

Anàlisi de Dades Òmiques PAC2

Oriol Nualart Mundó

30/5/2020

Enllaç al repositori de GitHub associat a la PAC: <https://github.com/onualart/tiroides.git>

CONTINGUTS

- 1. Abstract
 - 2. Objectius
 - 3. Materials i mètodes
 - 3.1. Disseny de l'experiment
 - 3.2. Software utilitzat
 - 3.3. Procediment d'anàlisi de les dades
 - * 3.3.1. Preparació de les dades
 - * 3.3.2. Control de qualitat i filtratge
 - * 3.3.3. Normalització
 - * 3.3.4. Anàlisi d'expressió diferencial
 - * 3.3.5. Anotació dels resultats
 - * 3.3.6. Comparació entre comparacions
 - * 3.3.7. Anàlisi de significació biològica
 - 4. Resultats
 - 5. Discussió
 - Referències
-

1. Abstract

El projecte Genotype-Tissue Expression (GTEx, (GTEx Consortium 2013)) pretén crear una gran base de dades pública amb informació de l'expressió i regulació gènica en 54 teixits, obtinguda de gairebé 1000 individus, i inclou dades de RNA-seq.

En aquesta anàlisi, partim d'una selecció aleatòria de dades de RNA-seq de mostres de tiroides tretes del projecte GTEx, tant de teixit sa com de teixit parcialment i extensament infiltrat, per detectar gens diferencialment expressatas i buscar patrons biològicament significatius.

2. Objectius

L'objectiu d'aquesta anàlisi és identificar gens que s'expressen diferencialment entre els grups de teixits, i a partir d'aquests gens expressats diferencialment, identificar *pathways* alterats en les comparacions.

Aquesta informació pot aportar coneixements sobre els processos subjacents a la infiltració linfoide de la tiroides, i pot ajudar a identificar potencials dianes terapèutiques.

3. Materials i mètodes

3.1. Disseny de l'experiment

Disposem de les dades corresponents a un total de 296 mostres de teixit de tiroides (que formen part del projecte GTEx), de les quals:

- 236 corresponen a teixit no infiltrat (NIT)
- 42 corresponen a teixits amb petits infiltrats focals (SFI)
- 14 corresponen a teixit amb infiltrats limfoides extensos (ELI).

Les mostres corresponen tant a homes com a dones i les dades inclouen el paràmetre del sexe.

Seleccionarem aleatoriament 10 mostres de cada grup, i a partir d'aquests subconjunts farem l'anàlisi. Aquesta inclourà les tres comparacions possibles entre els tres grups.

3.2. Software utilitzat

Fer fer aquesta anàlisi s'ha utilitzat R v3.6.3 (2020-02-29). Els paquets de R i Bioconductor utilitzats inclouen:

- DESeq2 v1.26.0
- AnnotationDbi v1.48.0
- org.Hs.eg.db v3.10.0
- clusterProfiler v3.14.3

3.3. Procediment d'anàlisi de les dades

El procediment detallat amb tot el codi de R utilitzat, juntament amb l'*output* dels fragments de codi, es pot trobar a l'arxiu “pipeline.pdf” al repositori de GitHub, generat a partir de l'arxiu de markdown “pipeline.Rmd”. El codi pressuposa tenir instal·lats aquests paquets.

3.3.1. Preparació de les dades

Partim de dos arxius csv, un d'ells amb la taula de *counts*, amb les mostres com a columnes i els gens com a files, i l'altre amb la taula *targets*, que conté la informació sobre les mostres.

El primer que fem és carregar els arxius *targets* i *counts* com a *data frames* (que anomenem *targets_big* i *counts_big*). Corregim el nom amb què estan etiquetades les mostres a *targets_big* per que el format es correspongui amb el dels noms que apareixen a *counts_big*. (Ja que el que a *targets_big* apareix com a guions, a *counts_big* són punts.)

```
targets_big <- read.csv2("./data/targets.csv", header = TRUE, sep = ",")  
  
counts_big <- read.csv2("./data/counts.csv", header = TRUE, sep = ";")  
  
library(stringr)  
  
sample_names <- targets_big$Sample_Name  
sample_names <- str_replace_all(sample_names, "-", ".")  
targets_big$Sample_Name <- sample_names
```

Un cop tenim els dos *data frames*, fem la selecció aleatòria de les 10 mostres de cada grup que necessitem per a l'anàlisi.

Primer establim la “llavor” que ens permetrà fer reproduïble la selecció pseudoaleatòria amb la funció *set.seed*.

```
set.seed(123)
```

Després utilitzem la funció *sample* del paquet *dplyr* per seleccionar els números de fila dintre del *data frame* *targets_big* corresponents a deu mostres a l'atzar del grup “NIT”, a deu del grup “SFI” i a deu del grup “ELI”. Aquests números de fila els posem en un vector.

```
library(dplyr)  
  
##  
## Attaching package: 'dplyr'  
  
## The following objects are masked from 'package:stats':  
##  
##     filter, lag  
  
## The following objects are masked from 'package:base':  
##  
##     intersect, setdiff, setequal, union  
  
samplNIT <- sample(which(targets_big$Group == "NIT"), 10)  
samplSFI <- sample(which(targets_big$Group == "SFI"), 10)  
samplELI <- sample(which(targets_big$Group == "ELI"), 10)  
  
samples <- c(samplNIT, samplSFI, samplELI)
```

Tot seguit creem un *data frame* *targets* reduït que conté només les mostres seleccionades.

```
targets <- targets_big[samples,]
```

Utilitzant els noms de les mostres segons aquest *data frame* *targets* reduït, creem un *data frame* *counts* que contingui només les mostres seleccionades.

```
columns <- targets$Sample_Name  
counts <- counts_big[columns]
```

Finalment, de cara a poder fer alguns dels procediments de més endavant, corregim el nom de les files i columnes de *counts*.

Del nom de les files, que correspon a l'identificador Ensembl dels gens, n'eliminem el número de versió de la part final, deixant només l'identificador en sí.

El nom de les columnes, que és l'identificador de les mostres, el substituïm pel “ShortName”, que resulta més manejable.

```
X <- c()

for (i in 1:length(counts_big$X)) {
  X[i] <- substring(counts_big$X[i], 1, 15)
}

row.names(counts) <- X
colnames(counts) <- targets$ShortName
```

A partir dels dos *data frames* construïm l'objecte *DESeqDataSet* que utilitzarem per fer l'anàlisi (amb la funció *DESeqDataSetFromMatrix* del paquet *DESeq2*). Al crear-lo, especificuem els factors que volem incloure en el disseny, en aquest cas els paràmetres *sex* i *Group*. Després, amb la funció *factor* especificuem que volem incloure al disseny els tres nivells del factor *Group*, i que “NIT” és el nivell de referència.

```
library(DESeq2)

## Loading required package: S4Vectors
## Loading required package: stats4
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
## 
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:dplyr':
## 
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
## 
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
## 
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which, which.max, which.min

##
## Attaching package: 'S4Vectors'
```

```

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## The following object is masked from 'package:grDevices':
##
##     windows

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment

## Loading required package: Biobase

## Welcome to Bioconductor
##
## Vignettes contain introductory material; view with
##   'browseVignettes()'. To cite Bioconductor, see
##   'citation("Biobase")', and for packages 'citation("pkgname")'.

## Loading required package: DelayedArray

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians

## The following object is masked from 'package:dplyr':
##
##     count

## Loading required package: BiocParallel

##
## Attaching package: 'DelayedArray'

## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges

## The following objects are masked from 'package:base':
##
##     aperm, apply, rowsum

```

```

dds <- DESeqDataSetFromMatrix(countData = counts,
                               colData = targets,
                               design = ~ sex + Group)
dds$Group <- factor(dds$Group, levels = c("NIT", "SFI", "ELI"))
dds

## class: DESeqDataSet
## dim: 56202 30
## metadata(1): version
## assays(1): counts
## rownames(56202): ENSG00000223972 ENSG00000227232 ... ENSG00000210195
##   ENSG00000210196
## rowData names(0):
## colnames(30): S7SE-_NIT ZC5H-_NIT ... 13QJC_ELI YJ89-_ELI
## colData names(9): Experiment SRA_Sample ... Group ShortName

```

3.3.2. Control de qualitat i filtratge

Per fer els primers gràfics del control de qualitat partirem de les dades del *data frame counts*.

