Functional analysis of gene lists

Ferran Briansó and Alex Sánchez-Pla.
Statistics department. UB
& Statistics and Bioinformatics Unit (UEB). VHIR.

April 10, 2016

Contents

1 Introduction

This document provides information on how to extract subsets of genes from previously available gene lists by setting different filtering conditions such as the fold change, the p-value or the availability of `Entrez` identifier.

1.1 From gene lists to Functional Analysis

The main, but not the only, goal of creating a gene list is to use it as input for some type of functional analysis such as Enrichment Analyis (ORA) or Gene set Enrichment Analysis (GSEA).

Functional analysis can be made, on a first approach on

- A list of genes selected by being differentially expressed in a given experimental setting.
- The whole list of genes -or even the whole expression matrix- that has been used in the analysis.

Most tools require that gene list consist of gene identifiers in some standard notation such as `Entrez`, `ENSEMBL` or other related to these.

These gene lists can be easily extracted from output tables provided by microarrays or RNA-seq data analysis tools.

The analysis below is applied on a set of three gene lists obtained from a breast cancer study, but it can be easily extended to more lists or other studies.

1.2 Data Input Format for gene list selection

In principle a filtering tool might read the file header and, once this is done, create an interactive dialog to query for the values that would be applied for subsetting the lists rows or columns.

In practice, and in our work environment most lists will be extracted from the standard output of our microarray analysis pipeline[1]. These files are generically described as "Expression_and_TopTables" because they consist of tables having:

1. The Gene Symbols and the Entrez Identifiers in the first two columns
2. The standard output of the limma software known as "topTable"

optionally the Expression values that have been used to compute the Toptable. [2].

```
x1<- AvsB <- read.table(file.path(dataDir, "ExpressAndTop_AvsB.csv2"), head=T, sep=";", dec=
x2<- AvsL <- read.table(file.path(dataDir, "ExpressAndTop_AvsL.csv2"), head=T, sep=";", dec=
x3<- BvsL <- read.table(file.path(dataDir, "ExpressAndTop_BvsL.csv2"), head=T, sep=";", dec=

dim(x1);

## [1] 6221   18

cat("\nHeader of top Table for comparison AvsB\n")

##
## Header of top Table for comparison AvsB

cat("--------------------------------------------\n")

## --------------------------------------------

head(x1[1:10, 1:8])

##              SymbolsA EntrezsA     logFC  AveExpr          t      P.Value
## 204667_at       FOXA1     3169 -3.038344 8.651157 -14.362164 5.741793e-11
## 215729_s_at     VGLL1    51442  3.452290 6.137595  12.814829 3.439769e-10
## 220192_x_at     SPDEF    25803 -3.016315 9.521883 -10.859194 4.337504e-09
## 214451_at      TFAP2B     7021 -5.665059 7.432823 -10.829548 4.519412e-09
## 217528_at       CLCA2     9635 -5.622086 6.763101  -9.666128 2.431610e-08
```

[1]In this point we assume that the user is familiarized with standard microarray analysis "a la Bioconductor". If this is not so the reader can browse through the slides and examples in `http://eib.stat.ub.edu/Omics+Data+Analysis`

