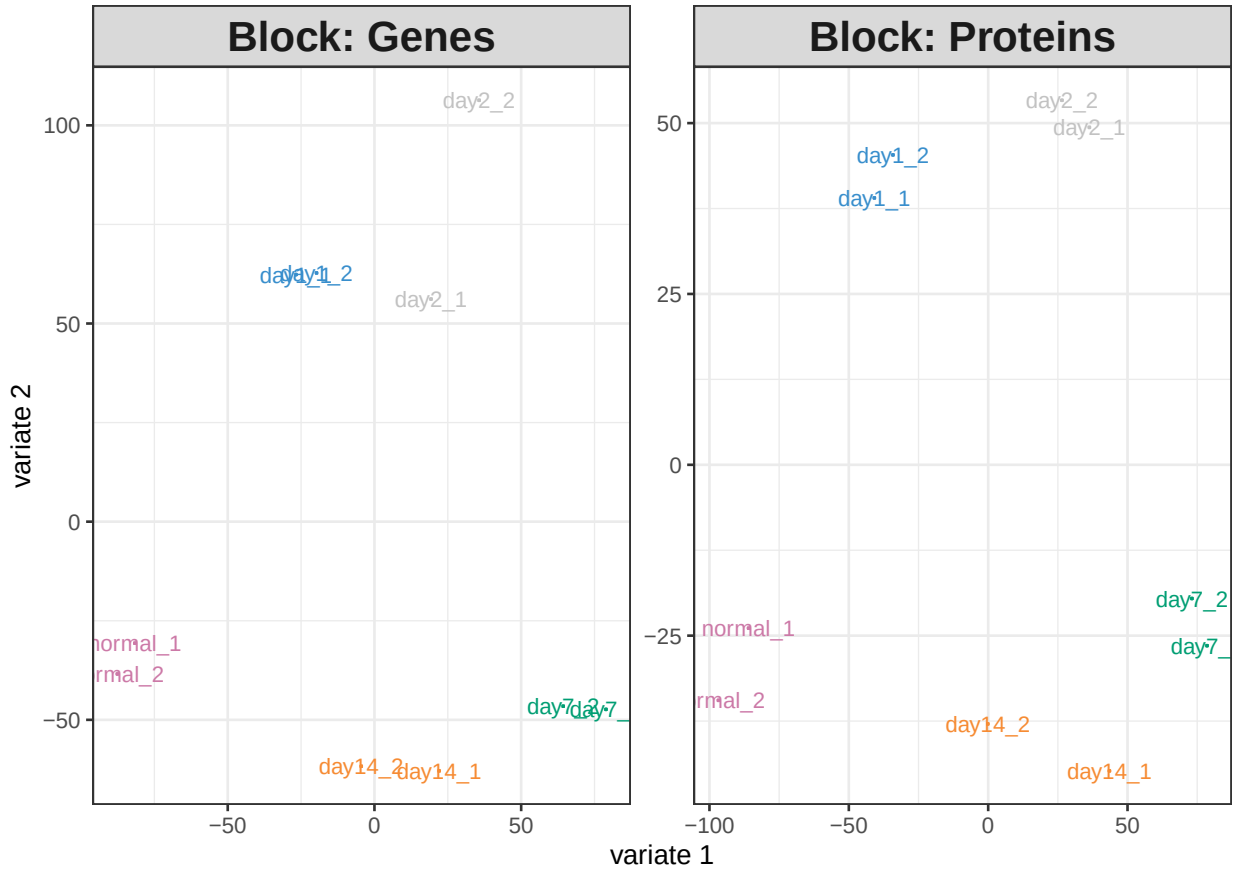



```
plotLoadings(spls.fa.pfa, comp=2, size.title = 1,name.var = NULL, max.name.length = 50)
```

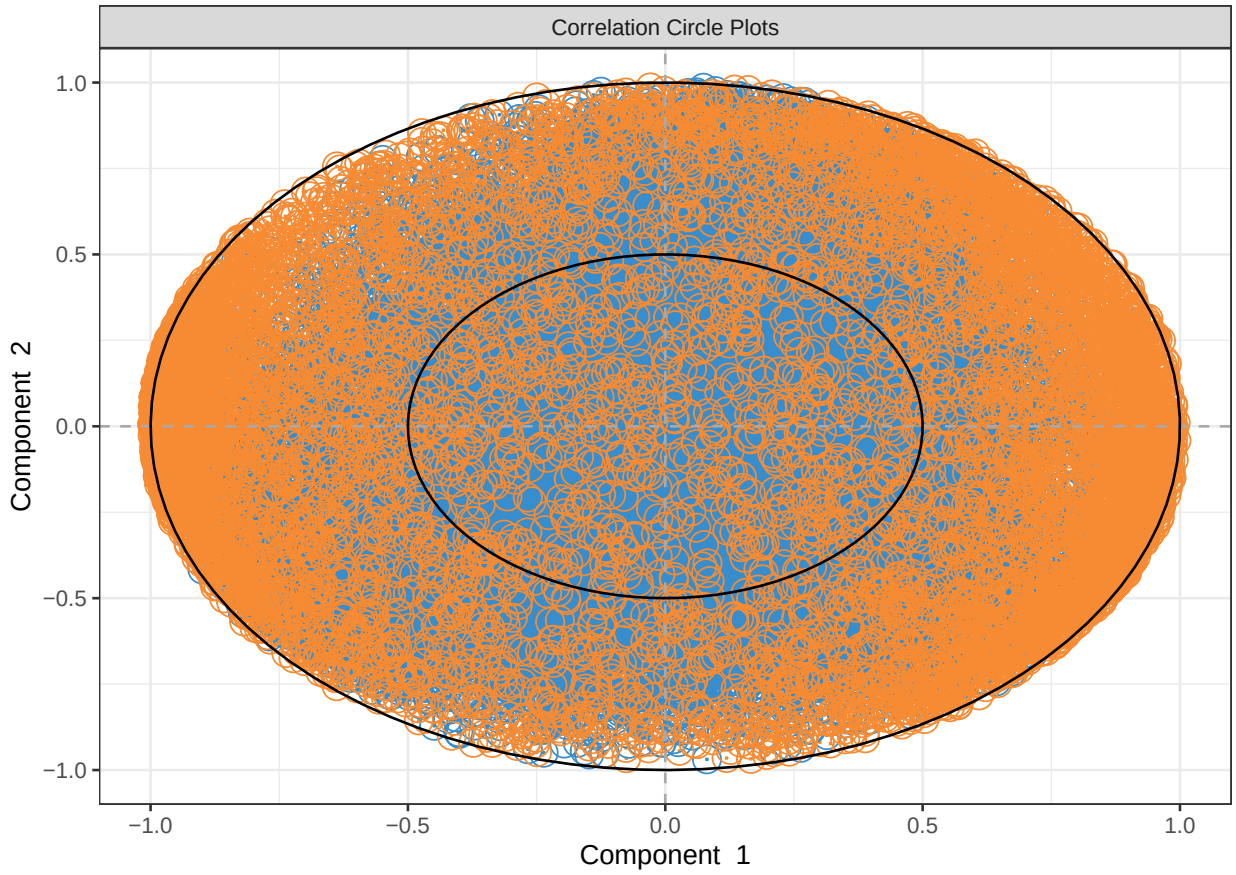


Multi-block PLS-DA

```
block.plsda.fa.pfa <- block.plsda(
  X = list(Genes = fa_t,
           Proteins = pfa_t),
  Y = metadata$condition)

plotIndiv(block.plsda.fa.pfa)
```




```
plotVar(block.plsda.fa.pfa, var.names = c(FALSE, FALSE))
```



Multi-block S-PLS-DA / DIABLO

```
data <- list(Genes = fa_t,
             Proteins = pfa_t)

lapply(data, dim)
```

```
$Genes
[1]    10 22459

$Proteins
[1]    10 8044
```

```
Y <- metadata$condition
Y
```

```
 [1] "normal" "normal" "day1"   "day1"   "day2"   "day2"   "day7"   "day7"   "day14"  "day14"
```

```
design = matrix(0.1, ncol = length(data), nrow = length(data),
                dimnames = list(names(data), names(data)))
```

```
diag(design) = 0

design
```

Design

```
         Genes Proteins
Genes      0.0      0.1
Proteins   0.1      0.0
```

Tout d'abord, nous ajustons un modèle DIABLO sans sélection de variable pour évaluer la performance globale et choisir le nombre de composants pour le modèle DIABLO final. La fonction perf est exécutée avec une validation croisée 10 fois répétée 10 fois.

```
sgccda.res = block.splsda(X = data, Y = Y, ncomp = 10, design = design)

#set.seed(123) # for reproducibility, only when the `cpus' argument is not used
# this code takes a couple of min to run
#perf.diablo = perf(sgccda.res, validation = 'Mfold', folds = 2, nrepeat = 3)

#perf.diablo  # lists the different outputs
#plot(perf.diablo)
```

```
#perf.diablo$choice.ncomp$WeightedVote
```

```
#ncomp = perf.diablo$choice.ncomp$WeightedVote["Overall.BER", "centroids.dist"]
#ncomp
```

Le nombre de composantes à garder est de 4

Tuning keepX Je choisis le nombre optimal de variables à sélectionner dans chaque ensemble de données à l'aide de la fonction tune.block.splsda.