Primer transformem les dades en *pseudocounts* utilitzant la fórmula $y = \log_2(K + 1)$. Això ens permetrà visualitzar millor les possibles diferències entre mostres, ja que les dades crues queden massa compactades al ser representades en gràfics.

```
pseudoCounts <- log2(counts + 1)
```

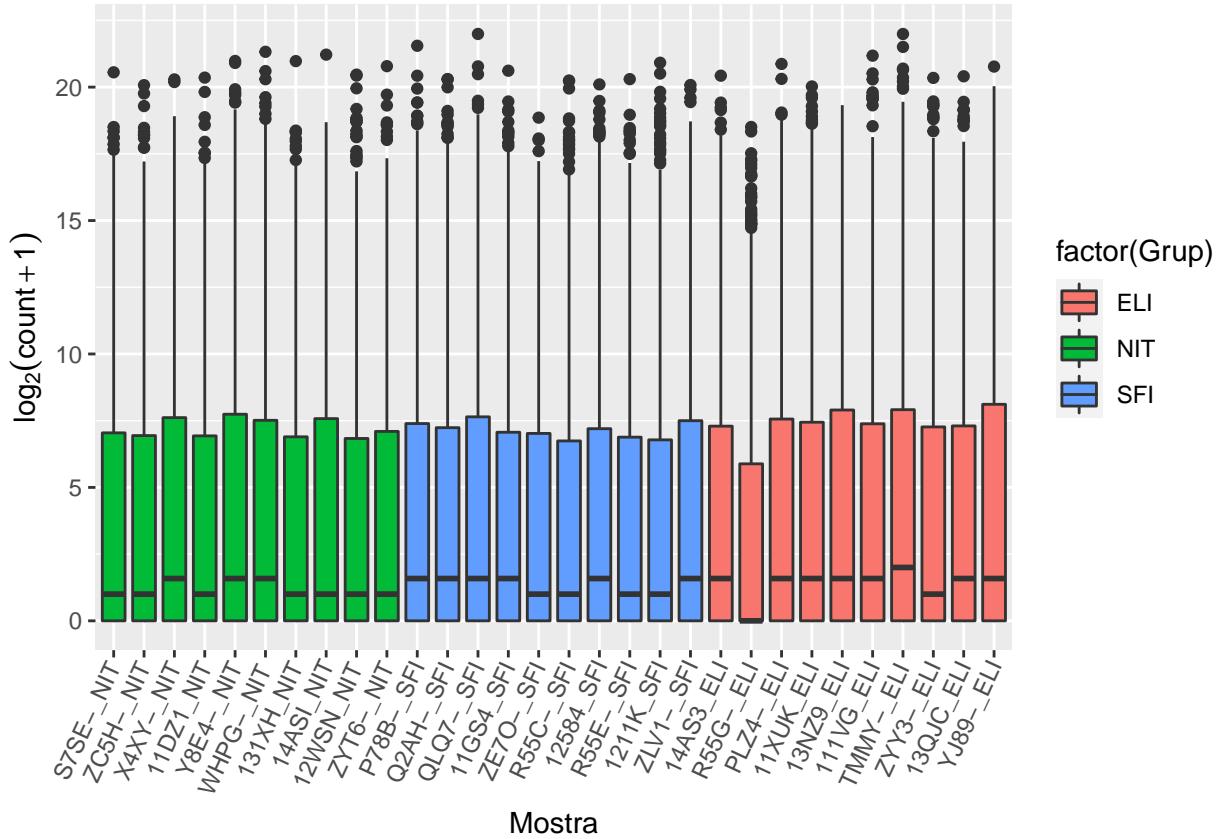
Generem un boxplot per comparar els *pseudocounts* entre mostres.

```

library(reshape2)
library(ggplot2)
df <- melt(pseudoCounts, variable.name = "Samples")

## No id variables; using all as measure variables
df$Grup <- c(rep("NIT", 562020), rep("SFI", 562020), rep("ELI", 562020))
ggplot(df, aes(x = Samples, y = value)) + geom_boxplot(aes(fill = factor(Grup))) + xlab("Mostra") +
ylab(expression(log[2](count + 1))) + theme(axis.text.x = element_text(angle = 65, hjust = 1.2, vjust =

```

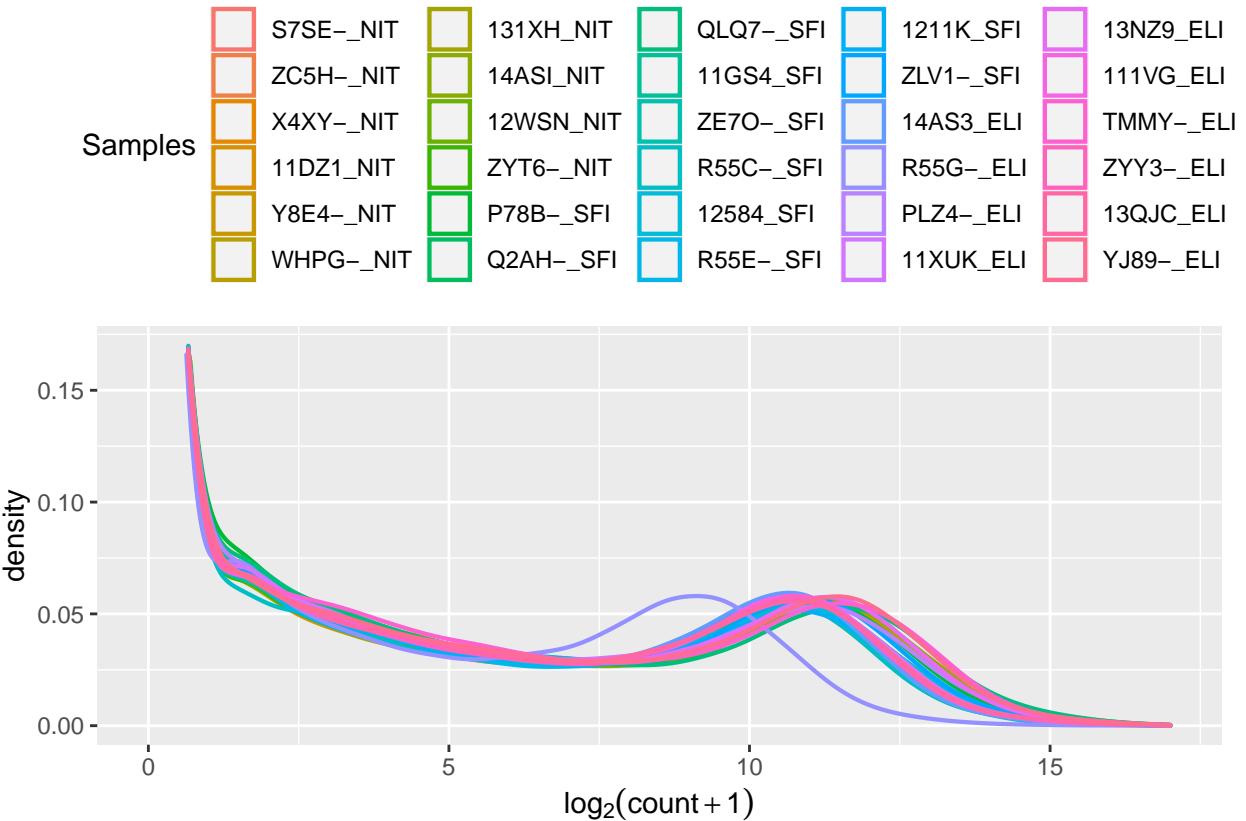


No observem grans diferències entre les mostres. L'única que potser destaca és la mostra R55G_ELI, que té un número de *counts* inferior a la resta.

Comparem les mostres també amb un histograma.

```
ggplot(df, aes(x = value, colour = Samples)) + ylim(c(0, 0.17)) + xlim(c(0, 17)) +
  geom_density(alpha = 0.2, size = 0.75) +
  theme(legend.position = "top") + xlab(expression(log[2](count + 1)))
```

Warning: Removed 556 rows containing non-finite values (stat_density).



De nou, trobem que en general no hi ha grans variacions entre les mostres, excepte en el cas de R55G_EL1, que en aquest gràfic sí que es veu més clarament diferenciada.

De totes maneres, amb el que podem veure en els gràfics (la distribució dels quartils i els *outliers* al *boxplot*, i la forma de la corva a l'histogramma), és probable que aquesta diferència afecti de forma semblant a tots els gens, i en aquest cas quedarà en part corregit amb la normalització.

