A quick gene selection, annotation and GO analysis

Alex Sánchez[1,2]

[1]Statistics and Bioinformatics Unit (VHIR)
[2] Statistics Department. University of Barcelona

Contents

```
if(!(require(printr))) {
  install.packages(
    printr,
    type = source,
    repos = c(http://yihui.name/xran, http://cran.rstudio.com)
  )
}
```

1 Introduction

Most gene expression studies undergo one phase where, after gene selection has been performed, one wishes to:

1. Annotate the genes or transcripts, that is associate, to each probeset or transcript, some identifiers in the appropriate databases that can be used to understand better the results or that are needed to proceed with further analyses (for instance GO Analysis needs "Entrez" identifiers).

2. Do some type of Gene Set Enrichment Analyses such as Overrepresentation Analysis (ORA) or classical Gene Set Enrichment Analysis (GSEA).

This document is an illustration which does not intend to be exhaustive, on how to do this with some of these packages.

1.1 Obtaining gene lists

The first step in annotation analysis is to obtain the gene lists, usually as the output of some differential expression analysis.

```
topTab <- read.table("https://raw.githubusercontent.com/alexsanchezpla/scripts/master/Exemple_Analisis_
colnames(topTab)

##  [1] "SymbolsA"  "EntrezsA"  "logFC"     "AveExpr"   "t"         "P.Value"
##  [7] "adj.P.Val" "B"         "A.PF14"    "A.PF19"    "A.PF23"    "A.PF39"
## [13] "A.PF46"    "B.PF24"    "B.PF25"    "B.PF28"    "B.PF34"    "B.PF42"

head(topTab)

##              SymbolsA EntrezsA  logFC AveExpr       t          P.Value
## 204667_at       FOXA1     3169 -3.038   8.651 -14.362 0.00000000005742
## 215729_s_at     VGLL1    51442  3.452   6.138  12.815 0.00000000034398
## 220192_x_at     SPDEF    25803 -3.016   9.522 -10.859 0.00000000433750
## 214451_at      TFAP2B     7021 -5.665   7.433 -10.830 0.00000000451941
## 217528_at       CLCA2     9635 -5.622   6.763  -9.666 0.00000002431610
## 217284_x_at    SERHL2   253190 -4.313   9.133  -9.528 0.00000002996253
##                 adj.P.Val      B A.PF14 A.PF19 A.PF23 A.PF39 A.PF46 B.PF24
## 204667_at   0.0000003572 14.649  9.822  9.514  9.919  9.601  9.592  6.484
## 215729_s_at 0.0000010699 13.149  4.737  4.761  6.255  4.820  4.848  8.266
## 220192_x_at 0.0000070288 10.928 10.484 10.915 10.511 11.510 10.265  7.824
## 214451_at   0.0000070288 10.891 10.177 10.060 11.201 10.889 10.404  4.818
## 217528_at   0.0000302541  9.363 10.534 10.036 11.326  8.053 10.619  4.581
## 217284_x_at 0.0000310662  9.171 11.727  9.741 11.436 12.819 12.687  7.274
##              B.PF25 B.PF28 B.PF34 B.PF42
## 204667_at     6.551  7.001  6.685  6.535
## 215729_s_at   8.963  8.304  8.769  8.381
## 220192_x_at   7.810  7.522  8.427  7.020
## 214451_at     4.784  4.976  4.912  4.916
## 217528_at     4.538  4.519  4.357  4.463
## 217284_x_at   7.298  7.491  7.562  7.217
```

2 Annotating the genes

This table has already been "annotated" in the script that has performed the original analysis, but, *what would we have had to do if it hadn't been?*

We might have used either a specific annotation package for the array or the BioMaRt package.

```
atVHIR <- FALSE
if (atVHIR){
    http_proxy="http://conf_www.ir.vhebron.net:8080/"
    https_proxy="http://conf_www.ir.vhebron.net:8080/"
    }
```

2.1 Using a microarray annotation package

If we hadn't had 'Entrez' Identifiers, but only the probeset identifiers which depend on the array type we might have done as follows:

```
probeIDsAll <- rownames(topTab)
probeIDsUp <- probeIDsAll [topTab$adj.P.Val<0.05 & topTab$logFC > 0]
probeIDsDown <- probeIDsAll [topTab$adj.P.Val<0.05 & topTab$logFC < 0]
```

```
require(hgu133a.db)
keytypes(hgu133a.db)

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

geneEntrezsUp <- select(hgu133a.db, keys = probeIDsUp, columns=c("ENTREZID", "SYMBOL"))

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

geneEntrezsDown <- select(hgu133a.db, keys = probeIDsUp, columns=c("ENTREZID", "SYMBOL"))

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

geneEntrezsUniverse <- select(hgu133a.db, keys = probeIDsAll, columns=c("ENTREZID", "SYMBOL"))

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

head(geneEntrezsUp)

##       PROBEID ENTREZID SYMBOL
## 1 215729_s_at    51442  VGLL1
## 2   205044_at     2568  GABRP
## 3   209337_at    11168  PSIP1
## 4   209786_at    10473  HMGN4
## 5   204061_at     5613   PRKX
## 6   207039_at     1029 CDKN2A
```