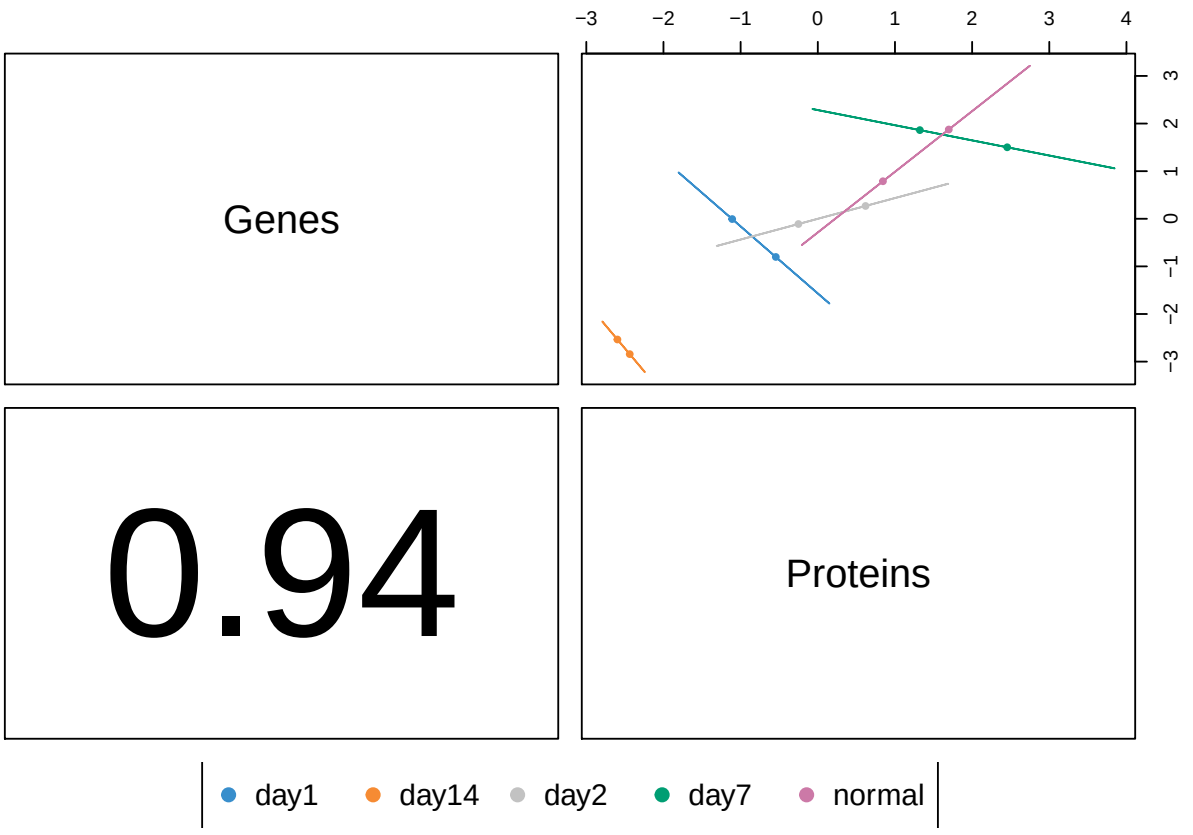
```
# #set.seed(123) # for reproducibility, only when the `cpus' argument is not used
#  test.keepX = list (microbiote = c(1:9, seq(10, 45, 5), seq(50,150,10)),
#                     caecum = c(1:9, seq(10, 45, 5), seq(50,150,10)),
#                     hypothalamus = c(1:9, seq(10,45 , 5), seq(50,150,10)))
#
#  tune.TCGA = tune.block.splsda(X = data, Y = Y, ncomp = ncomp,
#                                test.keepX = test.keepX, design = design,
#                                validation = 'Mfold', folds = 10, nrepeat = 1,
#                                 dist = "centroids.dist")
#
#
# list.keepX = tune.TCGA$choice.keepX
#
# #qsub -cwd -V -N test_keepX -q long.q -pe thread 4 -o test_keepX.out -e test_keepX.err -b y "Rscript
#
list.keepX <- list(Genes = c(4, 6,5,5), Proteins = c(5,7,5,6))
list.keepX
```

```
$Genes
[1] 4 6 5 5

$Proteins
[1] 5 7 5 6
```

```
sgccda.res = block.splsda(X = data, Y = Y, ncomp = 4,
                          keepX = list.keepX, design = design)
```

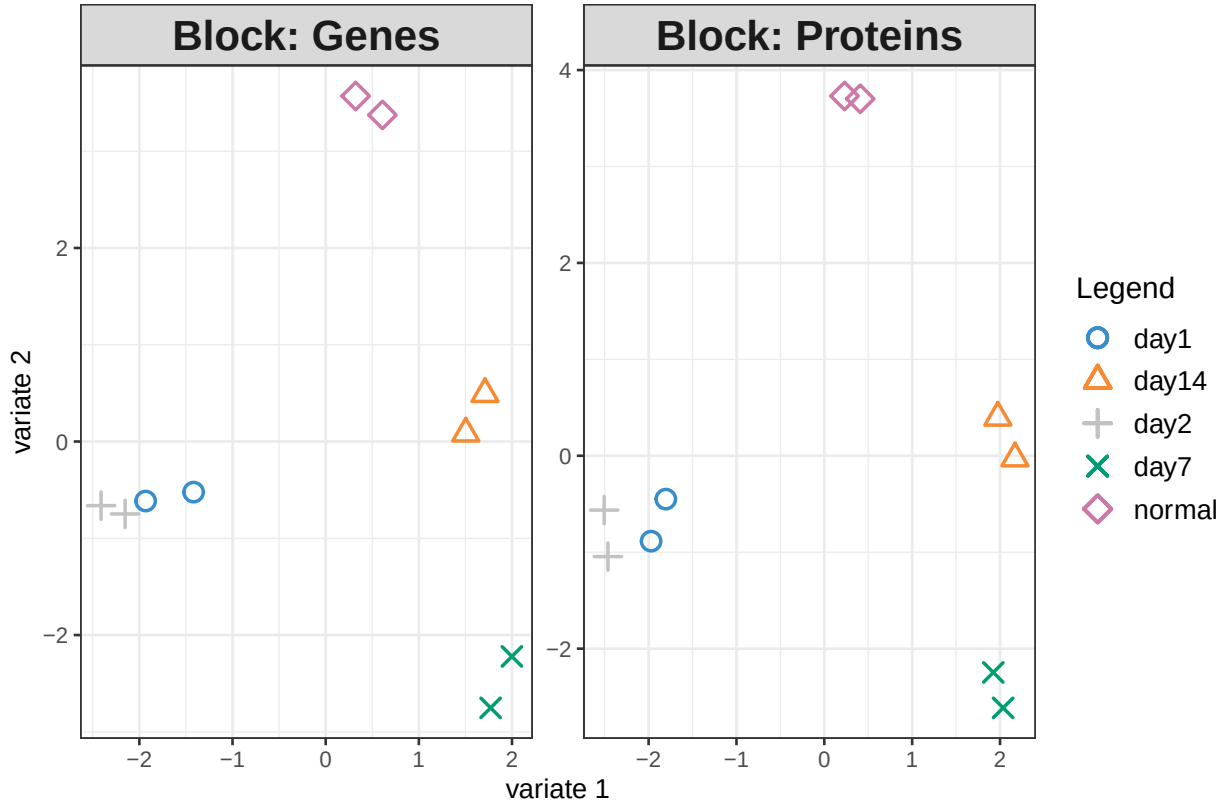
Final model plotDIABLO est un tracé de diagnostic pour vérifier si la corrélation entre les composants de chaque ensemble de données a été maximisée comme spécifié dans la matrice de conception.

```
plotDiablo(sgccda.res, ncomp = 4)
```

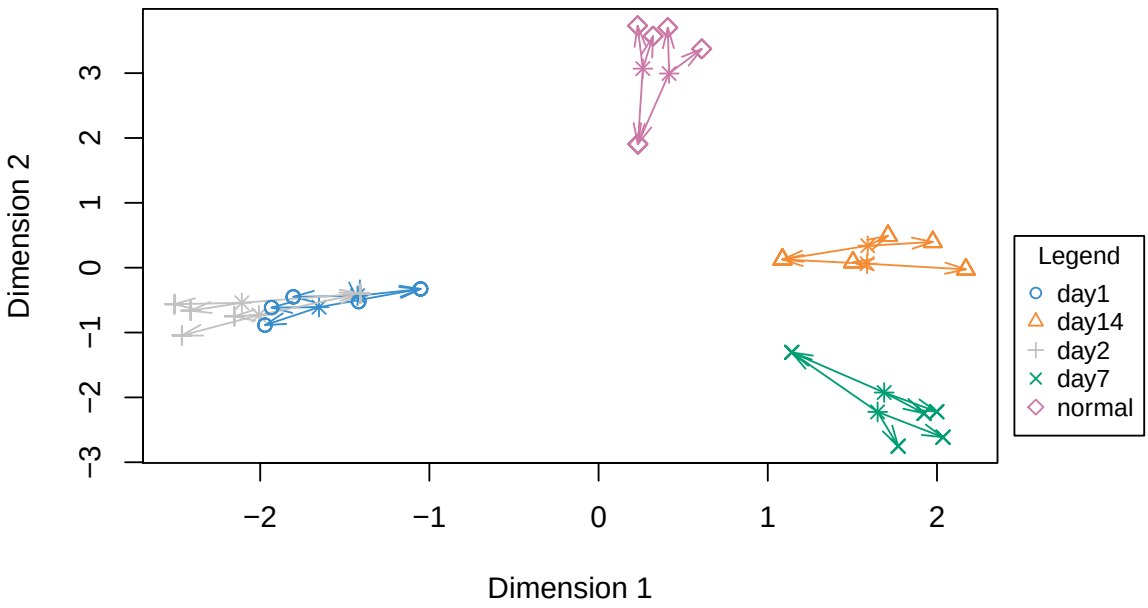



```
plotIndiv(sgccda.res, ind.names = FALSE, legend = TRUE, title = 'DIABLO')
```