Per altra banda, es tracta d'una mostra del grup de teixit extensivament infiltrat, i per tant esperem una major variabilitat en els nivells d'expressió gènica. En aquest cas els nivells d'expressió semblen força diferents dels de la resta de mostres, però no podem descartar que la variació sigui deguda a la condició d'estudi -cosa que aportaria informació rellevant-, i per tant optarem per mantenir-la en l'anàlisi.

Per generar la resta de gràfics del control de qualitat ens convé estabilitzar la variança. Ho farem amb la funció *rlog* (Michael I Love, Wolfgang Huber, Simon Anders 2014), i no amb l'alternativa *vst*, ja que el temps de computació no és un problema.

Abans, però, filtrem el *DESeqDataSet* per eliminar les files amb un número total de *counts* de 0 o 1, que no ens aportarien cap informació però poden generar soroll innecessari.

El número de gens abans i després del filtratge és aquest.

```
nrow(dds)
## [1] 56202
dds <- dds[ rowSums(counts(dds)) > 1, ]
nrow(dds)
## [1] 43364
```

Comprovem que el filtratge redueix el número de gens a 43364.

```
rld <- rlog(dds)

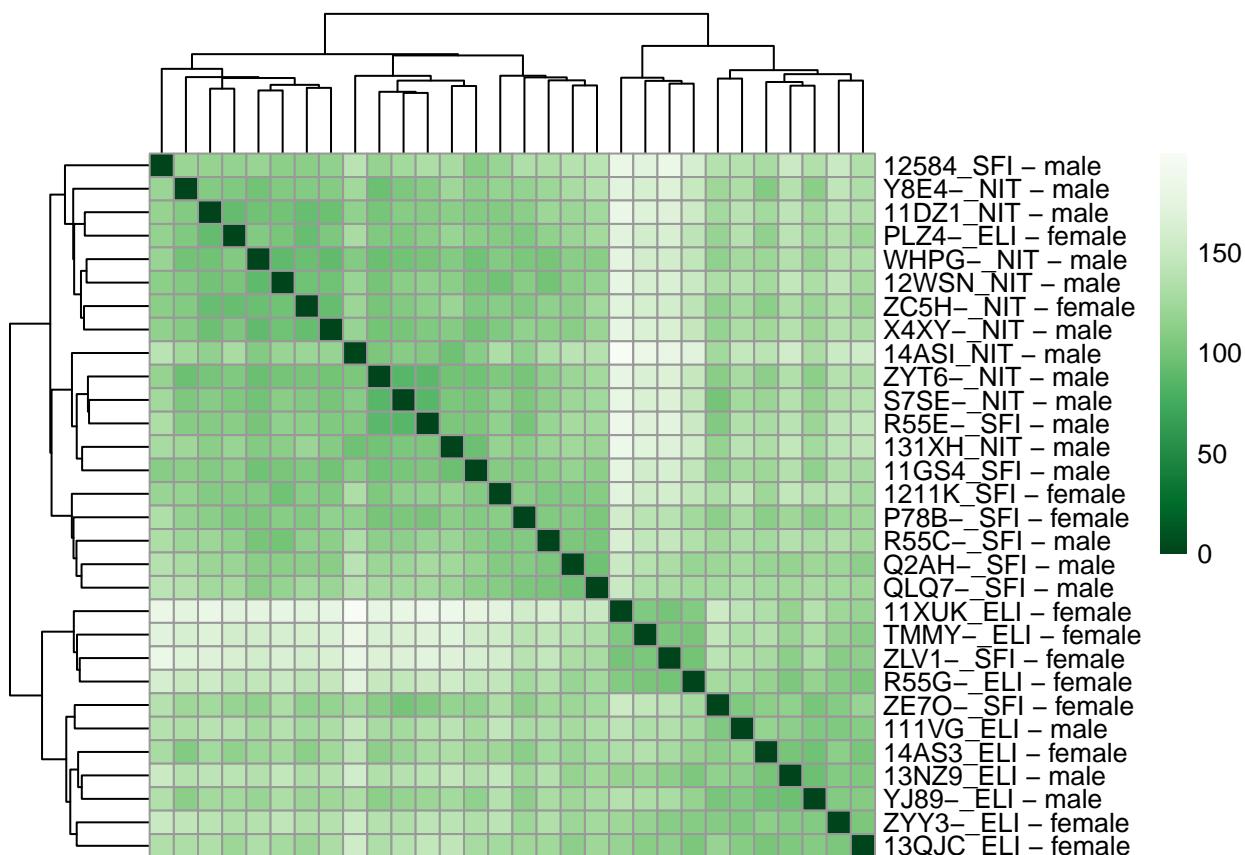
## rlog() may take a few minutes with 30 or more samples,
## vst() is a much faster transformation
```

Un cop estabilitzada la variança, podem fer una ullada a la distància entre mostres. Ho fem amb un *heatmap* que compara cada mostra amb les altres.

```
sampleDists <- dist(t(assay(rld)))

library("pheatmap")
library("RColorBrewer")

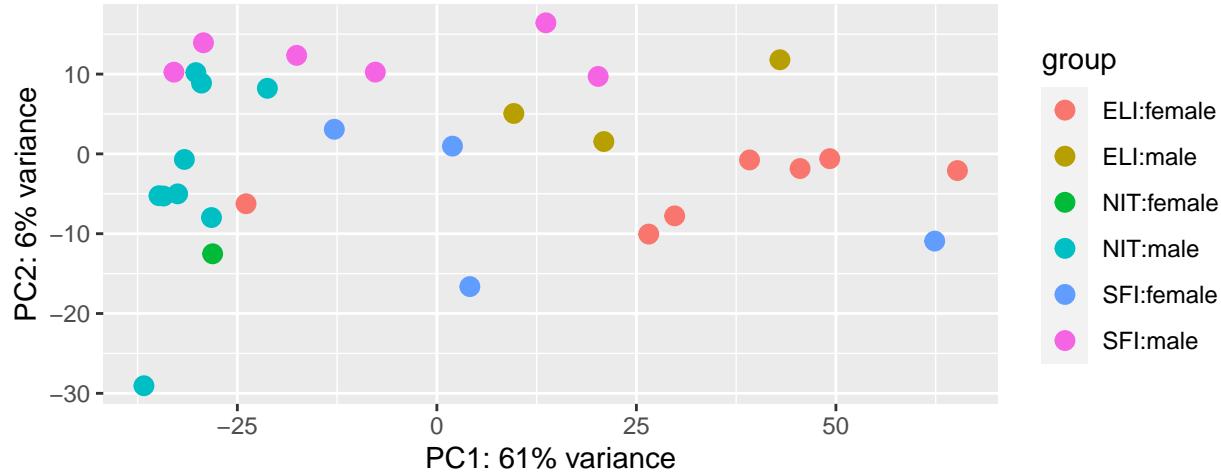
sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste( rld$ShortName, rld$sex, sep = " - " )
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Greens")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows = sampleDists,
         clustering_distance_cols = sampleDists,
         col = colors)
```



Observem que les mostres es separen en dos grups bastant diferenciats, un d'ells incloent la majoria de mostres del grup ELI i algunes del grup SFI, i l'altre la resta. A part d'aquesta distinció, no hi ha cap mostra que destaquï especialment respecte a les altres.

També mirem com es distribueixen les mostres segons l'anàlisi de components principals.

```
plotPCA(rld, intgroup = c("Group", "sex"))
```

