Analyse des données génomiques 2020
Sélection de gènes associés à un signal biologique d'intérêt

Sandrine Lagarrigue et David Causeur

Nom Prénom :

Dans la suite, on reprend les étapes de sélection de gènes décrite dans le podcast video. **Modifiez les commandes ci-après afin de les adapter au sujet de votre étude de cas.** En particulier,

- les données d'expression pouront concerner un autre tissu que celui du foie (muscle ou tissu adipeux) ;
- de même, les facteurs de variation des profils d'expression pourront ne pas être limités au régime et au génotype. En effet, ils pourront aussi être choisis dans le tableau de données supplémentaires sur les poulets également disponible (profils d'acide gras, ...).

1 Préparation de la session de travail

La commande d'installation ci-après du package *limma* à partir de la plateforme *bioconductor* ne doit être exécutée que si le package n'est pas déjà installé.

```
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")

BiocManager::install("limma", update = FALSE)
```

Une fois installé, le package doit être chargé dans la session de travail :

```
require(limma)
```

De même, la commande d'installation ci-après du package *FAMT* ne doit être exécutée que si le package n'est pas déjà installé.

```
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")

BiocManager::install("impute", update = FALSE)

install.packages("FAMT")
```

Une fois installé, le package doit être chargé dans la session de travail :

```
require(FAMT)
```

Enfin, la commande d'installation ci-après du package *fdrtool* ne doit être exécutée que si le package n'est pas déjà installé.

```
install.packages("fdrtool")
```

Une fois installé, le package doit être chargé dans la session de travail :

```
require(fdrtool)
```

2 Importation des données

2.1 Données d'expression

On peut importer les données d'expression au format gènes x microarray grâce à la fonction *read.table* :

```
## Import gene expression data
expressions = read.table("foie.txt", header = TRUE)
dim(expressions)  # Numbers of rows and columns
```

```
[1] 46060    46
```

```
head(expressions[, 1:10])  # Displays first 6 rows of first 10 columns
```

```
                                F_01    F_02    F_03   F_04   F_05   F_06
seq_RIGG02544                  5.171  4.7836  4.7823  5.018  5.266  5.398
A_87_P009088                  -2.260 -2.2637 -2.3401 -2.156 -2.341 -2.371
A_87_P113528                   5.852  5.5067  5.6472  5.776  5.792  5.852
T001527_G001030_SCAMP4_420081  0.203 -0.0378 -0.0255  0.174 -0.154 -0.354
A_87_P023872                   5.015  5.0264  5.0499  4.854  4.619  4.921
T041025_G024289               -1.311 -1.3476 -0.8141 -1.090 -1.488 -1.424
                                F_07   F_08   F_09    F_10
seq_RIGG02544                  4.825  5.481  5.296  5.0215
A_87_P009088                  -2.110 -2.120 -2.111 -1.8170
A_87_P113528                   5.643  5.528  5.825  5.7134
T001527_G001030_SCAMP4_420081 -0.327 -0.180  0.109 -0.0949
A_87_P023872                   4.745  4.892  4.942  4.9657
T041025_G024289               -1.795 -0.984 -0.623 -0.8102
```

2.2 Dispositif expérimental

Toute l'information sur le dispositif expérimental est contenue dans le fichier 'covariates.txt'. On peut importer ce fichier grâce à la fonction *read.table* :

```
covariates = read.table("covariates.txt", header = TRUE)
dim(covariates)  # Numbers of rows and columns
```

```
[1] 46  2
```

```
str(covariates)  # Overview of the data
```

```
'data.frame':   46 obs. of  2 variables:
 $ Diet    : Factor w/ 2 levels "BL","HL": 1 1 2 2 2 2 1 1 1 1 ...
 $ Genotype: Factor w/ 2 levels "G","M": 2 1 2 1 1 2 1 2 2 1 ...
```

Si votre étude de cas consiste à étudier les relations entre l'expression des gènes et une des variables supplémentaires décrivant notamment les profils d'acide gras des poulets, vous devez importer ici les données du fichier *external.txt*, en procédant de la même manière que ci-dessus pour *covariates.txt*.