DIABLO

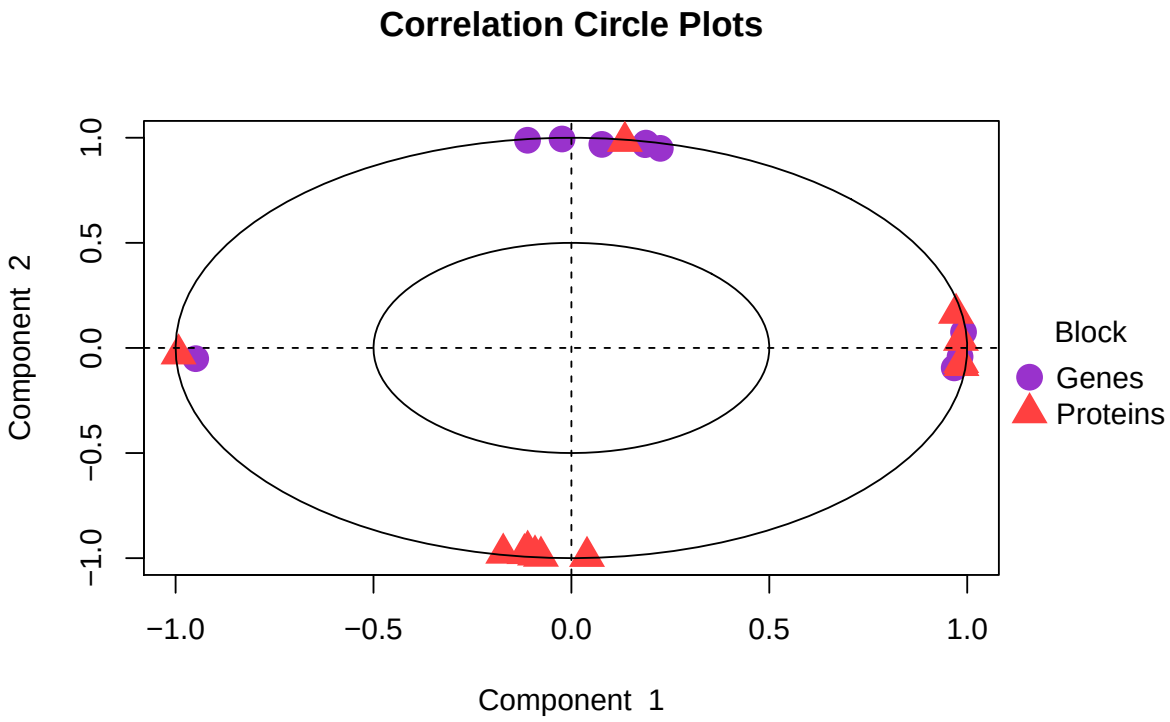



```
plotArrow(sgccda.res, ind.names = FALSE, legend = TRUE, title = 'DIABLO')
```

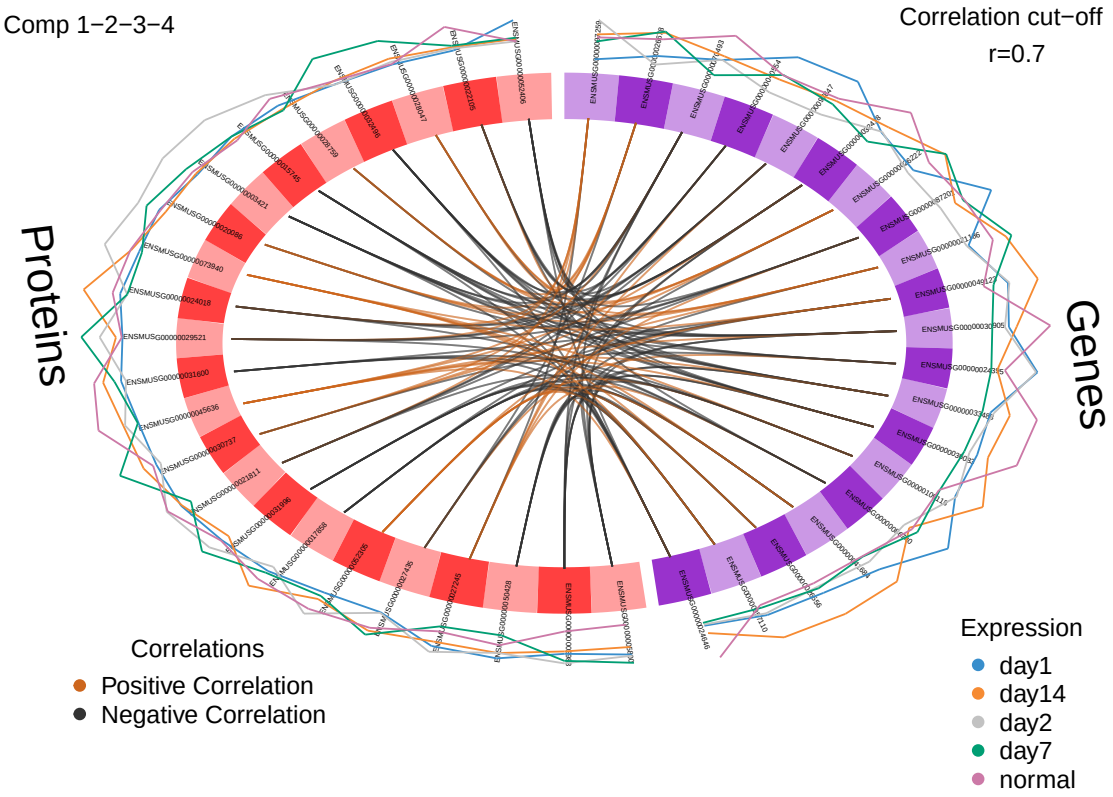



```
plotVar(sgccda.res, var.names = FALSE, style = 'graphics', legend = TRUE,
        pch = c(16, 17), cex = c(2,2), col = c('darkorchid', 'brown1'))
```

Variable plots




```
circosPlot(sgccda.res, cutoff = 0.7, line = TRUE,
           color.blocks= c('darkorchid', 'brown1'),
           color.cor = c("chocolate3","grey20"), size.labels = 1.5)
```



WGCNA

Reconstruct the co-expression network from all the time points of the FA transcriptomics data. Propose to filter and remove all the zero expressed genes, the NAs and the less informative genes from the transcriptomics data. (I remove all the genes that are not expressed in at least 9 out of the 18 conditions (expression $>$ 1 TPM in 9) and then filter with the coefficient of variation $>$ 0.75).

Then apply the first part of the network reconstruction steps as we saw them on the WGCNA course until the module predictions.

Instead of using WGCNA's module prediction routines, apply a universal threshold of 0.5 on the adjacency matrix, and obtain an adjacency matrix that is reduced in size. This is the network.

```
#Load the WGCNA package
library(WGCNA)
```

J'utilise les données de transtriptomique **fa normalisées**

J'harmonise les row.names entre la count table et les metadatas

```
# #je remets en data frame car j'ai un problème de couleur module(je n'ai pas les noms des genes dans l
# datExpr_view <- datExpr[,1:10]
# datExpr_view
# class(datExpr)
```

```
# datExpr <- as.data.frame(datExpr)


head(fa_expr_norm)
```

```
                    normal_1    normal_2   normal_3     day1_1      day1_2       day1_3      day2_1
ENSMUSG00000000001 719.032852 1247.299971 809.375515 2711.22417 1094.745589 1331.5880610 1185.658698 123
ENSMUSG00000000003   0.000000    0.000000   0.000000    0.00000    0.000000    0.0000000    0.000000
ENSMUSG00000000028  10.420766    7.448588   1.700369   42.82759   13.485108   21.9214582  166.379146   1
ENSMUSG00000000031   1.488681    0.000000   0.000000   15.46552    4.290716    0.5620887    1.105509
ENSMUSG00000000037  47.637787    6.771444   0.000000    0.00000   17.162865    3.9346207   31.507014
ENSMUSG00000000049  41.683064   25.731487  27.205900   36.87931    3.064797   28.6665222    6.080301   1
```

```
class(fa_expr_norm)
```

```
[1] "data.frame"
```

```
fa_meta  <- read.csv("fa_meta.csv",sep="," , row.names = 1, header = TRUE)
fa_meta <- fa_meta[,-c(3,4)]
fa_meta
```

```
         sampleName condition
normal_1   normal_1    normal
normal_2   normal_2    normal
normal_3   normal_3    normal
day1_1       day1_1      day1
day1_2       day1_2      day1
day1_3       day1_3      day1
day2_1       day2_1      day2
day2_2       day2_2      day2
day2_3       day2_3      day2
day3_1       day3_1      day3
day3_2       day3_2      day3
day3_3       day3_3      day3
day7_1       day7_1      day7
day7_2       day7_2      day7
day7_3       day7_3      day7
day14_1     day14_1     day14
day14_2     day14_2     day14
day14_3     day14_3     day14
```

```
fa_annotation  <- read.csv("fa_annotations.csv",sep=";" , row.names = 1, header = TRUE)
head(fa_annotation)
```

```
                  external_gene_name chromosome_name start_position end_position strand
ENSMUSG00000000001              Gnai3               3      108014596    108053462     -1
ENSMUSG00000000028              Cdc45              16       18599197     18630737     -1
ENSMUSG00000000037              Scml2               X      159865521    160041209      1
ENSMUSG00000000049               Apoh              11      108234180    108305222      1
ENSMUSG00000000056               Narf              11      121128079    121146682      1
ENSMUSG00000000058               Cav2               6       17281184     17289114      1
```

```
fa_trait <- read.csv("fa_trait.csv",sep="," , row.names = 1, header = TRUE)
fa_trait
```

```
         control day1 day2 day3 day7 day14
normal_1       1    0    0    0    0     0
normal_2       1    0    0    0    0     0
normal_3       1    0    0    0    0     0
day1_1         0    1    0    0    0     0
day1_2         0    1    0    0    0     0
day1_3         0    1    0    0    0     0
day2_1         0    0    1    0    0     0
day2_3         0    0    1    0    0     0
day3_1         0    0    0    1    0     0
day3_2         0    0    0    1    0     0
day3_3         0    0    0    1    0     0
day7_1         0    0    0    0    1     0
day7_2         0    0    0    0    1     0
day7_3         0    0    0    0    1     0
day14_1        0    0    0    0    0     1
day14_2        0    0    0    0    0     1
day14_3        0    0    0    0    0     1
```

```
## Classer suivant ordre colonne Name de fa_meta
sample_order <- order(fa_meta$sampleName)
fa_meta <- fa_meta[sample_order, ]

## Ordre des echantillons de data table idem fa_meta table
fa_expr_norm <- fa_expr_norm[, row.names(fa_meta)]
head(fa_expr_norm)
```

```
                     day1_1      day1_2       day1_3    day14_1    day14_2    day14_3      day2_1    d
ENSMUSG00000000001 2711.22417 1094.745589 1331.5880610 728.668808 603.846446 646.124076 1185.658698 123
ENSMUSG00000000003    0.00000    0.000000    0.0000000   0.000000   0.000000   0.000000    0.000000    0
ENSMUSG00000000028   42.82759   13.485108   21.9214582  16.244209  10.802262  33.839559  166.379146   23
ENSMUSG00000000031   15.46552    4.290716    0.5620887   1.160301   0.000000   0.000000    1.105509    0
ENSMUSG00000000037    0.00000   17.162865    3.9346207   6.961804   2.160452   1.057486   31.507014    0
ENSMUSG00000000049   36.87931    3.064797   28.6665222  30.167817   2.160452  59.219228    6.080301   14
```

```
fa_meta
```

```
         sampleName condition
day1_1       day1_1      day1
day1_2       day1_2      day1
day1_3       day1_3      day1
day14_1     day14_1     day14
day14_2     day14_2     day14
day14_3     day14_3     day14
day2_1       day2_1      day2
day2_2       day2_2      day2
day2_3       day2_3      day2
day3_1       day3_1      day3
day3_2       day3_2      day3
```

```
day3_3       day3_3      day3
day7_1       day7_1      day7
day7_2       day7_2      day7
day7_3       day7_3      day7
normal_1   normal_1    normal
normal_2   normal_2    normal
normal_3   normal_3    normal
```