2.2 Using BiomaRt

Biomart is a powerful engine for linking identifiers. It is a bit cryptic at the first approach because in order to use it we must define *filters* (what we input for searching), *attributes* (what we output) and *values* (which values we input).

```
biodataset <- useDataset("hsapiens_gene_ensembl", useMart("ensembl"))
listDatasets(biodataset)$dataset

##  [1] "pformosa_gene_ensembl"       "ttruncatus_gene_ensembl"
##  [3] "mmurinus_gene_ensembl"       "meugenii_gene_ensembl"
##  [5] "loculatus_gene_ensembl"      "gmorhua_gene_ensembl"
##  [7] "tnigroviridis_gene_ensembl"  "tguttata_gene_ensembl"
##  [9] "btaurus_gene_ensembl"        "mmulatta_gene_ensembl"
## [11] "cporcellus_gene_ensembl"     "fcatus_gene_ensembl"
## [13] "oprinceps_gene_ensembl"      "acarolinensis_gene_ensembl"
## [15] "xmaculatus_gene_ensembl"     "pcapensis_gene_ensembl"
## [17] "eeuropaeus_gene_ensembl"     "mlucifugus_gene_ensembl"
## [19] "ptroglodytes_gene_ensembl"   "xtropicalis_gene_ensembl"
## [21] "celegans_gene_ensembl"       "scerevisiae_gene_ensembl"
## [23] "gaculeatus_gene_ensembl"     "saraneus_gene_ensembl"
## [25] "tbelangeri_gene_ensembl"     "olatipes_gene_ensembl"
## [27] "oniloticus_gene_ensembl"     "dordii_gene_ensembl"
## [29] "csabaeus_gene_ensembl"          "trubripes_gene_ensembl"
## [31] "etelfairi_gene_ensembl"         "dnovemcinctus_gene_ensembl"
## [33] "nleucogenys_gene_ensembl"       "csavignyi_gene_ensembl"
## [35] "cintestinalis_gene_ensembl"     "aplatyrhynchos_gene_ensembl"
## [37] "itridecemlineatus_gene_ensembl" "ggorilla_gene_ensembl"
## [39] "pmarinus_gene_ensembl"          "ggallus_gene_ensembl"
## [41] "sscrofa_gene_ensembl"           "ocuniculus_gene_ensembl"
## [43] "tsyrichta_gene_ensembl"         "drerio_gene_ensembl"
## [45] "vpacos_gene_ensembl"            "amexicanus_gene_ensembl"
## [47] "choffmanni_gene_ensembl"        "falbicollis_gene_ensembl"
## [49] "hsapiens_gene_ensembl"          "rnorvegicus_gene_ensembl"
## [51] "lchalumnae_gene_ensembl"        "dmelanogaster_gene_ensembl"
## [53] "pabelii_gene_ensembl"           "mmusculus_gene_ensembl"
## [55] "mgallopavo_gene_ensembl"        "lafricana_gene_ensembl"
## [57] "cfamiliaris_gene_ensembl"       "mfuro_gene_ensembl"
## [59] "sharrisii_gene_ensembl"         "amelanoleuca_gene_ensembl"
## [61] "oaries_gene_ensembl"            "mdomestica_gene_ensembl"
## [63] "psinensis_gene_ensembl"         "ogarnettii_gene_ensembl"
## [65] "panubis_gene_ensembl"           "pvampyrus_gene_ensembl"
## [67] "cjacchus_gene_ensembl"          "ecaballus_gene_ensembl"
## [69] "oanatinus_gene_ensembl"
```

```r
filters<-listFilters(biodataset)
# We need to find the filter to link with Affymetrx arrays hgu133a
u133aFilters<- grep("u133a", filters[,1] )
u133aFilters <- filters[u133aFilters,]
myu133aFilter <- u133aFilters[3,1]
myu133aFilter
```

```
## [1] "affy_hg_u133a"
```

```r
atributs<- listAttributes(biodataset)
entrezAtributs<- grep("entrez", atributs[,1])
entrezAtribut <- atributs[entrezAtributs,]
myentrezAtribut <- entrezAtribut[2,1]
myentrezAtribut
```

```
## [1] "entrezgene"
```

```r
# Now we can do the search
entrezfromProbesUp <- getBM(filters= myu133aFilter,
                           attributes= c(myentrezAtribut, myu133aFilter),
                           values= probeIDsUp,
                           mart= biodataset,uniqueRows=TRUE)
head(entrezfromProbesUp)
```

```
##   entrezgene affy_hg_u133a
## 1       6201   200082_s_at
## 2      54881     218104_at
## 3      10054   201177_s_at
## 4         NA   214511_x_at
## 5      23231     212314_at
## 6       2869   204396_s_at
```

2.3 The gene list for pathway Analysis

In this example we had already had the Entrez and Symbol identifiers so we can extract these directly from the topTable.