3 Tests d'association entre expression et variables expérimentales

Pour automatiser les tests d'association à l'échelle du génome, on utilise le package *FAMT*. Pour commencer, on crée un objet *FAMTdata* qui contient à la fois les données d'expression et les données avec lesquelles on souhaite tester l'association :

```
# First, create an FAMTdata object gathering expressions and covariates
# Column names of expressions should correspond to an ID variable in
# covariates An ID variable is added in 1st column of covariates
covariates = data.frame(ID = rownames(covariates), covariates)

# as.FAMT data creates the FAMTdata object - idcovar gives the column number
# of the ID variable in covariates
foie.famt = as.FAMTdata(expressions, covariates, idcovar = 1)
```

```
$`Rows with missing values`
integer(0)

$`Columns with missing values`
integer(0)
```

```
# Overview of data in the FAMTdata object
summaryFAMT(foie.famt)
```

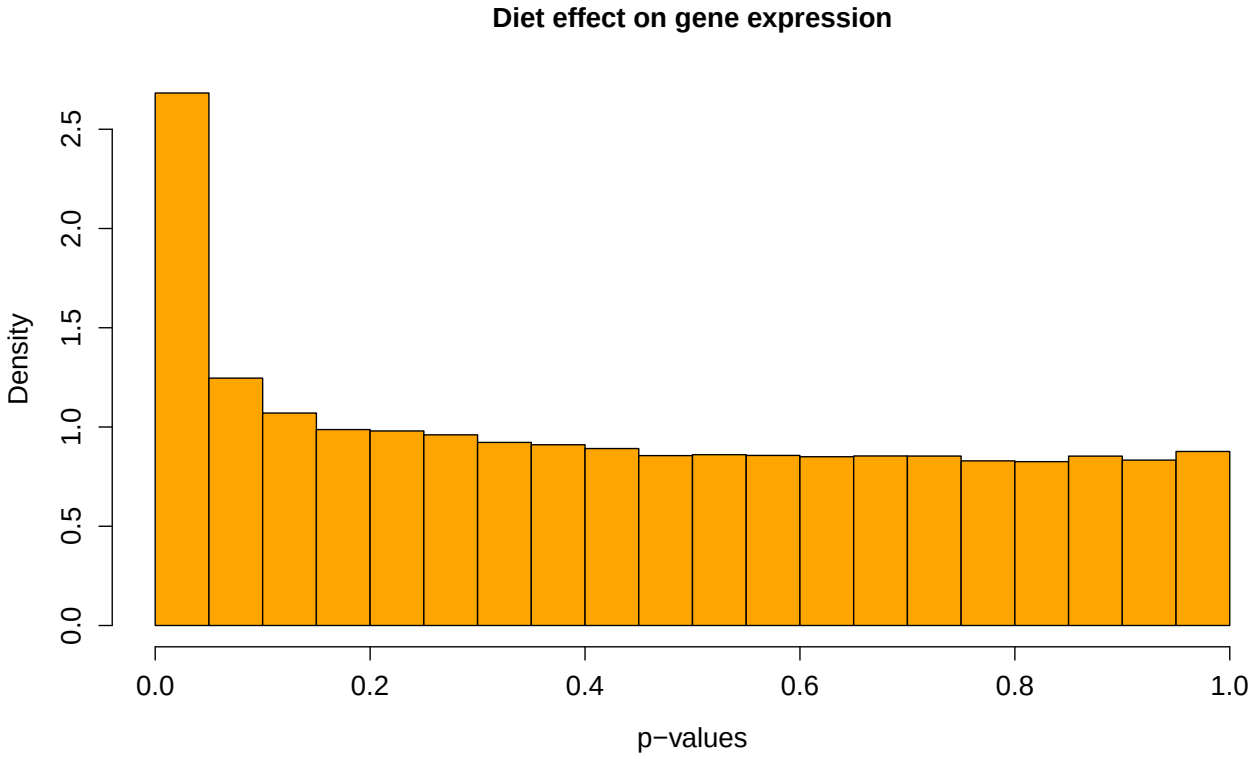
```
$expression
$expression$`Number of tests`
[1] 46060

$expression$`Sample size`
[1] 46


$covariates
       ID     Diet    Genotype
 F_01   : 1   BL:22   G:22
 F_02   : 1   HL:24   M:24
 F_03   : 1
 F_04   : 1
 F_05   : 1
 F_06   : 1
 (Other):40

$annotations
             ID
 A_87_P000029:    1
 A_87_P000046:    1
 A_87_P000062:    1
 A_87_P000138:    1
 A_87_P000163:    1
 A_87_P000174:    1
 (Other)     :46054
```