Filtres

Je filtre et je supprime tous les gènes non exprimés, les NA et je supprime tous les gènes qui ne sont pas exprimés dans au moins 9 des 18 conditions (expression> 1 TPM dans 9)

Puis filtre avec le coefficient de variation> 0,75).

```
dim(fa_expr_norm)
```

```
[1] 46679    18
```

```
#voir le nombre de lignes avec zero counts
rs <- rowSums(fa_expr_norm)
nbgenes_at_zeros <- length(which(rs==0))
nbgenes_at_zeros
```

```
[1] 11457
```

```
#Je supprime les lignes avec zero counts
fa_expr_norm <- fa_expr_norm[rowSums(fa_expr_norm[, -1])>0, ]
dim(fa_expr_norm)
```

```
[1] 35107    18
```

```
#Je supprime les NA
library(tidyr)
fa_expr_norm <- fa_expr_norm %>% drop_na()
dim(fa_expr_norm)
```

```
[1] 35107    18
```

```
#Je ne garde que les genes où il y a moins de 9 échantillons avec des counts supérieurs ou égaux à 1.

#fa_969 <- fa_expr_norm[rowSums((fa_expr_norm[, -1])>=1) >= 9, ]
#dim(fa_969)

nbexpr <- apply(fa_expr_norm, 1, function(x){length(which(x>=1))})
isexpr <- which(nbexpr>=9)
fa_filtre <- fa_expr_norm[isexpr,]
dim(fa_filtre)
```

```
[1] 24608    18
```

Je filtre avec le coefficient de variation> 0.75 et je **TRANSPOSE** la count table

```
#Je calcul le cv par gène
gene_mean <- apply(fa_filtre, 1, mean)
gene_sd <- apply(fa_filtre, 1, sd)
gene_cv <- gene_sd / gene_mean

#Je filtre les genes avec un cv > 0.75
fa_cv <- fa_expr_norm[gene_cv > 0.75, ]
dim(fa_cv)
```

```
[1] 13899    18
```

```
######TRANSPOSER LA COUNT TABLE#######
fa_cv <- t(fa_cv)
```

```
gsg = goodSamplesGenes(fa_cv, verbose = 3);
```

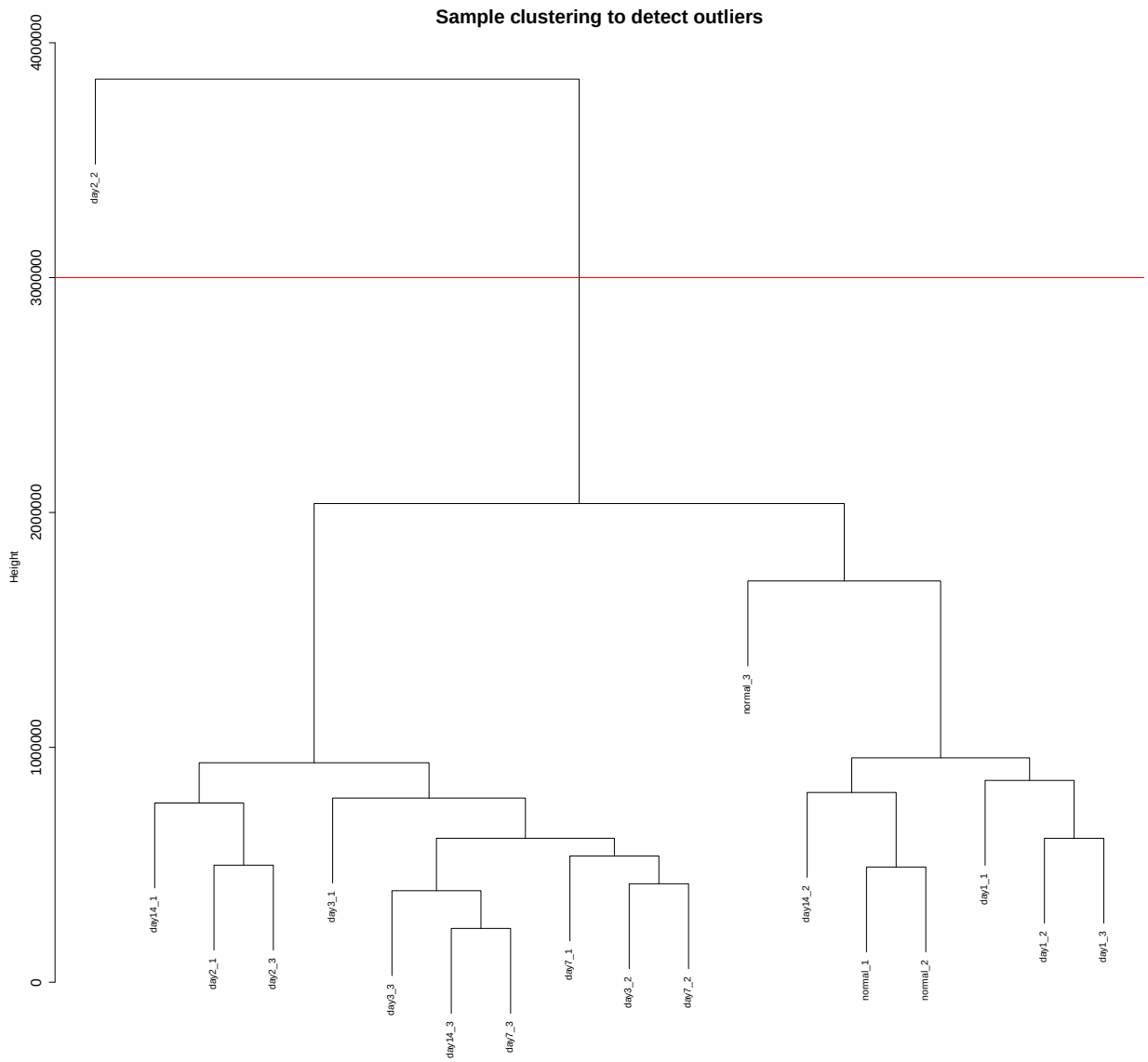
```
 Flagging genes and samples with too many missing values...
  ..step 1
```

```
gsg$allOK
```

```
[1] TRUE
```

Arbre permettant de détecter les valeurs abérrantes

```
sampleTree = hclust(dist(fa_cv), method = "average");
# Plot the sample tree: Open a graphic output window of size 12 by 10 inches
# The user should change the dimensions if the window is too large or too small.
options(repr.plot.width = 12, repr.plot.height = 10)
plot(sampleTree, main = "Sample clustering to detect outliers", sub="", xlab="",
     cex.lab = 1.2, cex.axis = 1.5, cex.main = 2)
# Plot a line to show the cut
abline(h = 3000000, col = "red");
```



On va couper le cluster pour retirer l'echantillon aberrant

```
# Determine cluster under the line
clust = cutreeStatic(sampleTree, cutHeight = 3000000, minSize = 10)
table(clust)
```

```
clust
 0  1
 1 17
```