[2]Although some type of analyses require only the gene identifiers other need also the expressions. For this reason these output files contain "all that is needed" for further analyses

```
## 217284_x_at   SERHL2   253190 -4.313116 9.133307  -9.528373 2.996253e-08
##                adj.P.Val         B
## 204667_at   3.571969e-07 14.648730
## 215729_s_at 1.069940e-06 13.148992
## 220192_x_at 7.028816e-06 10.928314
## 214451_at   7.028816e-06 10.891489
## 217528_at   3.025409e-05  9.363419
## 217284_x_at 3.106615e-05  9.171294

cat("\nHeader of top Table for comparison AvsL\n")

##
## Header of top Table for comparison AvsL

cat("--------------------------------------------\n")

## --------------------------------------------

dim(x2); head(x2[1:10, 1:8])

## [1] 6221   18
##             SymbolsA EntrezsA     logFC  AveExpr         t      P.Value
## 205009_at       TFF1     7031  4.735050 8.692478 10.564670 6.548729e-09
## 205862_at      GREB1     9687  3.958563 6.082835  9.983993 1.513906e-08
## 205225_at       ESR1     2099  3.964939 9.300546  9.849787 1.846739e-08
## 209443_at   SERPINA5     5104  2.198392 7.586226  8.531873 1.448630e-07
## 217528_at      CLCA2     9635 -4.429254 6.763101 -7.615275 6.877151e-07
## 205696_s_at    GFRA1     2674  2.333785 6.239876  7.600491 7.058428e-07
##                adj.P.Val         B
## 205009_at   3.829521e-05 10.133204
## 205862_at   3.829521e-05  9.434088
## 205225_at   3.829521e-05  9.266376
## 209443_at   2.252982e-04  7.488507
## 217528_at   7.318414e-04  6.101859
## 205696_s_at 7.318414e-04  6.078427

cat("\nHeader of top Table for comparison BvsL\n")

##
## Header of top Table for comparison BvsL

cat("--------------------------------------------\n")

## --------------------------------------------

dim(x3); head(x3[1:10, 1:8])
```

```
## [1] 6221   18
##              SymbolsA EntrezsA     logFC   AveExpr          t      P.Value
## 204667_at       FOXA1     3169  2.961042  8.651157  13.996760 8.630583e-11
## 215729_s_at     VGLL1    51442 -3.744599  6.137595 -13.899875 9.630024e-11
## 205009_at        TFF1     7031  5.729322  8.692478  12.783054 3.575181e-10
## 205225_at        ESR1     2099  3.939276  9.300546   9.786035 2.030957e-08
## 205862_at       GREB1     9687  3.774303  6.082835   9.519268 3.038123e-08
## 218211_s_at      MLPH    79083  2.808408 10.932769   8.813968 9.162548e-08
##                 adj.P.Val         B
## 204667_at   2.995419e-07 14.040383
## 215729_s_at 2.995419e-07 13.953517
## 205009_at   7.413733e-07 12.893342
## 205225_at   3.158646e-05  9.422443
## 205862_at   3.780032e-05  9.061947
## 218211_s_at 8.534352e-05  8.062668
```

2 Input data preprocessing

Sometimes lists may need some preprocessing (e.g. in this example the gene list has multiple transcripts per gene identifier that have to be unitized previous to the analysis).

We have prepared two functions that encapsulate some standard functionalities for gene list filtering.

```
source("https://raw.githubusercontent.com/alexsanchezpla/scripts/master/usefulFunctions/gene
source("https://raw.githubusercontent.com/alexsanchezpla/scripts/master/usefulFunctions/extr
```