Dans le cas présent, on s'intéresse à l'effet sur l'expression des gènes du régime ajusté de l'effet génotype :

```
# Fits the linear model: here, Diet+Genotype (x = 2nd and 3rd columns of
# covariates) Test for the Diet effect (test = 2nd column of covariates) nbf
# = 0 for a standard fit
foie.lmfit = modelFAMT(foie.famt, x = c(2, 3), test = 2, nbf = 0)
```

On récupère maintenant les p-values du test de la manière suivante :

```
# p-values for the diet effect can be found in $pval
pvalD = foie.lmfit$pval
hist(pvalD, main = "Diet effect on gene expression", proba = TRUE, col = "orange",
    xlab = "p-values", cex.lab = 1.25, cex.axis = 1.25, cex.main = 1.25)
```



On détermine les gènes positifs en s'assurant que la proportion de faux positifs est de l'ordre de 5% (vous pouvez préférer contrôler le risque de faux positifs en modifiant l'argument *method* de la fonction *p.adjust*) :

```
BHpvalD = p.adjust(pvalD, method = "BH")  # adjusted p-values (BH)
BHPositives = BHpvalD <= 0.05  # TRUE if positive, FALSE if not
sum(BHPositives)  # Number of positive genes
```

```
[1] 1440
```

On affiche ci-après les p-values des 10 gènes pour lesquels l'effet est le plus significatif :

```
sort(pvalD)[1:10]
```

```
                    seq_RIGG13390                     seq_RIGG10144
                         1.45e-15                         1.85e-13
                     A_87_P017262       T009221_G005739_SCD_395706
                         3.98e-13                         4.49e-13
                     A_87_P035286                     A_87_P271093
                         8.67e-13                         2.12e-12
                     A_87_P035264                     A_87_P122623
                         2.19e-12                         3.07e-12
                     A_87_P163123 T034913_G004703_LOC769138_769138
                         3.74e-12                         3.76e-12
```

4 Amélioration de la puissance des tests

On propose ici trois méthodes permettant d'améliorer la puissance des tests.

4.1 Méthode de la q-value

On peut estimer la proportion π_0 des gènes dont l'expression moyenne ne varie pas en fonction du régime grâce à la fonction *pval.estimate.eta0* du package *fdrtool* :

```
pi0 = pval.estimate.eta0(pvalD)  # Estimation of the proportion of true nulls
```

On en déduit une estimation moins biaisée des FDR, ce qui permet d'augmenter la puissance de la procédure de sélection :

```
BHpvalD = p.adjust(pvalD, method = "BH")  # adjusted p-values (BH)
qvaluesD = pi0 * BHpvalD  # q-values
positives = qvaluesD <= 0.05
# Number of positive genes with a control of the FDR at level 0.05
sum(positives)
```

```
[1] 1564
```

La $i$ème q-value estime le FDR si on choisit comme seuil de décision la $i$ème plus petite p-value. Par conséquent, si on souhaite limiter la liste des gènes positifs aux 100 premiers gènes, le FDR de cette liste est estimé de la manière suivante :

```
# Suppose we choose to keep 100 genes: what is the estimated FDR?
sort(qvaluesD)[100]
```

```
A_87_P073746
     1.7e-05
```

4.2 Tests modérés

Le package *limma* permet de mettre en oeuvre les tests (de Fisher ou de Student) modérés. Ce sont prrobablement les tests les plus utilisés pour la sélection de gènes.

Dans un premier temps, on défnit le modèle d'association avec lequel on travaille :

```
# First, set the design matrix
design = model.matrix(~Genotype + Diet, data = covariates)
head(design)
```

```
     (Intercept) GenotypeM DietHL
F_01           1         1      0
F_02           1         0      0
F_03           1         1      1
F_04           1         0      1
F_05           1         0      1
F_06           1         1      1
```

Ensuite, on ajuste ce modèle d'association pour chacun des gènes :

```
# Then, fit the 2-way analysis of variance model
fit = lmFit(expressions, design)
head(fit$coefficients)  # Display the first 6 rows
```

```
                              (Intercept) GenotypeM   DietHL
seq_RIGG02544                       4.863    0.4434  0.19454
A_87_P009088                       -2.134   -0.0731 -0.11068
A_87_P113528                        5.631    0.2306 -0.05024
T001527_G001030_SCAMP4_420081      -0.138    0.1018  0.05046
A_87_P023872                        4.914    0.1808 -0.00373
T041025_G024289                    -0.979    0.2256  0.02819
```