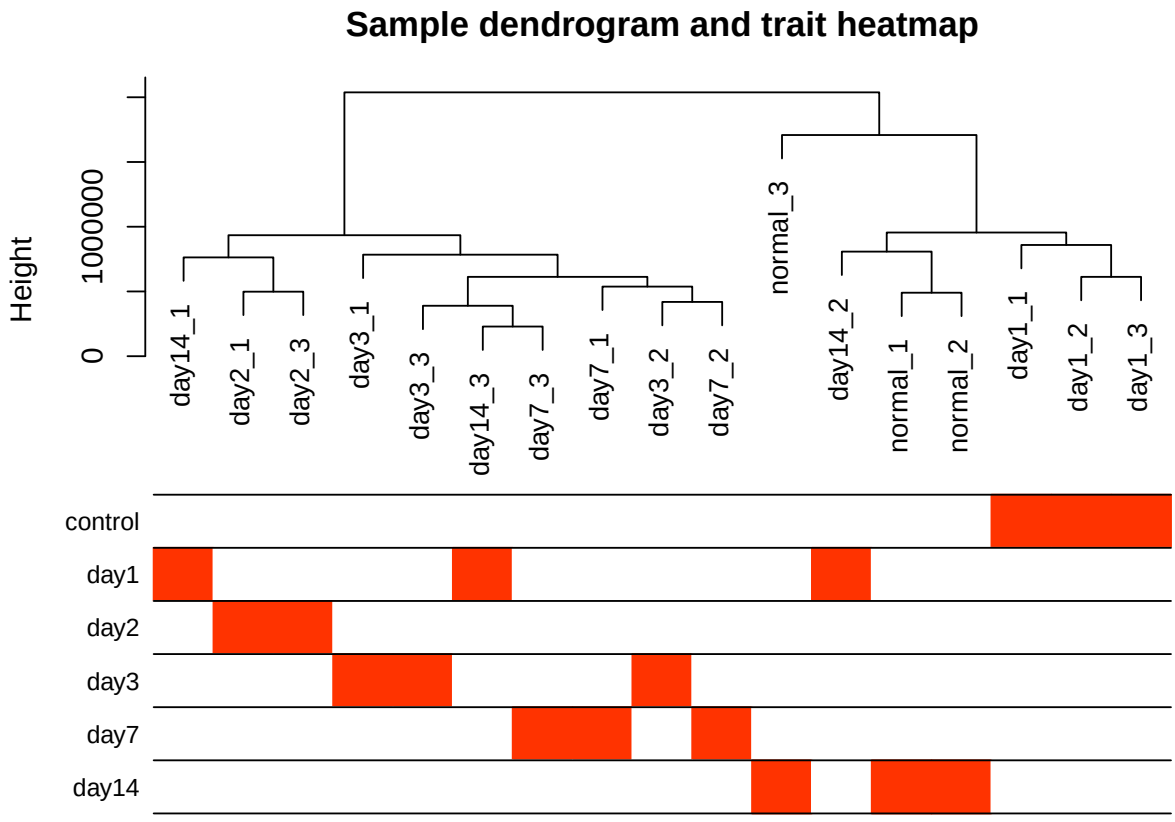
```
# clust 1 contains the samples we want to keep.
keepSamples = (clust==1)
datExpr = fa_cv[keepSamples, ]
```

```
nGenes = ncol(datExpr)
nSamples = nrow(datExpr)
#head(datExpr)
#class(datExpr)
datExpr <- as.data.frame(datExpr)
#head(datExpr)
# datExpr_view <- datExpr[,1:10]
# datExpr_view
```

Import trait table

Liaison échantillons et mesures de phénotype avec la table “trait” (analyse binaire)

```
# Re-cluster samples
sampleTree2 = hclust(dist(datExpr), method = "average")
# Convert traits to a color representation: white means low, red means high, grey means missing entry
traitColors = numbers2colors(fa_trait, signed = FALSE);
# Plot the sample dendrogram and the colors underneath.
options(repr.plot.width = 15, repr.plot.height = 12)
plotDendroAndColors(sampleTree2, traitColors,
                    groupLabels = names(fa_trait),
                    main = "Sample dendrogram and trait heatmap")
```



```
# Allow multi-threading within WGCNA. This helps speed up certain calculations.
# At present this call is necessary for the code to work.
# Any error here may be ignored but you may want to update WGCNA if you see one.
# See note above.
allowWGCNAThreads()
```

```
Allowing multi-threading with up to 56 threads.
```

```
# Load the data saved in the first part
lnames = load(file = "fa-dataInput.RData");
#The variable lnames contains the names of loaded variables.
lnames
```

```
[1] "datExpr"  "fa_trait"
```

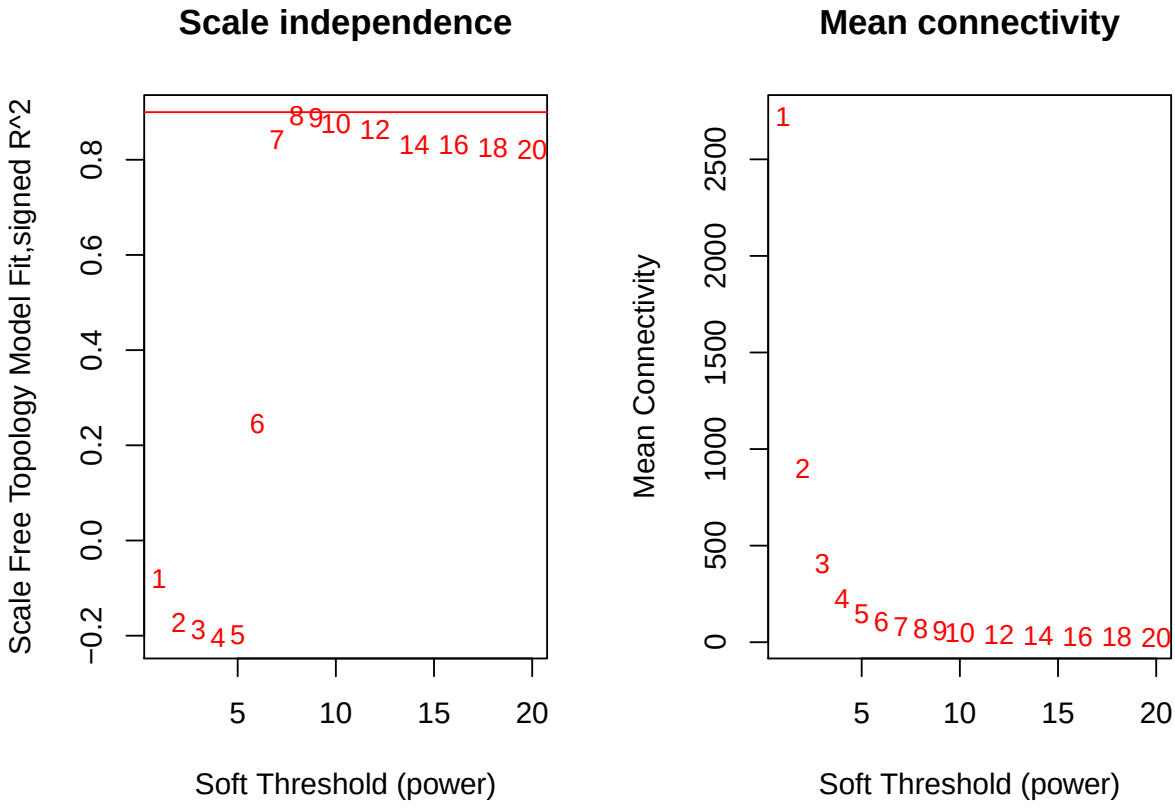
Détection des modules et construction du réseau WGCNA

```
# Choose a set of soft-thresholding powers
powers = c(c(1:10), seq(from = 12, to = 20, by = 2)) #on prend de 12 à 20, 2 en 2
# Call the network topology analysis function
sft = pickSoftThreshold(datExpr, powerVector = powers, verbose = 5)
```

```
pickSoftThreshold: will use block size 3218.
 pickSoftThreshold: calculating connectivity for given powers...
   ..working on genes 1 through 3218 of 13899
   ..working on genes 3219 through 6436 of 13899
   ..working on genes 6437 through 9654 of 13899
   ..working on genes 9655 through 12872 of 13899
   ..working on genes 12873 through 13899 of 13899
   Power SFT.R.sq  slope truncated.R.sq mean.k. median.k. max.k.
1      1   0.0801  3.010          0.7320  2720.0   2730.00   3590
2      2   0.1710  3.880          0.4230   899.0    886.00   1410
3      3   0.1870  3.610          0.0882   406.0    399.00    714
4      4   0.2040  2.900         -0.0127   228.0    212.00    429
5      5   0.1980  0.439          0.7670   148.0    125.00    329
6      6   0.2450 -0.394          0.3250   107.0     79.20    293
7      7   0.8430 -0.848          0.7980    82.9     53.00    269
8      8   0.8920 -0.914          0.8850    67.5     36.80    252
9      9   0.8880 -0.929          0.8930    57.0     26.30    239
10    10   0.8760 -0.933          0.8750    49.5     19.20    229
11    12   0.8640 -0.912          0.8800    39.6     10.80    213
12    14   0.8320 -0.909          0.8230    33.4      6.43    203
13    16   0.8310 -0.888          0.8280    29.2      3.97    194
14    18   0.8260 -0.869          0.8310    26.1      2.52    188
15    20   0.8200 -0.857          0.8250    23.8      1.66    183
```

```
# Plot the results:
par(mfrow = c(1, 2));
options(repr.plot.width = 14, repr.plot.height = 10);
# Scale-free topology fit index as a function of the soft-thresholding power
plot(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
     xlab = "Soft Threshold (power)", ylab = "Scale Free Topology Model Fit,signed R^2", type = "n",
```

```
     main = paste("Scale independence"));
text(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
     labels = powers, cex = 0.9, col = "red");
# this line corresponds to using an R^2 cut-off of h
abline(h = 0.90, col = "red")
# Mean connectivity as a function of the soft-thresholding power
plot(sft$fitIndices[,1], sft$fitIndices[,5],
     xlab = "Soft Threshold (power)", ylab = "Mean Connectivity", type = "n",
     main = paste("Mean connectivity"))
text(sft$fitIndices[,1], sft$fitIndices[,5], labels = powers, cex = 0.9, col = "red")
```