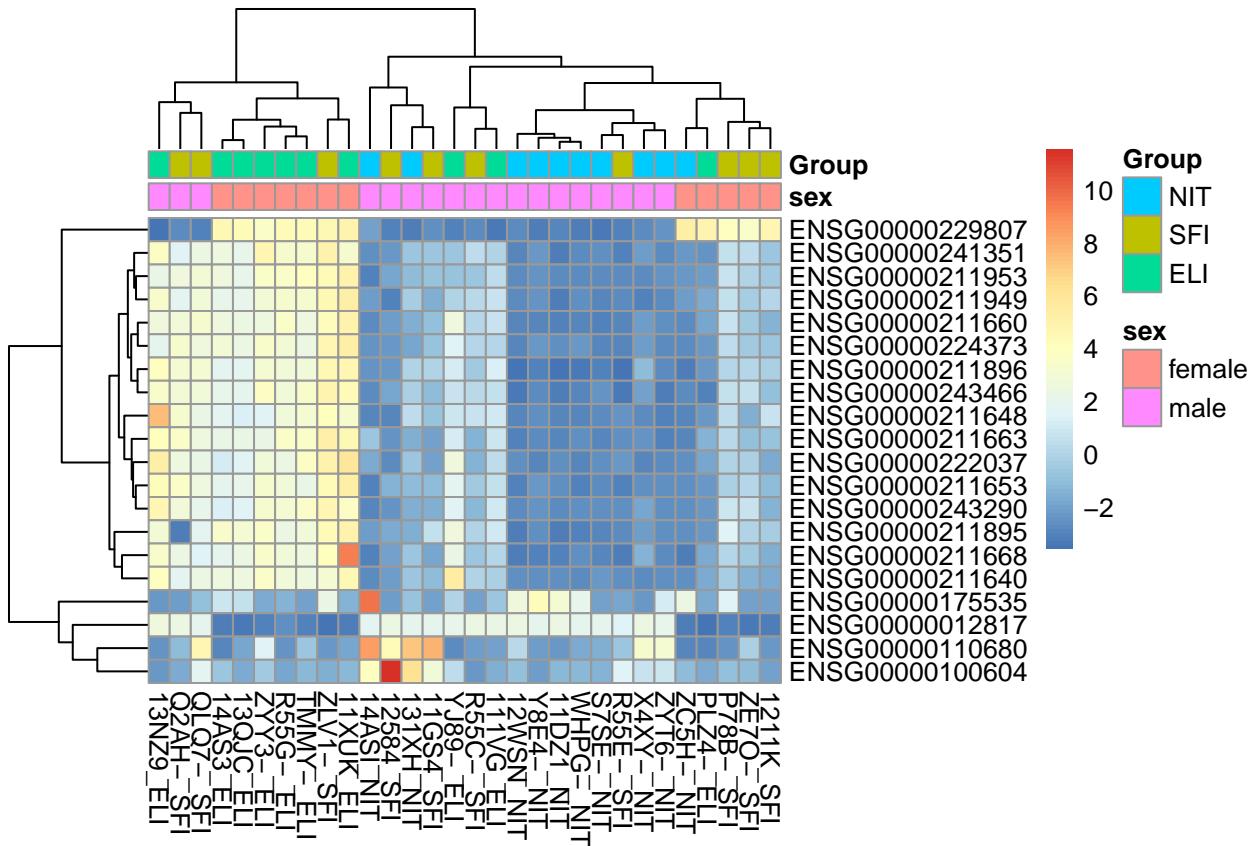


Veiem que la primera component principal explica la major part de la variabilitat (un 61%). En aquest eix, gairebé totes les mostres del grup ELI queden clarament diferenciades de les del grup NIT. Les mostres del grup SFI queden més repartides.

Un altre gràfic que podem fer per veure com s'agrupen les mostres és un heatmap de la variança dels 20 gens amb expressió més variable. Per fer-lo, de nou treballem a partir de les dades amb variança reduïda.

```
library("genefilter")
## 
## Attaching package: 'genefilter'
## The following objects are masked from 'package:matrixStats':
##   rowSds, rowVars
topVarGenes <- head(order(rowVars(assay(rld))), decreasing = TRUE), 20)

mat  <- assay(rld)[topVarGenes, ]
mat  <- mat - rowMeans(mat)
anno <- as.data.frame(colData(rld)[, c("sex", "Group")])
pheatmap(mat, annotation_col = anno)
```



Tornem a observar que les mostres s'agrupen principalment en dos grups, un que inclou la majoria de mostres del grup ELI i algunes del grup SFI, i un altre que inclou la resta de mostres. També crida l'atenció que dintre d'aquests dos grups les mostres es separen clarament per sexes.

3.3.3. Normalització

Per tenir en compte les diferències sistemàtiques d'origen tècnic entre mostres, apliquem una normalització. Aquesta consisteix en multiplicar el número de *counts* de cada mostra del nostre *DESeqDataSet* per una constant $C_{\{j\}}$ que calculem a partir de les dades de *counts* de la mateixa mostra. El tipus de normalització que apliquem variarà en funció del tipus de càlcul que fem per obtenir $C_{\{j\}}$.

En aquest cas utilitzarem el mètode RLE (*Relative Log Expression*, (Anders, S, Huber, W 2010)), que, partint de la mitjana geomètrica de l'expressió entre mostres, crea una “mostra de pseudo-referència” sobre la qual centrar les dades de les diferents mostres. Després de centrar les dades, busca la mitjana entre gens per a cada mostra.

Aquest mètode es pot implementar amb la funció *estimateSizeFactors* del paquet *DESeq2*.

```
dds <- estimateSizeFactors(dds)
```

3.3.4. Anàlisi d'expressió diferencial

L'anàlisi d'expressió diferencial consisteix en fer, per a cada contrast d'interès, tests de significació en un model binomial negatiu. Els paràmetres del model són desconeguts, però podem obtenir-ne una estimació.

La funció *estimateDispersions* del paquet *DESeq2* fa una estimació dels paràmetres de dispersió de les dades, a partir de la qual podrem obtenir els paràmetres del model.

- Primer fa un càlcul preliminar de la dispersió de cada gen mitjançant una estimació per màxima veosimilitud (McCarthy, DJ, Chen, Y, Smyth, GK 2012).
- Després encongieix els valors acostant-los a la corva ajustada definida per aquesta mateixa estimació preliminar.

```
dds <- estimateDispersions(dds)

## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
```

Un cop estimats els paràmetres de dispersió podem passar a fer l'anàlisi d'expressió diferencial en sí. Convé recordar que al crear l'objecte *DESeqDataSet* ja hem especificat els factors que volem incloure al nostre disseny.

La funció *nbinomWaldTest* del paquet *DESeq2* fa el càlcul del test per a cada gen dintre de cada comparació, iafegeix els resultats al *DESeqDataSet* com a metadades. Aquests resultats inclouen el Log₂ del fold-change, el p-valor i el p-valor ajustat del contrast.

```
dds <- nbinomWaldTest(dds)
```

Utilitzem la funció *results* per explorar els resultats dels diferents contrasts.

Mirem primer un resum dels resultats del contrast entre els grups “ELI” i “NIT”.

```
resELIvsNIT <- results(dds, contrast=c("Group", "NIT", "ELI"))

summary(resELIvsNIT)
```

```
##
## out of 43364 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)      : 881, 2%
## LFC < 0 (down)    : 2154, 5%
## outliers [1]       : 191, 0.44%
## low counts [2]     : 15096, 35%
## (mean count < 2)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

També mirem alguns dels valors associats als gens que tenen un diferencial d'expressió positiu i negatiu més gran.

```
resSigELIvsNIT <- subset(resELIvsNIT, padj < 0.1)
valSigELIvsNIT <- resSigELIvsNIT[c("log2FoldChange", "pvalue", "padj")]
head(valSigELIvsNIT[ order(valSigELIvsNIT$log2FoldChange, decreasing = TRUE), ], 3)

## log2 fold change (MLE): Group NIT vs ELI
## Wald test p-value: Group NIT vs ELI
## DataFrame with 3 rows and 3 columns
##          log2FoldChange           pvalue           padj
##          <numeric>           <numeric>           <numeric>
## ENSG00000108849  4.9071129578193  0.00128138864454107  0.0244578850936638
## ENSG00000170419  4.74493205228696  5.57176867461188e-06  0.000402155653154441
## ENSG00000079689  4.67587739452871  2.44597214624369e-05  0.0012836553261698