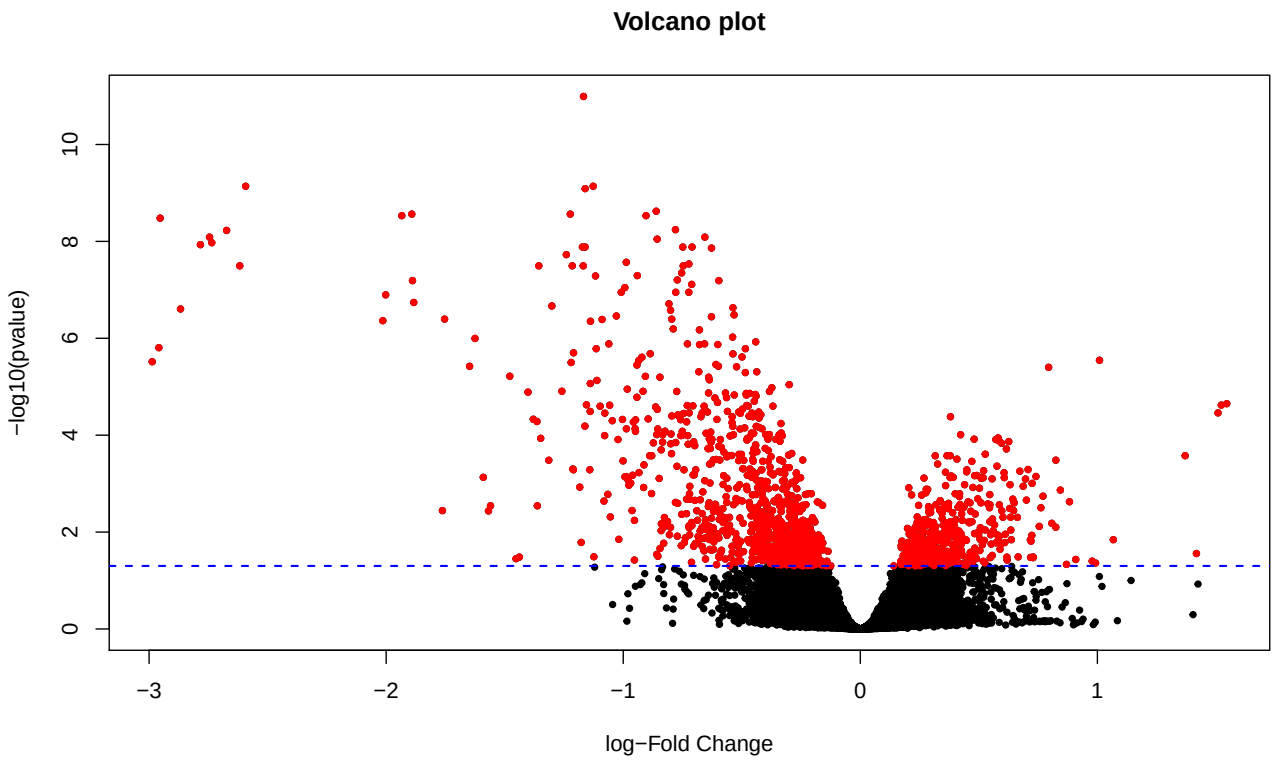
On peut maintenant calculer les statistiques de Fisher modérées et les p-values associées :

```
# Now, calculate the moderated tests statistics and corresponding p-values
fit = eBayes(fit)
# Display the top 10 most significant genes
topTable(fit, coef = 3)  # Coef=column number in fit$coefficients for the test
```

```
                                  logFC AveExpr      t  P.Value adj.P.Val
seq_RIGG13390                    -1.168  -1.137 -12.39 2.22e-16  1.02e-11
seq_RIGG10144                    -1.127  -1.904 -10.73 3.35e-14  7.27e-10
T009221_G005739_SCD_395706       -2.593  -0.184 -10.62 4.74e-14  7.27e-10
A_87_P017262                     -1.161  -1.054 -10.49 7.08e-14  8.15e-10
A_87_P035286                     -0.861   0.392 -10.08 2.59e-13  2.38e-09
A_87_P035264                     -1.223   4.416  -9.95 3.92e-13  2.73e-09
A_87_P122623                     -1.892  -1.873  -9.93 4.15e-13  2.73e-09
T034913_G004703_LOC769138_769138 -1.934  -1.810  -9.87 5.12e-13  2.95e-09
A_87_P271093                     -0.904  -2.886  -9.83 5.76e-13  2.95e-09
A_87_P212618                     -2.953   1.381  -9.76 7.19e-13  3.31e-09
                                    B
seq_RIGG13390                    26.2
seq_RIGG10144                    21.7
T009221_G005739_SCD_395706       21.3
A_87_P017262                     21.0
A_87_P035286                     19.8
A_87_P035264                     19.4
A_87_P122623                     19.3
T034913_G004703_LOC769138_769138 19.1
A_87_P271093                     19.0
A_87_P212618                     18.8
```