On voit qu'à partir de puissance 7 on atteint le seuil donc pas besoin d'aller jusqu'à puissance 20 on va mettre matrice de corrélation à la puissance 8

Graph 2: quand on augmete la puissance on perd des corrélations

Adjacency matrix

Ci dessous, je choisis 8 comme puissance la plus basse qui construit la scale free topology. Ensuite, la fonction de génere des modules de taille 30, et fusionne les modules similaires à plus de 25% et enregistre la matrice de chevauchement topologique dans un objet.

```r
#coupe le jeu de données en differents blocks pour eviter methode de calculation plus lourde
net = blockwiseModules(datExpr, power = 8,
                       TOMType = "signed", minModuleSize = 30,
                       reassignThreshold = 0.5, mergeCutHeight = 0.25,
                       numericLabels = TRUE, pamRespectsDendro = FALSE,
                       saveTOMs = TRUE, nThreads = 8,
                       saveTOMFileBase = "fa_TOM",
                       verbose = 3)
```

```
 Calculating module eigengenes block-wise from all genes
   Flagging genes and samples with too many missing values...
    ..step 1
  ..Excluding 128 genes from the calculation due to too many missing samples or zero variance.
    ..step 2
 ....pre-clustering genes to determine blocks..
   Projective K-means:
   ..k-means clustering..
   ..merging smaller clusters...
Block sizes:
gBlocks
   1    2    3
4789 4702 4280
 ..Working on block 1 .
    TOM calculation: adjacency..
    ..will use 8 parallel threads.
     Fraction of slow calculations: 0.000000
    ..connectivity..
    ..matrix multiplication (system BLAS)..
    ..normalization..
    ..done.
   ..saving TOM for block 1 into file fa_TOM-block.1.RData
 ....clustering..
 ....detecting modules..
 ....calculating module eigengenes..
 ....checking kME in modules..
     ..removing 4 genes from module 1 because their KME is too low.
     ..removing 17 genes from module 2 because their KME is too low.
     ..removing 13 genes from module 3 because their KME is too low.
     ..removing 15 genes from module 4 because their KME is too low.
     ..removing 1 genes from module 5 because their KME is too low.
     ..removing 2 genes from module 6 because their KME is too low.
     ..removing 1 genes from module 8 because their KME is too low.
 ..Working on block 2 .
    TOM calculation: adjacency..
    ..will use 8 parallel threads.
     Fraction of slow calculations: 0.000000
    ..connectivity..
    ..matrix multiplication (system BLAS)..
    ..normalization..
    ..done.
   ..saving TOM for block 2 into file fa_TOM-block.2.RData
 ....clustering..
 ....detecting modules..
```

```
....calculating module eigengenes..
....checking kME in modules..
    ..removing 12 genes from module 2 because their KME is too low.
    ..removing 1 genes from module 3 because their KME is too low.
    ..removing 10 genes from module 4 because their KME is too low.
    ..removing 21 genes from module 5 because their KME is too low.
    ..removing 8 genes from module 6 because their KME is too low.
..Working on block 3 .
   TOM calculation: adjacency..
   ..will use 8 parallel threads.
    Fraction of slow calculations: 0.000000
   ..connectivity..
   ..matrix multiplication (system BLAS)..
   ..normalization..
   ..done.
  ..saving TOM for block 3 into file fa_TOM-block.3.RData
....clustering..
....detecting modules..
....calculating module eigengenes..
....checking kME in modules..
    ..removing 5 genes from module 1 because their KME is too low.
    ..removing 22 genes from module 2 because their KME is too low.
    ..removing 4 genes from module 3 because their KME is too low.
    ..removing 3 genes from module 4 because their KME is too low.
    ..removing 10 genes from module 5 because their KME is too low.
    ..removing 1 genes from module 6 because their KME is too low.
 ..reassigning 146 genes from module 1 to modules with higher KME.
 ..reassigning 109 genes from module 2 to modules with higher KME.
 ..reassigning 99 genes from module 3 to modules with higher KME.
 ..reassigning 107 genes from module 4 to modules with higher KME.
 ..reassigning 158 genes from module 5 to modules with higher KME.
 ..reassigning 128 genes from module 6 to modules with higher KME.
 ..reassigning 37 genes from module 7 to modules with higher KME.
 ..reassigning 26 genes from module 8 to modules with higher KME.
 ..reassigning 4 genes from module 9 to modules with higher KME.
 ..reassigning 2 genes from module 10 to modules with higher KME.
 ..reassigning 195 genes from module 11 to modules with higher KME.
 ..reassigning 139 genes from module 12 to modules with higher KME.
 ..reassigning 143 genes from module 13 to modules with higher KME.
 ..reassigning 129 genes from module 14 to modules with higher KME.
 ..reassigning 94 genes from module 15 to modules with higher KME.
 ..reassigning 45 genes from module 16 to modules with higher KME.
 ..reassigning 5 genes from module 17 to modules with higher KME.
 ..reassigning 5 genes from module 18 to modules with higher KME.
 ..reassigning 2 genes from module 19 to modules with higher KME.
 ..reassigning 122 genes from module 20 to modules with higher KME.
 ..reassigning 129 genes from module 21 to modules with higher KME.
 ..reassigning 87 genes from module 22 to modules with higher KME.
 ..reassigning 69 genes from module 23 to modules with higher KME.
 ..reassigning 129 genes from module 24 to modules with higher KME.
 ..reassigning 55 genes from module 25 to modules with higher KME.
..merging modules that are too close..
    mergeCloseModules: Merging modules whose distance is less than 0.25
      Calculating new MEs...
```

Nombre et taille des modules

```
table(net$colors)
```

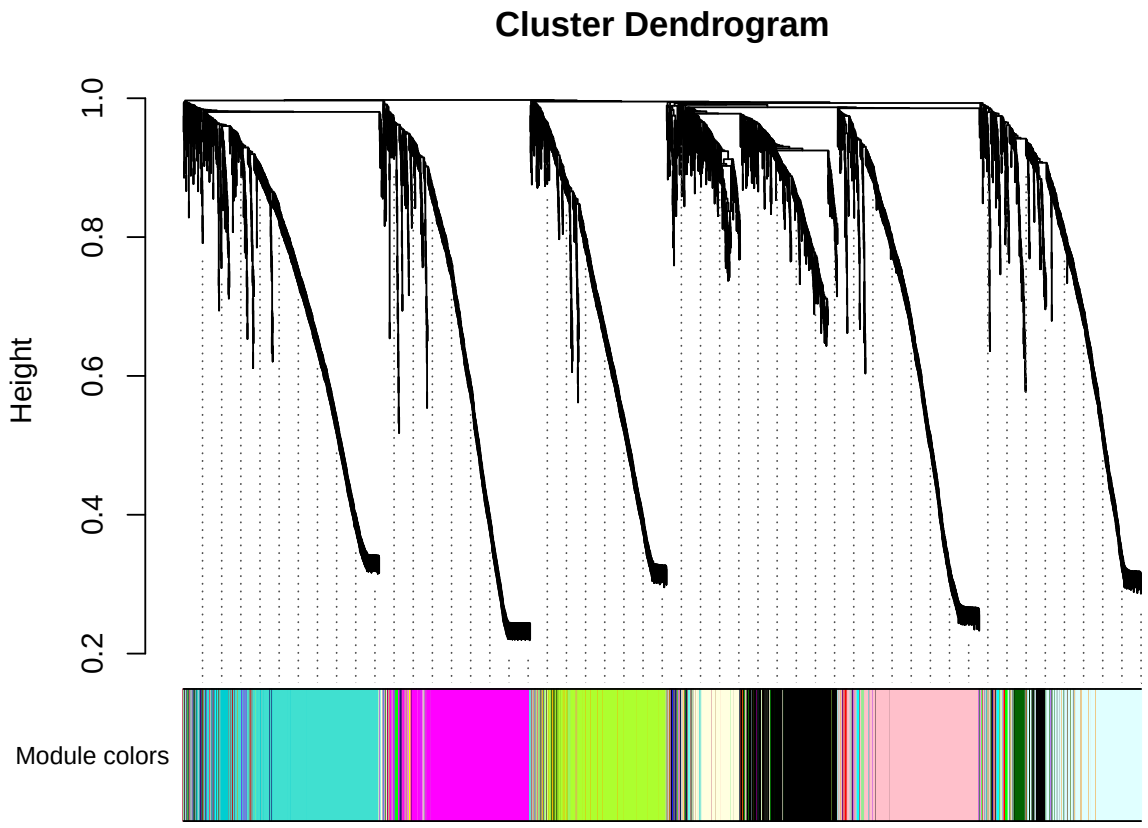
```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25
299 802 796 780 755 735 730 714 696 687 684 674 672 667 638 613 571 521 504 397 186 179 132 129 128 109
```

Il y a **26 modules**

Cluster Dendrogram

Ci-dessous la représentation des modules et du clustering des gènes

```
# Convert labels to colors for plotting
mergedColors = labels2colors(net$colors)
#mergedColors

# Plot the dendrogram and the module colors underneath
plotDendroAndColors(net$dendrograms[[1]], mergedColors[net$blockGenes[[1]]],
                    "Module colors",
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05)
```