head(valSigELIvsNIT[ order(valSigELIvsNIT$log2FoldChange), ], 3)
```

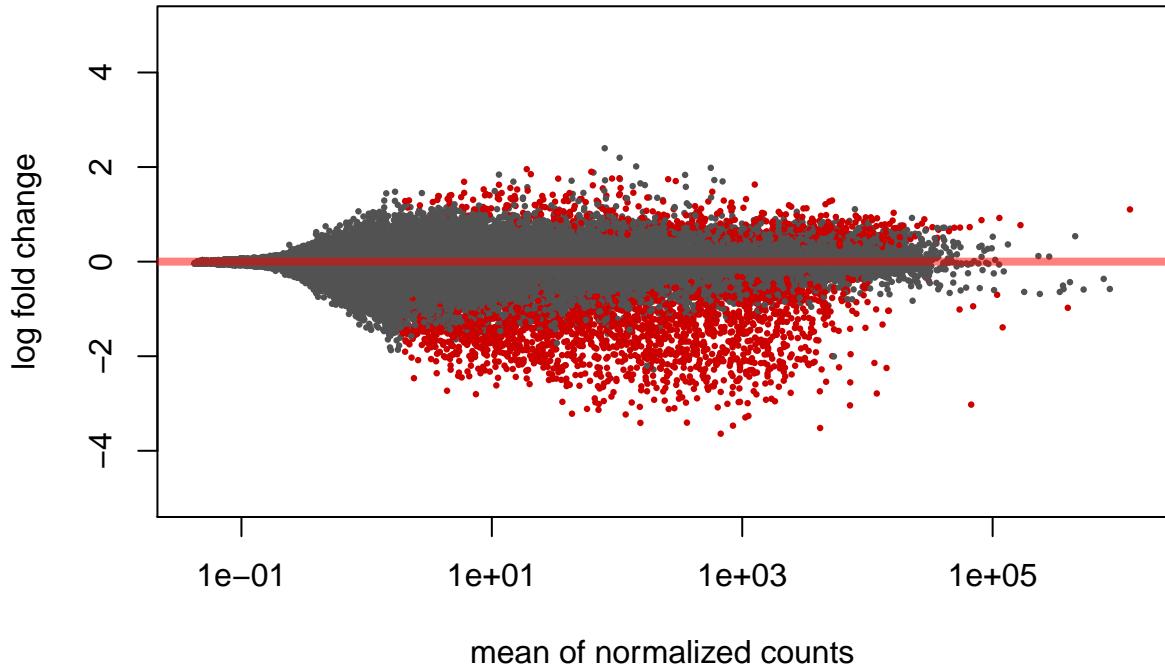
```

## log2 fold change (MLE): Group NIT vs ELI
## Wald test p-value: Group NIT vs ELI
## DataFrame with 3 rows and 3 columns
##           log2FoldChange          pvalue         padj
##           <numeric>          <numeric>      <numeric>
## ENSG00000211654 -22.36784577138 1.90368126580899e-18 1.06899317800238e-14
## ENSG00000170054 -21.8165051466117 5.01323261897087e-34 1.40756532242845e-29
## ENSG00000253742 -21.6755614741358 1.09220599710142e-33 1.53329338903082e-29

Fem un MA plot del contrast. Per fer-lo, primer hem d'encongrir el log2 dels fold changes amb la funció lfcShrink.
shrELIvsNIT <- lfcShrink(dds, contrast=c("Group", "NIT", "ELI"))

## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895
plotMA(shrELIvsNIT, ylim = c(-5, 5))

```



Observem que en el teixit infiltrat, respecte el no infiltrat, hi ha un major nombre de gens inhibits que de gens sobreexpressats, i aquests gens inhibits mostren un major fold change i un nivell més alt de significació (un p-valor més petit). Això potser s'explica perquè les diferències d'expressió en el teixit infiltrat tendeixen per naturalesa a ser en forma d'inhibició de molts gens reguladors.

Explorem el contrast entre els grups “SFI” i “NIT”.

Resum dels resultats:

```
resSFIvsNIT <- results(dds, contrast=c("Group", "NIT", "SFI"))

summary(resSFIvsNIT)

##
## out of 43364 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)      : 55, 0.13%
## LFC < 0 (down)    : 277, 0.64%
## outliers [1]       : 191, 0.44%
## low counts [2]     : 13432, 31%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Gens amb un diferencial d'expressió positiu i negatiu més gran:

```
resSigSFIvsNIT <- subset(resSFIvsNIT, padj < 0.1)
valSigSFIvsNIT <- resSigSFIvsNIT[c("log2FoldChange", "pvalue", "padj")]
head(valSigSFIvsNIT[ order(valSigSFIvsNIT$log2FoldChange, decreasing = TRUE), ], 3)
```

```
## log2 fold change (MLE): Group NIT vs SFI
## Wald test p-value: Group NIT vs SFI
## DataFrame with 3 rows and 3 columns
##           log2FoldChange          pvalue          padj
##           <numeric>          <numeric>          <numeric>
## ENSG00000216522 3.48987029993807 0.000422348293258026 0.050854496314927
## ENSG00000248560 3.48229581665463 0.000258269176448894 0.0350738976108062
## ENSG00000213642 3.26712295520834 9.38743594434013e-05 0.0158821115886208
head(valSigSFIvsNIT[ order(valSigSFIvsNIT$log2FoldChange), ], 3)
```

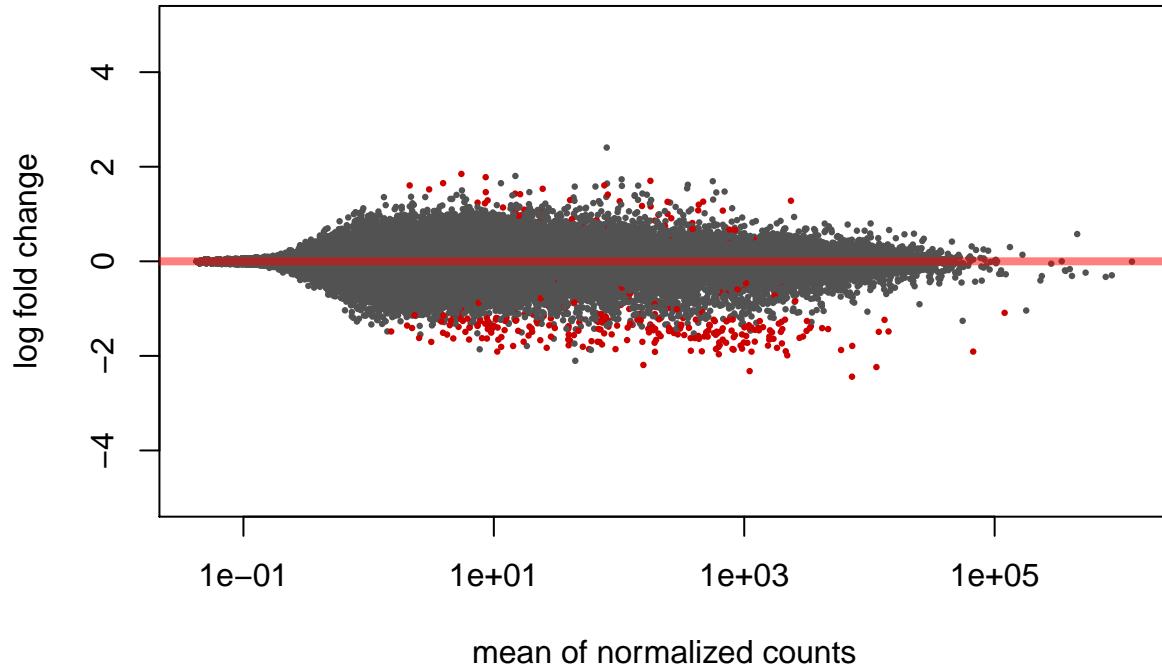
```
## log2 fold change (MLE): Group NIT vs SFI
## Wald test p-value: Group NIT vs SFI
## DataFrame with 3 rows and 3 columns
##           log2FoldChange          pvalue          padj
##           <numeric>          <numeric>          <numeric>
## ENSG00000211654 -21.9099968985243 2.76331922069389e-21 1.64367753885314e-17
## ENSG00000253131 -20.4081759057272 2.0070185733501e-28 2.98453696950027e-24
## ENSG00000253742 -20.3181310036864 2.27831159641197e-35 6.77592651888885e-31
```

MA plot:

```
shrSFIvsNIT <- lfcShrink(dds, contrast=c("Group", "NIT", "SFI"))

## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895
```

```
plotMA(shrSFIvsNIT, ylim = c(-5, 5))
```



En aquest cas també trobem diferències en el mateix sentit: el grup de teixit parcialment infiltrat té més gens inhibits que gens sobreexpressats respecte el grup de teixit no infiltrat. La diferència és que el número de gens diferencialment expressats és menor que en l'anterior comparació.

Explorem els resultats del contrast entre els grups “ELI” i “SFI”.