Although we skip it here it may be interesting to compare the entrez identifiers obtained from the three distinct approaches. They should be identical, but there may be small discrepancies...

```
geneListUp <- topTab$EntrezsA [topTab$adj.P.Val<0.05 & topTab$logFC > 0]
head(geneListUp)

## [1] 51442  2568 11168 10473  5613  1029

geneListDown <- topTab$EntrezsA [topTab$adj.P.Val<0.05 & topTab$logFC < 0]
length(geneListDown)

## [1] 268

geneUniverse <- topTab$EntrezsA
length(geneUniverse)

## [1] 6221

write.csv(geneListUp, file="selectedAvsB.up.csv")
write.csv(geneListDown, file="selectedAvsB.down.csv")
write.csv(geneUniverse, file="geneUniverse.csv")
```

3 Pathway Analysis

We start by removing NA's (if any) and ensuring that we have unique identifiers.

```
# Remove potential NAs values
geneEntrezsUp <- unique(geneListUp[!is.na(geneListUp)])
geneEntrezsDown <- unique(geneListDown[!is.na(geneListDown)])
geneEntrezsUniverse <- unique(geneUniverse[!is.na(geneUniverse)])
```

We will use the `GOstats package` which proceeds in two steps:

1. First we create the appropriate objects
2. Next we use them to do the enrichment analysis
3. In a final step we generate an html report with the test results

First we create the appropriate objects

```
require(GOstats)
## Creamos los "hiperparametros" en que se basa el analisis
GOparams = new("GOHyperGParams",
               geneIds=geneEntrezsUp, universeGeneIds=geneEntrezsUniverse,
               annotation="org.Hs.eg.db", # might have use hgu133a.db instead
               ontology="BP",
               pvalueCutoff=0.001, conditional=FALSE,
               testDirection="over")
KEGGparams = new("KEGGHyperGParams",
                 geneIds=geneEntrezsUp, universeGeneIds=geneEntrezsUniverse,
                 annotation="org.Hs.eg.db", # might have use hgu133a.db instead
                 pvalueCutoff=0.01, testDirection="over")
```

Next we use them to do the enrichment analysis

```
GOhyper = hyperGTest(GOparams)
KEGGhyper = hyperGTest(KEGGparams)
cat("GO\n")

## GO

print(head(summary(GOhyper)))

##       GOBPID       Pvalue OddsRatio ExpCount Count Size
## 1 GO:0000278 0.0000001734     2.263   32.925    63  564
## 2 GO:0000070 0.0000042038     4.321    4.845    17   83
## 3 GO:0000819 0.0000051295     4.032    5.429    18   93
## 4 GO:0007049 0.0000066093     1.846   51.840    82  888
## 5 GO:0035556 0.0000069666     1.757   68.010   101 1165
## 6 GO:0051782 0.0000072695     5.411    3.094    13   53
##                                  Term
## 1                   mitotic cell cycle
## 2 mitotic sister chromatid segregation
## 3         sister chromatid segregation
## 4                           cell cycle
## 5    intracellular signal transduction
## 6 negative regulation of cell division

cat("KEGG\n")

## KEGG

print(head(summary(KEGGhyper)))

##
## KEGG.db contains mappings based on older data because the original
##  resource was removed from the the public domain before the most
##  recent update was produced.  This package should now be considered
##  deprecated and future versions of Bioconductor may not have it
##  available.  Users who want more current data are encouraged to look
##  at the KEGGREST or reactome.db packages

##   KEGGID   Pvalue OddsRatio ExpCount Count Size
## 1  04110 0.001294     2.878    5.724    14   89
## 2  04114 0.002082     3.167    4.116    11   64
## 3  04914 0.002461     3.590    3.023     9   47
## 4  04010 0.004909     2.352    7.267    15  113
## 5  04062 0.006140     2.452    6.045    13   94
## 6  04971 0.007421     4.082    1.801     6   28
##                                     Term
## 1                             Cell cycle
## 2                         Oocyte meiosis
## 3 Progesterone-mediated oocyte maturation
## 4                   MAPK signaling pathway
## 5              Chemokine signaling pathway
## 6                   Gastric acid secretion
```

In a final step we generate an html report with the test results

```r
# Creamos un informe html con los resultados
GOfilename =file.path(paste("GOResults.AvsB.up",".html", sep=""))
KEGGfilename =file.path(paste("KEGGResults.AvsB.up",".html", sep=""))
htmlReport(GOhyper, file = GOfilename, summary.args=list("htmlLinks"=TRUE))
htmlReport(KEGGhyper, file = KEGGfilename, summary.args=list("htmlLinks"=TRUE))
```