Sauvegarde des résultats en .RData

```
#on transforme en couleurs

moduleLabels = net$colors
moduleColors = labels2colors(net$colors)
MEs = net$MEs;
head(MEs)
```

```
              ME10        ME13        ME18         ME9         ME1        ME19        ME12        ME23
day1_1  -0.07238678 -0.05224553 -0.04349821 -0.05329046 -0.07830409  0.4157200  0.96798795  0.74174292
day1_2  -0.04933208  0.96786593 -0.01574374 -0.05890943 -0.04434104  0.3920358 -0.03757649 -0.07550500
day1_3  -0.04975890 -0.01365559  0.96835907 -0.02935999 -0.03914447  0.3722032 -0.02576257 -0.07280541
day14_1 -0.03771846 -0.07454078 -0.05989101 -0.08794690 -0.08017217 -0.2358184 -0.05945834 -0.12446948
day14_2 -0.08620551 -0.08277742 -0.06936665 -0.07724318 -0.09439856 -0.2470120 -0.06524764 -0.08311593
day14_3 -0.03781602 -0.06884687 -0.07689919 -0.06888207 -0.05307051 -0.2408391 -0.07299399 -0.14055045
              ME7         ME8         ME0
day1_1  -0.1152950 -0.06506602 -0.01723925
day1_2  -0.1740444 -0.07430235 -0.28786238
day1_3  -0.1351143 -0.05630494 -0.17468333
day14_1  0.1162694 -0.05588556  0.17096996
day14_2  0.1866544 -0.04035597  0.35816433
day14_3 -0.1289810 -0.06817616  0.10376559
```

```
geneTree = net$dendrograms[[1]];
save(MEs, moduleLabels, moduleColors, geneTree,
     file = "Transcriptomique-networkConstruction-auto.RData")
```

```
lnames = load(file = "fa-dataInput.RData");
#The variable lnames contains the names of loaded variables.
lnames
```

```
[1] "datExpr"  "fa_trait"
```

```
# Load network data saved in the second part.
lnames = load(file = "Transcriptomique-networkConstruction-auto.RData");
lnames
```

```
[1] "MEs"          "moduleLabels" "moduleColors" "geneTree"
```

Quantification des associations module-trait

Les modules qui sont significativement associés aux traits cliniques sont mesurés. Nous avons déjà un profil de synthèse calculé (eigengene) pour chaque module, donc nous corrélons simplement les eigengènes avec des traits phénotypiques et recherchons les associations les plus significatives:

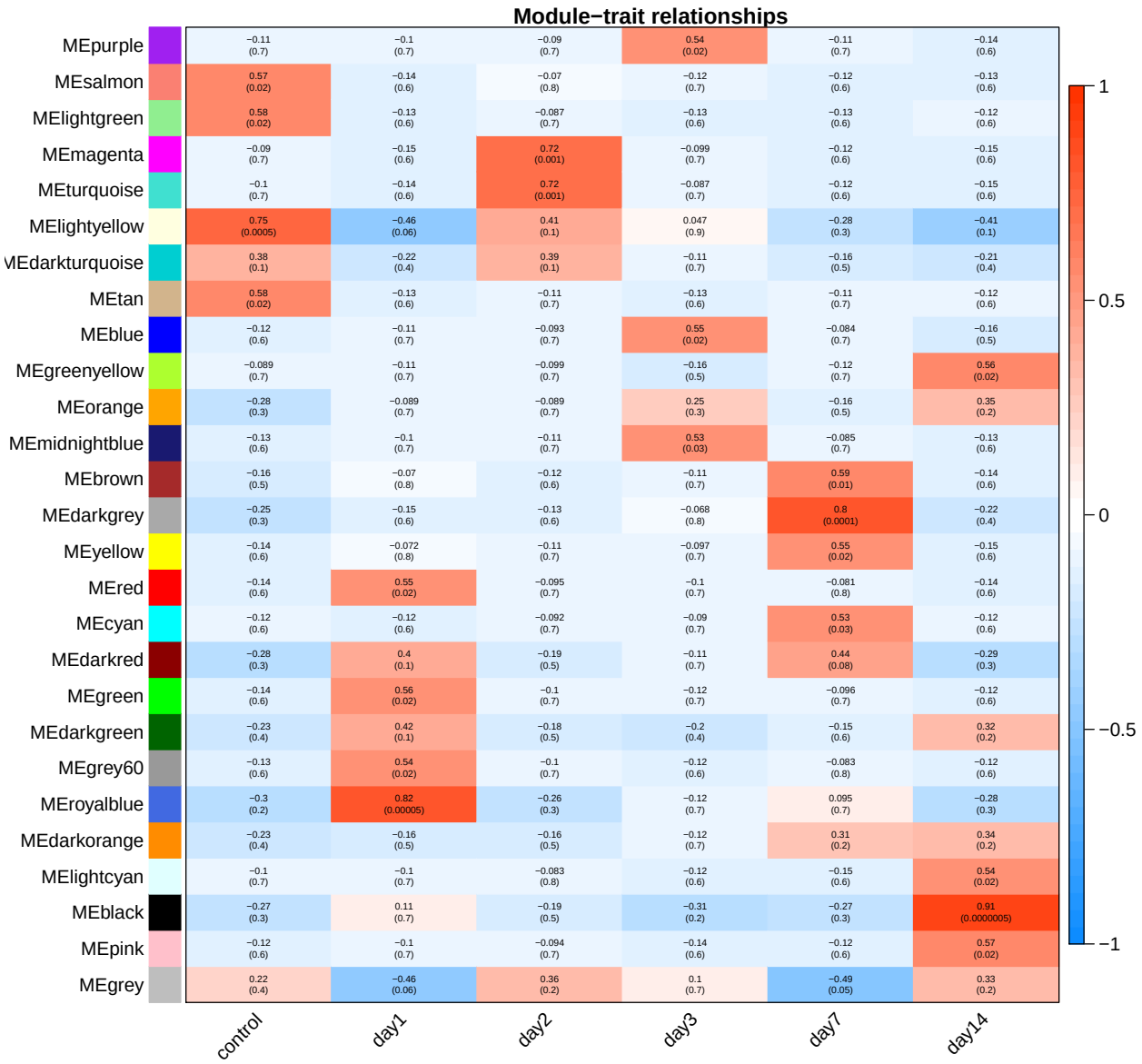
```
# fa_meta
# trait <- c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0)
# fa_metafinal <- data.frame(fa_meta, trait)
# fa_metafinal
#
# fa_metafinal <- fa_metafinal[-8,]
```

```
# fa_metafinal


# Define numbers of genes and samples
nGenes = ncol(datExpr);
nSamples = nrow(datExpr);
# Recalculate MEs with color labels
MEs0 = moduleEigengenes(datExpr, moduleColors)$eigengenes
MEs = orderMEs(MEs0)
moduleTraitCor = cor(MEs, fa_trait, use = "p");
moduleTraitPvalue = corPvalueStudent(moduleTraitCor, nSamples);
```

Heatmap relation module/trait

Représentation de chaque module eigengene et son coefficient de corrélation.

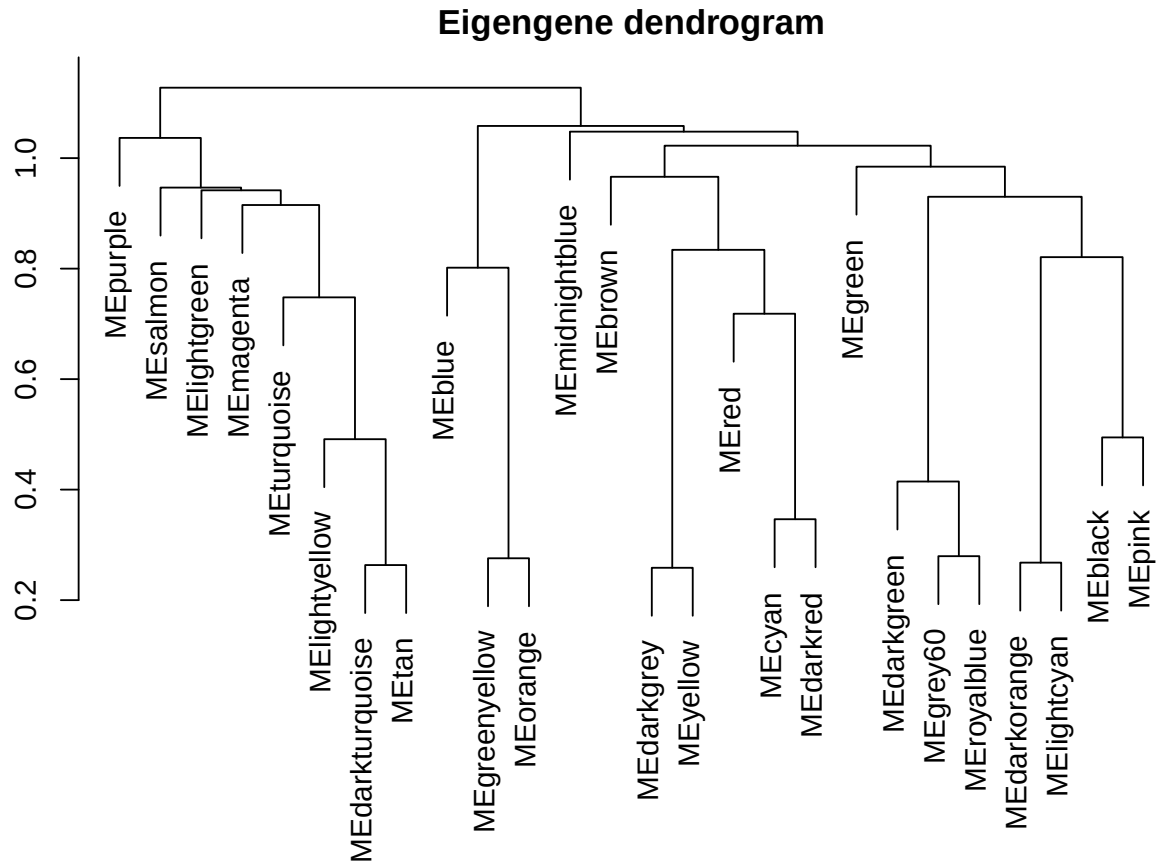
```
# Will display correlations and their p-values
textMatrix =  paste(signif(moduleTraitCor, 2), "\n(",
                           signif(moduleTraitPvalue, 1), ")", sep = "");
dim(textMatrix) = dim(moduleTraitCor)
par(mar = c(6, 8, 1, 1));
# Display the correlation values within a heatmap plot
labeledHeatmap(Matrix = moduleTraitCor,
               xLabels = names(fa_trait),
               yLabels = names(MEs),
               ySymbols = names(MEs),
               colorLabels = FALSE,
               colors = blueWhiteRed(50),
               textMatrix = textMatrix,
               setStdMargins = FALSE,
               cex.text = 0.5,
               zlim = c(-1,1),
               main = paste("Module-trait relationships"))
```