We can use the available functions to extract only the gene lists

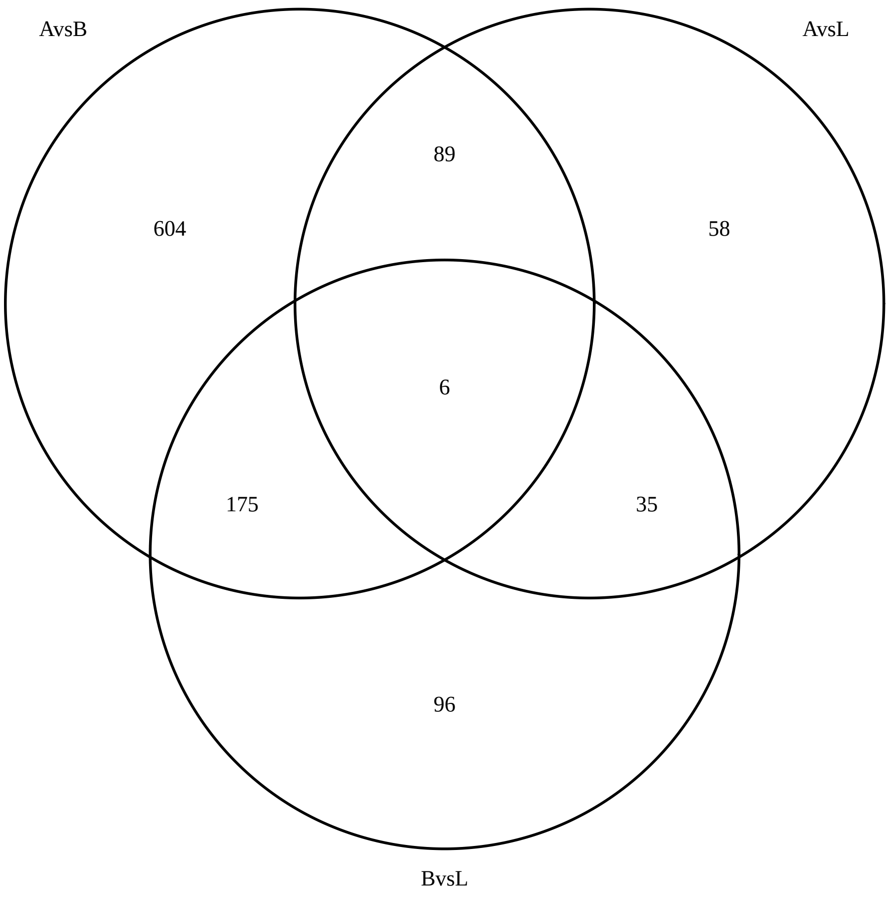
```
geneList1  <- genesFromTopTable (x1, entrezOnly = TRUE, uniqueIds=TRUE,
                                 adjOrrawP = "adj", Pcutoff = 0.1, FCcutoff = .75,
                                 id2Select = "SymbolsA" , cols2Select =3)
length(geneList1)

## [1] 874

geneList1Up  <- genesFromTopTable (x1, entrezOnly = TRUE, uniqueIds=TRUE,
                                 adjOrrawP = "adj", Pcutoff = 0.1, FCcutoff = .75, updown="u
                                 id2Select = "EntrezsA" , cols2Select =3)
length(geneList1Up)

## [1] 534

geneList1Down  <- genesFromTopTable (x1, entrezOnly = TRUE, uniqueIds=TRUE,
                                 adjOrrawP = "adj", Pcutoff = 0.1, FCcutoff = .75, updown="d
                                 id2Select = "EntrezsA" , cols2Select =3)
length(geneList1Down)

## [1] 340

geneList2 <- genesFromTopTable (x2, entrezOnly = TRUE, uniqueIds=TRUE,
                                 adjOrrawP = "adj", Pcutoff = 0.1, FCcutoff = .75,
                                 id2Select = "EntrezsA" , cols2Select =3)

geneList3 <- genesFromTopTable (x3, entrezOnly = TRUE, uniqueIds=TRUE,
                                 adjOrrawP = "adj", Pcutoff = 0.1, FCcutoff = .75,
                                 id2Select = "EntrezsA" , cols2Select =3)
```

Another possibility is to use function `extractInfo` do a “batch extraction”

```
List1 <- extractInfo(x1, "AvsB", "A|B", resultsDir, adjOrraw="adj",
                     pCutOff=0.1, fcCutoff=.75);
universeList1 <-List1[[2]]; geneList1<- List1[[1]];


cat("\nNumber of genes selectable (AvsB) with adjusted p-value < 0.1 and logFC > 0.75", leng

##
## Number of genes selectable (AvsB) with adjusted p-value < 0.1 and logFC > 0.75 874

List2 <- extractInfo(x2, "AvsL", "A|L", resultsDir, adjOrraw="adj", pCutOff=0.1, fcCutoff=.7
universeList2 <-List2[[2]]; geneList2<- List2[[1]];
cat("\nNumber of genes selectable (AvsL) with adjusted p-value < 0.1 and logFC > 0.75", leng

##
## Number of genes selectable (AvsL) with adjusted p-value < 0.1 and logFC > 0.75 188

List3 <- extractInfo(x3, "BvsL", "B|L", resultsDir, adjOrraw="adj", pCutOff=0.1, fcCutoff=.7
universeList3 <-List3[[2]]; geneList3<- List3[[1]];
cat("\nNumber of genes selectable (BvsL) with adjusted p-value < 0.1 and logFC > 0.75", leng

##
## Number of genes selectable (BvsL) with adjusted p-value < 0.1 and logFC > 0.75 312

# test
# pattern  <- "WL|PS"; cols2select<- grep(pattern, colnames(x1)); colnames(x1)[cols2select]
# pattern  <- "WL\\.M|PS\\.M"; cols2select<- grep(pattern, colnames(x1M)); colnames(x1M)[co
# pattern  <- "WL\\.F|PS\\.F"; cols2select<- grep(pattern, colnames(x1F)); colnames(x1F)[co
```

The following diagram shows which genes there are in common (or not) between the three lists.

```r
require(VennDiagram)
vd2<- venn.diagram(list(AvsB=geneList1, AvsL=geneList2,  BvsL=geneList3), filename=NULL)
grid.draw(vd2)
```



```r
dev.off()

## null device
##           1
```

3 Case study

Imagine a user wants to do the following analysis:

1. Select three lists from my study (**In this example we choose AvsB, AvsL, BvsL**) We can do a preliminar optional filtering to keep only

genes with Entrez Identifier and remove duplicates keeping only the most variable one.

```
AvsB0  <- genesFromTopTable (AvsB, entrezOnly = TRUE, uniqueIds=TRUE, adjOrrawP = "adj"
AvsL0  <- genesFromTopTable (AvsL, entrezOnly = TRUE, uniqueIds=TRUE, adjOrrawP = "adj"
BvsL0  <- genesFromTopTable (BvsL, entrezOnly = TRUE, uniqueIds=TRUE, adjOrrawP = "adj"
```

2. Filter lists with adjusted-p-value less than 0.05

```
AvsB1  <- genesFromTopTable (AvsB, entrezOnly = TRUE, uniqueIds=TRUE, adjOrrawP = "adj"
AvsL1  <- genesFromTopTable (AvsL, entrezOnly = TRUE, uniqueIds=TRUE, adjOrrawP = "adj"
BvsL1  <- genesFromTopTable (BvsL, entrezOnly = TRUE, uniqueIds=TRUE, adjOrrawP = "adj"

cat("AvsB: ", length(AvsB0), "-->", length(AvsB1), "\n")

## AvsB:  708 --> 434

cat("AvsL: ", length(AvsL0), "-->", length(AvsL1), "\n")

## AvsL:  336 --> 80

cat("BvsL: ", length(BvsL0), "-->", length(BvsL1), "\n")

## BvsL:  412 --> 132
```

3. Create separate lists with up and down regulated genes

```
AvsB1Up  <- genesFromTopTable (AvsB, entrezOnly = TRUE, uniqueIds=TRUE, adjOrrawP = "ad
AvsL1Up  <- genesFromTopTable (AvsL, entrezOnly = TRUE, uniqueIds=TRUE, adjOrrawP = "ad
BvsL1Up  <- genesFromTopTable (BvsL, entrezOnly = TRUE, uniqueIds=TRUE, adjOrrawP = "ad
cat("AvsB: ", length(AvsB1), "-->", length(AvsB1Up), "\n")

## AvsB:  434 --> 243

cat("AvsL: ", length(AvsL1), "-->", length(AvsL1Up), "\n")

## AvsL:  80 --> 44

cat("BvsL: ", length(BvsL1), "-->", length(BvsL1Up), "\n")

## BvsL:  132 --> 77
```

```
AvsB1Down  <- genesFromTopTable (AvsB, entrezOnly = TRUE, uniqueIds=TRUE, adjOrrawP = "
AvsL1Down  <- genesFromTopTable (AvsL, entrezOnly = TRUE, uniqueIds=TRUE, adjOrrawP = "
BvsL1Down  <- genesFromTopTable (BvsL, entrezOnly = TRUE, uniqueIds=TRUE, adjOrrawP = "
cat("AvsB: ", length(AvsB1), "-->", length(AvsB1Down), "\n")

## AvsB:  434 --> 191

cat("AvsL: ", length(AvsL1), "-->", length(AvsL1Down), "\n")

## AvsL:  80 --> 36

cat("BvsL: ", length(BvsL1), "-->", length(BvsL1Down), "\n")

## BvsL:  132 --> 55
```

4. Create a gene list with genes shared by AvsL and BvsL

```
commonAvsLandBvsL <- intersect(AvsL0, BvsL0)
length(commonAvsLandBvsL)

## [1] 104
```

5. The lists can be used from memory or written into files:

```
write.table(x=AvsB1Up, file = file.path(resultsDir, "AvsB1Up.txt"), row.names=FALSE, co
write.table(x=AvsL1Up, file = file.path(resultsDir, "AvsL1Up.txt"), row.names=FALSE, co
write.table(x=BvsL1Up, file = file.path(resultsDir, "BvsL1Up.txt"), row.names=FALSE, co
write.table(x=AvsB1Down, file = file.path(resultsDir, "AvsB1Down.txt"), row.names=FALSE
write.table(x=AvsL1Down, file = file.path(resultsDir, "AvsL1Down.txt"), row.names=FALSE
write.table(x=BvsL1Down, file = file.path(resultsDir, "BvsL1Down.txt"), row.names=FALSE
write.table(x=commonAvsLandBvsL, file = file.path(resultsDir, "commonAvsLandBvsL.txt"),
```