Enfin, on peut définir une procédure de sélection basée à la fois sur la p-value avec un contrôle du FDR au niveau 5% (Benjamini-Hochberg) et sur le log-ratio d'expression entre les deux régimes que l'on peut imposer supérieur à 1 (en valeur absolue) :

```
# Volcano plot
BHmoderatedpval = p.adjust(fit$p.value[, "DietHL"], method = "BH")
logFC = fit$coefficients[, 3]

plot(logFC, -log10(BHmoderatedpval), xlab = "log-Fold Change", ylab = "-log10(pvalue)",
    main = "Volcano plot", cex = 0.6, pch = 19)
points(logFC[BHmoderatedpval <= 0.05], -log10(BHmoderatedpval)[BHmoderatedpval <=
    0.05], cex = 0.6, pch = 19, col = "red")
abline(h = -log10(0.05), col = "blue", lty = 2, lwd = 1.5)
```




```
# Condition to be selected: BH adjusted p-value<=0.05 + |logFC|>=1
select = (BHmoderatedpval <= 0.05) & (abs(logFC) >= 1)

# Number of selected genes
sum(select)
```

```
[1] 83
```

4.3 Modèle pour l'hétérogénéité d'expression

La fonction *modelFAMT* détermine le nombre de facteurs d'hétérogénéité dans les données d'expression, estime ces facteurs d'hétérogénéité pour les soustraire aux données d'expresssion et calcule les p-values des tests sur les données ajustées de leur hétérogénéité :

```
# nbf = NULL let the function determine the proper number of factors
foie.lmfit = modelFAMT(foie.famt, x = c(2, 3), test = 2, nbf = NULL)
```

```
[1] "Fitting Factor Analysis Model with 1 factors"
[1] "Fitting Factor Analysis Model with 2 factors"
[1] "Fitting Factor Analysis Model with 3 factors"
[1] "Fitting Factor Analysis Model with 4 factors"
[1] "Fitting Factor Analysis Model with 5 factors"
[1] "Fitting Factor Analysis Model with 6 factors"
[1] "Fitting Factor Analysis Model with 7 factors"
[1] "Fitting Factor Analysis Model with 8 factors"
[1] "Calculating criterion for the model with 0 factors"
[1] "Calculating criterion for the model with 1 factors"
[1] "Calculating criterion for the model with 2 factors"
[1] "Calculating criterion for the model with 3 factors"
[1] "Calculating criterion for the model with 4 factors"
[1] "Calculating criterion for the model with 5 factors"
```

```
[1] "Calculating criterion for the model with 6 factors"
[1] "Calculating criterion for the model with 7 factors"
[1] "Calculating criterion for the model with 8 factors"
```

```
[1] "Fitting Factor Analysis Model with 3 factors"
[1] "Fitting Factor Analysis Model with 3 factors"
```

La fonction *summaryFAMT* permet de mesurer l'impact de cette prise en compte de l'hétérogénéité des données sur la sélection des gènes par la méthode de la q-value (*pi0=NULL*) :

```
# alpha = 0.05 is the FDR control level pi0 = NULL to let the functione
# estimate pi0
results = summaryFAMT(foie.lmfit, alpha = 0.05, pi0 = NULL)
results$pi0
```