Correlation entre la matrice de eigengene MEs et la matrice de datTraits On peut voir les fortes correlations rouge foncé ou bleu foncé

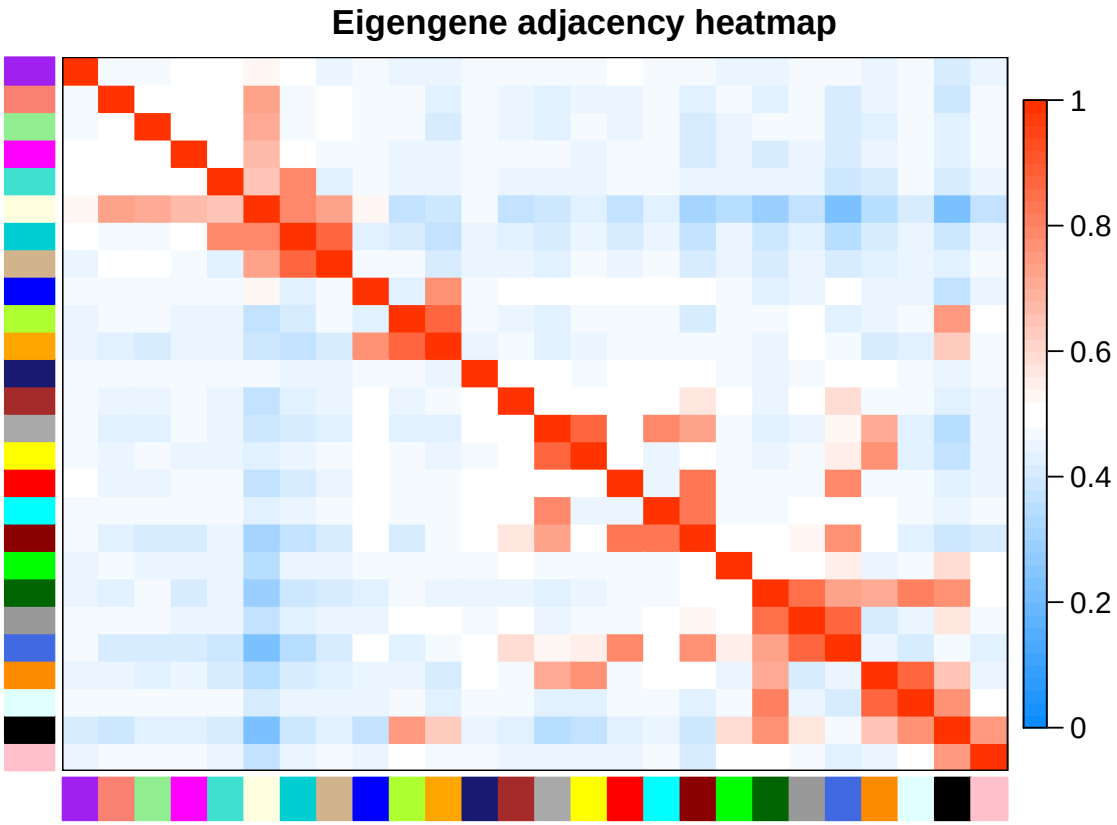
Eigengene view

```
plotEigengeneNetworks(MEs, "Eigengene dendrogram", marDendro = c(0,4,2,0),plotHeatmaps = FALSE)
```



Heatmap matrix

```
# Plot the heatmap matrix (note: this plot will overwrite the dendrogram plot)
par(cex = 1.0)
plotEigengeneNetworks(MEs, "Eigengene adjacency heatmap", marHeatmap = c(3,4,2,2),plotDendrograms = FALS
```



Annotation

```
#names(datExpr)
```

```
# annot <- fa_annotation
# head(fa_annotation)
# dim(annot)
# names(annot)
#
# dim(datExpr)
#
# probes = names(datExpr)
# probes2annot = match(probes, annot$external_gene_name)
# # The following is the number or probes without annotation:
# sum(is.na(probes2annot))
# # Should return 0.
# #head(annot)
# #head(datExpr)
# length(rownames(annot))
# length(colnames(datExpr))
# length(intersect(rownames(annot), colnames(datExpr))) ## Their intersection contains the same number o


# tdataexpr <- t(datExpr)
```

```
# tdataexpr <- as.data.frame(tdataexpr)
#
# class(annot)
# class(tdataexpr)
#
# head(annot)
# head(tdataexpr)
#
# dim(annot)
# dim(tdataexpr)
# annotbis <- annot[intersect(rownames(annot), rownames(tdataexpr)), ]
# rownames(annot)
# rownames(tdataexpr)
#
# head(rownames(annot))
# head(intersect(rownames(annot), rownames(tdataexpr)))
# annot["ENSMUSG00000000001", ]
# annot[head(intersect(rownames(annot), rownames(tdataexpr))), ]

# annotbis <- annot[(intersect(rownames(annot), rownames(tdataexpr))), ]
# dim(annotbis)
```

Export Cytoscape

```
# Recalculate topological overlap if needed
TOM=TOMsimilarityFromExpr(datExpr, power=8)
```

```
TOM calculation: adjacency..
..will use 56 parallel threads.
 Fraction of slow calculations: 0.000000
..connectivity..
..matrix multiplication (system BLAS)..
..normalization..
..done.
```

```
# Read in the annotation file
#annot=read.csv(file="data/GeneAnnotation.csv")
# Select modules
modules=c("darkgrey","grey")
# Select module probes
probes=colnames(datExpr)
inModule=is.finite(match(moduleColors, modules))
modProbes=probes[inModule]
#modGenes=annot$gene_symbol[match(modProbes, annot$substanceBXH)]
# Select the corresponding Topological Overlap
modTOM=TOM[inModule, inModule]
dimnames(modTOM)=list(modProbes, modProbes)
# Export the network into edge and node list files Cytoscape can read
cyt=exportNetworkToCytoscape(modTOM,
                        edgeFile=paste("CytoscapeInput-edges-",paste(modules, collapse="-"),".txt", sep
                        nodeFile=paste("CytoscapeInput-nodes-",paste(modules, collapse="-"),".txt", sep
```

```
                        weighted=TRUE,
                        threshold=0.5,
                        nodeNames=modProbes)
                        #altNodeName
```

Je ne comprends pas pourquoi mes exports nodes et edges sont vides donc pas possible de passer à Cytoscape

Mémo

Steps in WGCNA are: 1. Similarity co-expression matrix (Pearson's correlation). 2. Adjacency matrix (signed, unsigned, signed-hybrid). 3. Topological Overlap Matrix (TOM) constructed from adjacency matrix. (TOM formula is presented in below image) 4. Dissimilarity matrix (1-TOM) obtained from TOM. *By using TOM matrix, the network in WGCNA assumes not only co-expression information, also it assumes the network topological information.

Session info

```
#### Session info ####
sessionInfo()
```

```
R version 4.0.3 (2020-10-10)
Platform: x86_64-conda-linux-gnu (64-bit)
Running under: CentOS Linux 7 (Core)

Matrix products: default
BLAS/LAPACK: /shared/ifbstor1/software/miniconda/envs/r-4.0.3/lib/libopenblasp-r0.3.10.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.U
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] parallel  stats4    stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] tidyr_1.1.3                 WGCNA_1.70-3                fastcluster_1.1.25          dynamicTreeCut
[11] Biobase_2.50.0              MatrixGenerics_1.2.1        matrixStats_0.58.0          GenomicRanges_
[21] gprofiler2_0.2.0            factoextra_1.0.7            ggplot2_3.3.3               FactoMineR_2.4

loaded via a namespace (and not attached):
  [1] colorspace_2.0-1       ellipsis_0.3.2         htmlTable_2.2.1        corpcor_1.6.9          XVecto
 [13] RSpectra_0.16-0        fansi_0.4.2            codetools_0.2-18       splines_4.0.3          leaps_
 [25] cluster_2.1.1          GO.db_3.12.1           png_0.1-7              compiler_4.0.3         httr_
 [37] igraph_1.2.6           gtable_0.3.0           glue_1.4.2             GenomeInfoDbData_1.2.4 reshap
 [49] XML_3.99-0.6           zlibbioc_1.36.0        scales_1.1.1           RColorBrewer_1.1-2     yaml_
 [61] genefilter_1.72.1      foreach_1.5.1          checkmate_2.0.0        BiocParallel_1.24.1    rlang_
 [73] tidyselect_1.1.1       plyr_1.8.6             magrittr_2.0.1         R6_2.5.0               gener
 [85] survival_3.2-10        scatterplot3d_0.3-41   RCurl_1.98-1.3         crayon_1.4.1           rARPA
 [97] data.table_1.14.0      blob_1.2.1             digest_0.6.27          flashClust_1.01-2      xtabl
```