Resum dels resultats:

```
resELIvsSFI <- results(dds, contrast=c("Group", "SFI", "ELI"))

summary(resELIvsSFI)
```

```
##
## out of 43364 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)      : 1320, 3%
## LFC < 0 (down)    : 1512, 3.5%
## outliers [1]       : 191, 0.44%
## low counts [2]     : 17598, 41%
## (mean count < 4)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Gens amb un diferencial d'expressió positiu i negatiu més gran:

```

resSigELIvsSFI <- subset(resELIvsSFI, padj < 0.1)
valSigELIvsSFI <- resSigELIvsSFI[c("log2FoldChange", "pvalue", "padj")]
head(valSigELIvsSFI[ order(valSigELIvsSFI$log2FoldChange, decreasing = TRUE), ], 3)

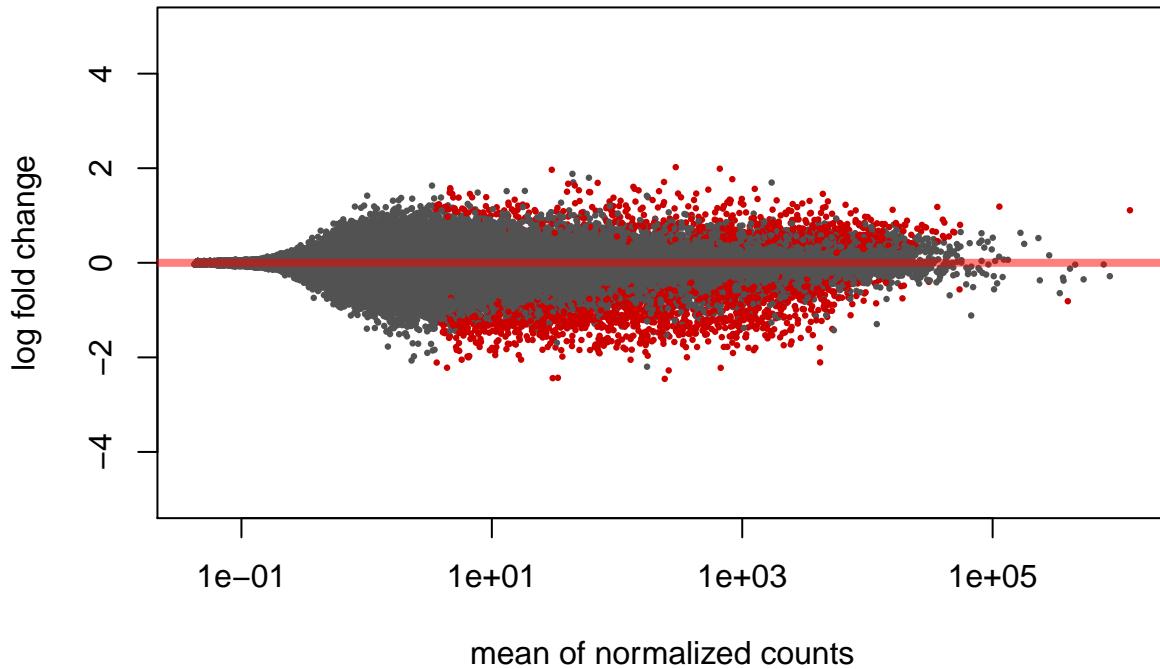
## log2 fold change (MLE): Group SFI vs ELI
## Wald test p-value: Group SFI vs ELI
## DataFrame with 3 rows and 3 columns
##           log2FoldChange          pvalue          padj
##           <numeric>          <numeric>          <numeric>
## ENSG00000181092 4.09926430328123 3.28763222370361e-05 0.00587980378470069
## ENSG00000089225 3.98119016224604 6.54329566980265e-05 0.00804542244015398
## ENSG00000164326 3.72378454936897 4.99181610057772e-05 0.00734885079009114
head(valSigELIvsSFI[ order(valSigELIvsSFI$log2FoldChange), ], 3)

## log2 fold change (MLE): Group SFI vs ELI
## Wald test p-value: Group SFI vs ELI
## DataFrame with 3 rows and 3 columns
##           log2FoldChange          pvalue          padj
##           <numeric>          <numeric>          <numeric>
## ENSG00000171195 -9.36895738684711 1.95805691553369e-05 0.00447118800131912
## ENSG00000254029 -6.23480756553029 0.000938714138255819 0.0277865903771905
## ENSG00000259721 -4.40925845029186 1.50308895511771e-05 0.00392260204358524

MA plot:
shrELIvsSFI <- lfcShrink(dds, contrast=c("Group", "SFI", "ELI"))

## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895
plotMA(shrELIvsSFI, ylim = c(-5, 5))

```



En aquest contrast estem comparant un grup de mostres de teixit extensament infiltrat amb un grup de mostres de teixit parcialment infiltrat. Observem que hi ha força gens diferencialment expressats, però en aquest cas no n'hi ha gaires més d'inhibits que de sobreexpressats. Tant el fold change com el nivell de significació dels gens més diferencialment expressats són menors que en les anteriors comparacions.

3.3.5. Anotació dels resultats

Per poder seguir amb l'anàlisi de les dades, necessitem associar als gens que hem detectat com a diferencialment expressats el *gene symbol* i l'*Entrez ID* corresponents. Ho fem a partir del paquet d'anotacions *org.Hs.eg.db*.

```
library("org.Hs.eg.db")

## Loading required package: AnnotationDbi
##
## Attaching package: 'AnnotationDbi'
## The following object is masked from 'package:dplyr':
##   select
##
library("AnnotationDbi")

columns(org.Hs.eg.db)

## [1] "ACCCNUM"      "ALIAS"        "ENSEMBL"       "ENSEMLPROT"    "ENSEMLTRANS"
```

```

## [6] "ENTREZID"      "ENZYME"        "EVIDENCE"       "EVIDENCEALL"    "GENENAME"
## [11] "GO"             "GOALL"          "IPI"            "MAP"           "OMIM"
## [16] "ONTOLOGY"       "ONTOLOGYALL"    "PATH"           "PFAM"          "PMID"
## [21] "PROSITE"        "REFSEQ"         "SYMBOL"         "UCSCKG"        "UNIGENE"
## [26] "UNIPROT"

keytypes(org.Hs.eg.db)

## [1] "ACCNUM"        "ALIAS"          "ENSEMBL"        "ENSEMLPROT"    "ENSEMLTRANS"
## [6] "ENTREZID"       "ENZYME"         "EVIDENCE"       "EVIDENCEALL"   "GENENAME"
## [11] "GO"             "GOALL"          "IPI"            "MAP"           "OMIM"
## [16] "ONTOLOGY"       "ONTOLOGYALL"    "PATH"           "PFAM"          "PMID"
## [21] "PROSITE"        "REFSEQ"         "SYMBOL"         "UCSCKG"        "UNIGENE"
## [26] "UNIPROT"

resSigELIvsNIT$symbol <- mapIds(org.Hs.eg.db, keys=row.names(resSigELIvsNIT),
                                    column="SYMBOL", keytype="ENSEMBL", multiVals="first")

## 'select()' returned 1:many mapping between keys and columns
resSigELIvsNIT$entrez <- mapIds(org.Hs.eg.db, keys=row.names(resSigELIvsNIT),
                                   column="ENTREZID", keytype="ENSEMBL",
                                   multiVals="first")

## 'select()' returned 1:many mapping between keys and columns
resSigELIvsNIT$genename <- mapIds(org.Hs.eg.db, keys=row.names(resSigELIvsNIT),
                                    column="GENENAME", keytype="ENSEMBL",
                                    multiVals="first")

## 'select()' returned 1:many mapping between keys and columns
resSigSFIvsNIT$symbol <- mapIds(org.Hs.eg.db, keys=row.names(resSigSFIvsNIT),
                                    column="SYMBOL", keytype="ENSEMBL", multiVals="first")

## 'select()' returned 1:many mapping between keys and columns
resSigSFIvsNIT$entrez <- mapIds(org.Hs.eg.db, keys=row.names(resSigSFIvsNIT),
                                   column="ENTREZID", keytype="ENSEMBL",
                                   multiVals="first")

## 'select()' returned 1:many mapping between keys and columns
resSigSFIvsNIT$genename <- mapIds(org.Hs.eg.db, keys=row.names(resSigSFIvsNIT),
                                    column="GENENAME", keytype="ENSEMBL",
                                    multiVals="first")

## 'select()' returned 1:many mapping between keys and columns
resSigELIvsSFI$symbol <- mapIds(org.Hs.eg.db, keys=row.names(resSigELIvsSFI),
                                    column="SYMBOL", keytype="ENSEMBL", multiVals="first")

## 'select()' returned 1:many mapping between keys and columns
resSigELIvsSFI$entrez <- mapIds(org.Hs.eg.db, keys=row.names(resSigELIvsSFI),
                                   column="ENTREZID", keytype="ENSEMBL",
                                   multiVals="first")

## 'select()' returned 1:many mapping between keys and columns

```

```

resSigELIvsSFI$genename <- mapIds(org.Hs.eg.db, keys=row.names(resSigELIvsSFI),
                                    column="GENENAME", keytype="ENSEMBL",
                                    multiVals="first")