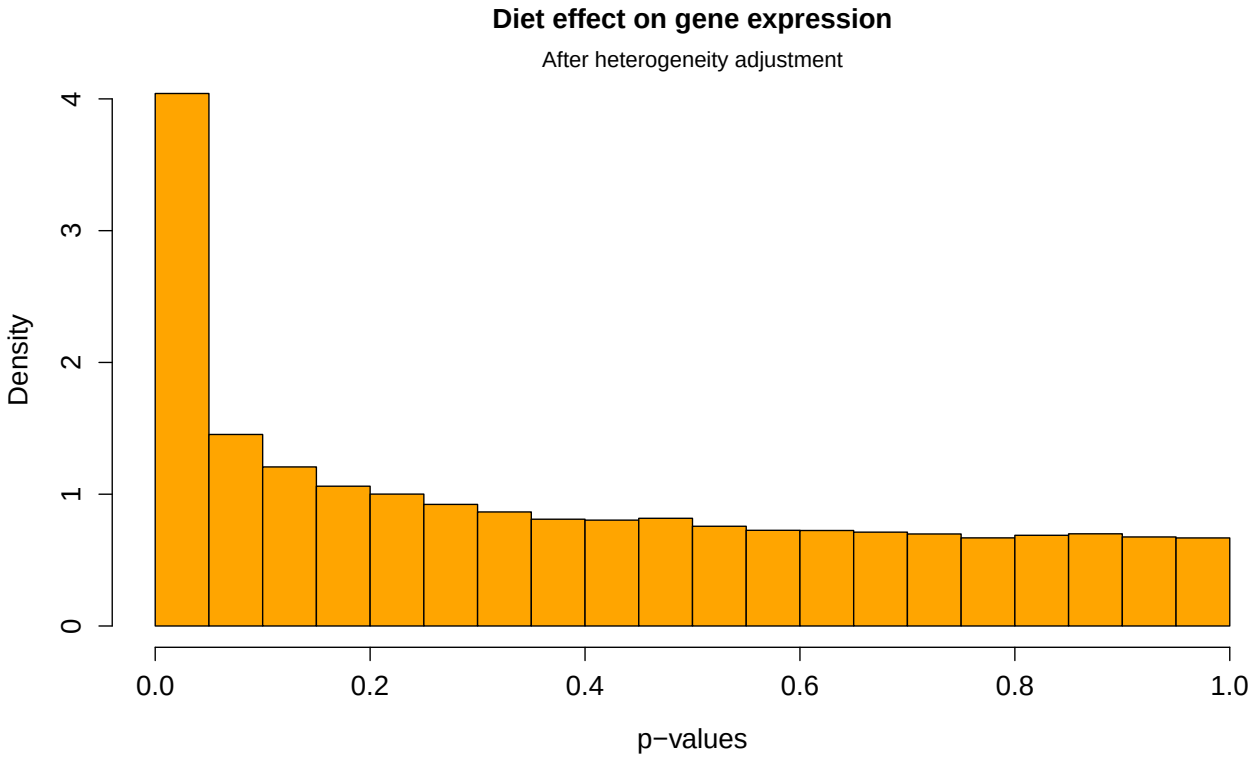
```
[1] 0.668
```

```
results$nbreject
```

```
  alpha Raw analysis FA analysis
1  0.05         1781        4179
```

Enfin, les p-values pour l'effet régime calculées sur les données ajustées de leur hétérogénéité peuvent être récupérées :

```
# Heterogeneity-adjusted p-values for the diet effect can be found in
# $adjpval
pvalD = foie.lmfit$adjpval
hist(pvalD, main = "Diet effect on gene expression", proba = TRUE, col = "orange",
    xlab = "p-values", cex.lab = 1.25, cex.axis = 1.25, cex.main = 1.25)
mtext("After heterogeneity adjustment")
```



Comme précédement avec *limma*, la sélection des gènes peut combiner un contrôle du FDR (méthode de la q-value) et l'exigence de log-ratios d'expression supérieurs à 1 :

```
# Heterogeneity-adjusted p-values for the diet effect can be found in
# $adjpval
qvaluesD = results$pi0 * p.adjust(pvalD, method = "BH")
select = (qvaluesD <= 0.05) & (abs(logFC) >= 1)

# Number of selected genes
sum(select)
```

```
[1] 88
```

5 Création des fichiers de gènes sélectionnés

Il reste à créer et exporter le tableau des données d'expression restreint aux gènes sélectionnés par la méthode choisie :

```
foiePositive = expressions[select, ]
write.table(foiePositive, "foiePositive.txt")
```

On peut aussi exporter le fichier des données d'expressions ajustées de leurs facteurs d'hétérogénéité accessibles dans *foie.lmfit$adjdata$expression*:

```
expressions_adjusted = foie.lmfit$adjdata$expression
foiePositive_adjusted = expressions_adjusted[select, ]
write.table(foiePositive_adjusted, "foiePositiveAdjusted.txt")
```