## 'select()' returned 1:many mapping between keys and columns

Per conservar els llistats de gens obtinguts, els guardem en arxius csv, ordenant abans els resultats segons el
p-valor ajustat.

resOrdELIvsNIT <- resSigELIvsNIT[order(resSigELIvsNIT$padj),]
resOrdSFIvsNIT <- resSigSFIvsNIT[order(resSigSFIvsNIT$padj),]
resOrdELIvsSFI <- resSigELIvsSFI[order(resSigELIvsSFI$padj),]

dfELIvsNIT <- as.data.frame(resOrdELIvsNIT)
write.csv(dfELIvsNIT, file = "results/resultsELIvsNIT.csv")

dfSFIvsNIT <- as.data.frame(resOrdSFIvsNIT)
write.csv(dfSFIvsNIT, file = "results/resultsSFIvsNIT.csv")

dfELIvsSFI <- as.data.frame(resOrdELIvsSFI)
write.csv(dfELIvsSFI, file = "results/resultsELIvsSFI.csv")

```

3.3.6. Comparació entre comparacions

Podem comparar els diferents contrastos per veure quants gens diferencialment expressats tenen en comú.

En aquest cas, farem un diagrama de Venn amb els gens que mostren un $\log_2\{2\}$ fold-change superior a 1.

Per fer-lo, abans hem de crear una matriu que, per a cada gen que apareix diferencialment expressat en algun dels contrastos, ens digui si ho està o no per a cada un dels contrastos (en mostrem aquí només els primers valors).

```

library(limma)

##
## Attaching package: 'limma'

## The following object is masked from 'package:DESeq2':
##
##     plotMA

## The following object is masked from 'package:BiocGenerics':
##
##     plotMA

selectedELIvsNIT <- subset(dfELIvsNIT, abs(log2FoldChange) > 1)
selectedELIvsNIT <- row.names(selectedELIvsNIT)

selectedSFIvsNIT <- subset(dfSFIvsNIT, abs(log2FoldChange) > 1)
selectedSFIvsNIT <- row.names(selectedSFIvsNIT)

selectedELIvsSFI <- subset(dfELIvsSFI, abs(log2FoldChange) > 1)
selectedELIvsSFI <- row.names(selectedELIvsSFI)

selectednames <- unique(c(selectedELIvsNIT, selectedSFIvsNIT, selectedELIvsSFI))

selected <- matrix(rep(0, length(selectednames)*3), ncol = 3)

```

```

colnames(selected) <- c("ELI vs NIT", "SFI vs NIT", "ELI vs SFI")
row.names(selected) <- selectednames

for (i in 1:length(selectednames)) {
    if (selectednames[i] %in% selectedELIvsNIT) {
        selected[i,1] <- 1
    }
}

for (i in 1:length(selectednames)) {
    if (selectednames[i] %in% selectedSFIvsNIT) {
        selected[i,2] <- 1
    }
}

for (i in 1:length(selectednames)) {
    if (selectednames[i] %in% selectedELIvsSFI) {
        selected[i,3] <- 1
    }
}

head(selected, 10)

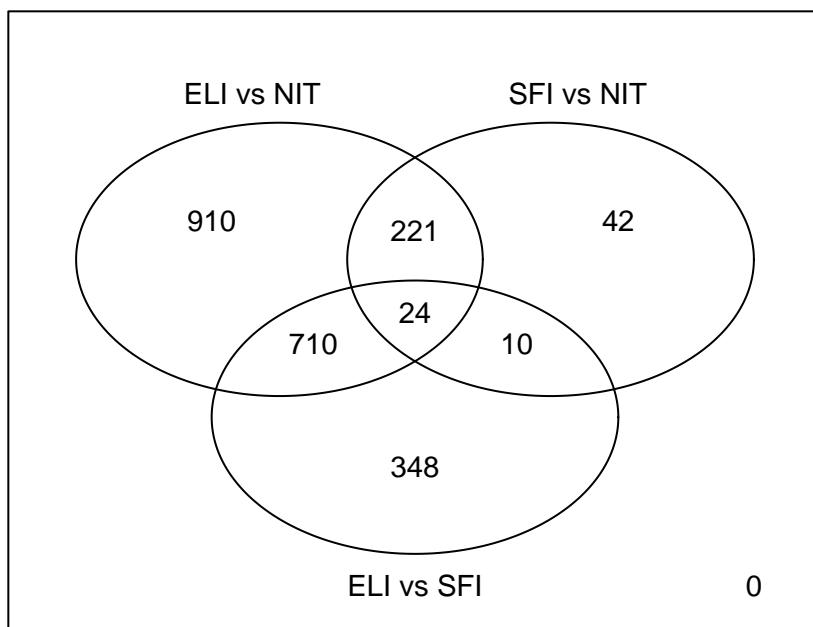
##          ELI vs NIT SFI vs NIT ELI vs SFI
## ENSG00000170054      1      1      0
## ENSG00000253742      1      1      0
## ENSG00000237638      1      1      0
## ENSG00000253131      1      1      0
## ENSG00000211654      1      1      0
## ENSG00000211685      1      1      0
## ENSG00000132465      1      1      0
## ENSG00000170476      1      1      0
## ENSG00000249096      1      1      1
## ENSG00000167483      1      1      1

selectedCounts <- vennCounts(selected)

vennDiagram(selectedCounts, cex=0.9)
title("Gens diferencialment expressats segons cada comparació.")

```

Gens diferencialment expressats segons cada comparació.



Suposant que els diferents graus d'infiltració en el teixit fossin una gradació contínua que es correspongués exactament amb els gens diferencialment expressats respecte el teixit sa, els gens alterats en teixits amb un grau baix d'infiltració formarien tots part del total de gens alterats en teixits més infiltrats. Això obviament no és així, però sí que hi ha una tendència a que gens alterats en teixits poc infiltrats ho estiguin també en teixits més infiltrats, com podem observar al diagrama.

Veiem doncs que la majoria dels gens diferencialment expressats en la comparació SFI vs NIT, que són relativament pocs, ho estan també en la comparació ELI vs NIT, però que aquesta última en té molts més. Aquesta diferència en la mida dels grups fa pensar que els teixits ELI tenen un grau d'afectació força més gran que els teixits SFI.

Pel que fa al contrast ELI vs SFI, podem interpretar la seva relació amb els altres contrastos en el mateix sentit. Una majoria dels gens que hi apareixen alterats ho fan també en el contrast ELI vs NIT, que en té força més.

3.3.7. Anàlisi de significació biològica

Farem l'anàlisi de significació biològica a partir de les anotacions de *pathways* de la Kyoto Encyclopedia of Genes and Genomes (KEGG, <https://www.genome.jp/kegg/>).

Abans de fer-la, però, hem de crear el llistat d'identificadors *Entrez* dels gens diferencialment expressats.

Al fer l'anàlisi d'expressió diferencial ja hem seleccionat els gens de cada comparació amb un p-valor ajustat que hem considerat significatiu (menor de 0,1). Per crear el llistat d'identificadors només els hem d'extreure dels *data frames* corresponents.

```
selectedIDs <- list(dfELIvsNIT$entrez, dfSFIvsNIT$entrez, dfELIvsSFI$entrez)
names(selectedIDs) <- c("ELIvsNIT", "SFIvsNIT", "ELIvsSFI")
```

L'altra cosa que hem de fer abans de començar l'anàlisi de significació biològica és generar un llistat amb tots els gens humans coneguts que tenen almenys una anotació a KEGG. Ho fem amb la funció *mappedkeys* del paquet *AnnotationDbi*, a partir de les anotacions del genoma humà contingudes al paquet *org.Hs.eg.db*.

```
library(AnnotationDbi)
mapped_genes <- mappedkeys(org.Hs.egPATH)
```

L'anàlisi de significació biològica consistirà en detectar, per a cada contrast, *pathways* comuns als diferents gens sobreexpressats. Ho farem utilitzant la funció *enrichKEGG* del paquet *clusterProfiler*, que fa servir tests de sobrerepresentació per generar la llista de *pathways*.

El codi utilitzat també guarda la llista corresponent a cada comparació, que inclou l'identificador del *pathway* i algunes estadístiques associades, en un arxiu csv, i per cada llista genera un gràfic de barres i un de ret.

```
library(clusterProfiler)
```

```
##
```

```
## Registered S3 method overwritten by 'enrichplot':
##   method           from
##   fortify.enrichResult DOSE

## clusterProfiler v3.14.3  For help: https://guangchuangyu.github.io/software/clusterProfiler
##
## If you use clusterProfiler in published research, please cite:
## Guangchuang Yu, Li-Gen Wang, Yanyan Han, Qing-Yu He. clusterProfiler: an R package for comparing bio
##
## Attaching package: 'clusterProfiler'

## The following object is masked from 'package:DelayedArray':
## 
##   simplify

listOfData <- selectedIDs[1:3]
comparisonsNames <- names(listOfData)

for (i in 1:length(listOfData)){
  genesIn <- listOfData[[i]]
  comparison <- comparisonsNames[i]

  enrich.KEGG <- enrichKEGG(gene = genesIn,
                            organism      = 'hsa',
                            pvalueCutoff = 0.1,
                            pAdjustMethod = "BH",
                            universe     = mapped_genes)

  if (length(rownames(enrich.KEGG@result)) != 0) {
    write.csv(as.data.frame(enrich.KEGG),
              file = paste0("./results/", "enrichKEGG.Results.", comparison,
                           ".csv"),
              row.names = FALSE)

    png(file=paste0("./results/","enrichKEGGBarplot.",comparison,".png"),
        width = 800)
    print(barplot(enrich.KEGG, showCategory = 15, font.size = 8,
                  title = paste0("Anàlisi d'enriquiment de pathways KEGG per ", comparison))
```

```

    dev.off()

    png(file = paste0("./results/","enrichKEGGcnetplot.",comparison,".png"))
    print(cnetplot(enrich.KEGG, categorySize = "geneNum",
                  showCategory = 15, vertex.label.cex = 0.75))
    dev.off()
}
}

```

4. Resultats.

Els resultats de l'anàlisi es poden trobar en arxius csv a la carpeta *results* del repositori.

D'una banda, hi ha tres llistats de gens diferencialment expressats, un per cada comparació, que inclouen els identificadors dels gens i les estadístiques de l'expressió diferencial.

També hi ha un llistat de *pathways* de KEGG per cada comparació, de nou amb estadístics associats.

La carpeta de resultats també inclou els gràfics que hem generat a partir dels llistats de *pathways*, que es poden veure a continuació.

```

library(knitr)
include_graphics(c("results/enrichKEGGBarplot.ELIvsNIT.png",
                  "results/enrichKEGGcnetplot.ELIvsNIT.png"))

```

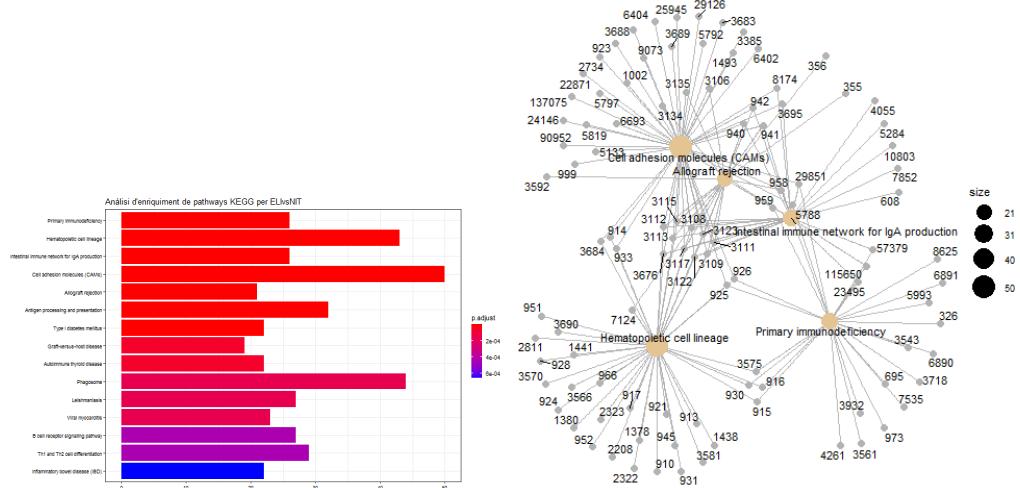


Figure 1: Gràfics de ret de termes GO i de pathways per NC vs CPT

```

include_graphics(c("results/enrichKEGGBarplot.ELIvsSFI.png",
                  "results/enrichKEGGcnetplot.ELIvsSFI.png"))

include_graphics(c("results/enrichKEGGBarplot.SFIvsNIT.png",
                  "results/enrichKEGGcnetplot.SFIvsNIT.png"))

```

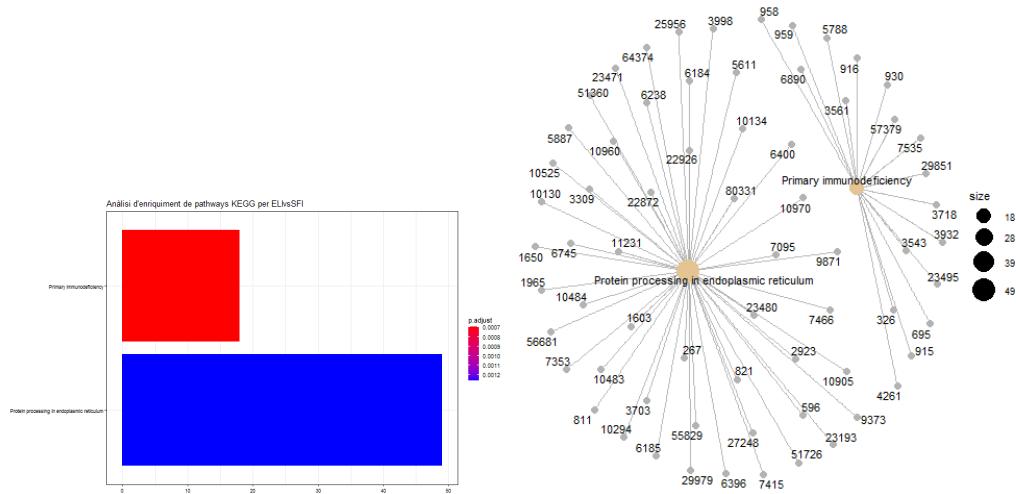


Figure 2: Gràfics de ret de termes GO i de pathways per NC vs CPT

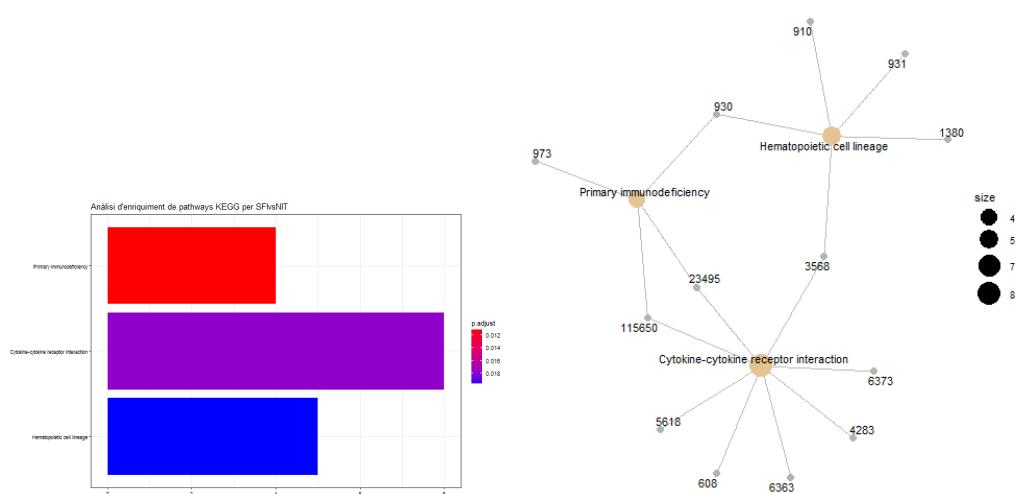


Figure 3: Gràfics de ret de termes GO i de pathways per NC vs CPT

5. Discussió

Referències

- Anders, S, Huber, W. 2010. *Differential Expression Analysis for Sequence Count Data*. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3218662/>.
- GTEX Consortium. 2013. *The Genotype-Tissue Expression (Gtex) Project*. <https://gtexportal.org/home/>.
- McCarthy, DJ, Chen, Y, Smyth, GK. 2012. *Differential Expression Analysis of Multifactor RNA-Seq Experiments with Respect to Biological Variation*. <http://dx.doi.org/10.1093/nar/gks042>.
- Michael I Love, Wolfgang Huber, Simon Anders. 2014. *Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with Deseq2*. <http://dx.doi.org/10.1186/s13059-014-0550-8>.