Next Generation Sequencing Analysis - Final exercise

Alvaro Ponce Cabrera

January 31, 2016

1 Perform a complete RNA-Seq Analysis

Write a report describing the steps you have performed, interpret the results and include some discussion making reference to the tables and figures you think are necessaries to be showed in the report.

1.1 Load the data

```
# setwd("~/PractiseData")
load("gbm.Rdata")
#rse object has been created from gbm.Rdata
```

1.2 Get Counts

The expression level in RNA-Seq experiments is cuantified by counts, number of reads which are mapped to our reference genome. In this case, we have the count data, and it's not necessary to align in order to obtain the counts. Counts can be found onto rse object, using "colData" function.

```
library(SummarizedExperiment) #Contain some functions to work with summarized experiment objects

pheno = colData(rse) #Obtaining the pheno data from rse.

#In rse we have information about 5 subtypes of glyoblastoma tumor, but we are only interested
#in 4 of them: Classical, Mesenchymal, Neural and Proneural, so we eliminate "G-CIMP" type and NAs
#using the next mask.

mask<-!pheno$Cluster%in%c("G-CIMP","NA") & !is.na(pheno$Cluster)


#Final object

rse.s<-rse[,mask]
pheno.s<-pheno[mask,]

(rse.s)

## class: RangedSummarizedExperiment
## dim: 20330 162
## metadata(3): Query: TCGAprepareParameters FilesInfo:
## assays(2): raw_counts scaled_estimate
## rownames(20330): A1BG|1 A1CF|29974 ... ZZEF1|23140 ZZZ3|26009
```

```
## rowRanges metadata column names(3): gene_id entrezgene
##   transcript_id.transcript_id_TCGA-06-0171-02A-11R-2005-01
## colnames(162): TCGA-06-0171-02A-11R-2005-01 TCGA-76-4925-01A-01R-1850-01 ...
##   TCGA-19-2625-01A-01R-1850-01 TCGA-28-5213-01A-01R-1850-01
## colData names(17): patient sample ... Gcimp2012 stringAsFactor

#Counts can be obtained using "assays" funtion.We can se that we have "raw_counts" into "assays" when
#we execute rse.s.

cc<-assays(rse.s)$raw_counts
dim(cc) #Check the dimension

## [1] 20330   162

#Get group and confirm we have only the interested ones

group<-colData(rse.s)$Cluster
table(group)

## group
##   Classical      G-CIMP Mesenchymal          NA      Neural   Proneural
##          43           0          55           0          32          32
```

1.3 Prepare for CQN normalization. Annotation.

Finally, we have the count data, now it is necessary the normalization of the counts. In this analysis, we are going to do a CQN normalization, that corrects for library size, gene length, GC-content. Before the normalization, we need some annotation data (gene length and GC-content), so we need to obtain it

```
#Rownames of cc (count data) has the next format "Gene symbol | EntrezID". We need to use one of
#these to obtain normalization data.

#We separate both gene symbol and entrezID of each row and put it in a vector.
b<- unlist(strsplit(rownames(cc), split='|', fixed=TRUE))

#The even elements correspond to entrezID, we select it.

m<-seq(2,length(b),by=2)

#Acces to the gene symbol and entrezID vector only in even positions to obtain the entrezID of each
#row from cc.

entrezID<-b[m]

#Annotation using biomaRt package

library(biomaRt)
#Selecting the data base
listMarts()

##                biomart               version
## 1 ENSEMBL_MART_ENSEMBL      Ensembl Genes 83
## 2     ENSEMBL_MART_SNP  Ensembl Variation 83
## 3 ENSEMBL_MART_FUNCGEN Ensembl Regulation 83
```

```
## 4    ENSEMBL_MART_VEGA               Vega 63
## 5                pride        PRIDE (EBI UK)

mart  <- useMart(biomart="ensembl")
mart

## Object of class 'Mart':
##  Using the ENSEMBL_MART_ENSEMBL BioMart database
##  Using the  dataset

#We select ensembl database, now, we need to find the data for homo sapiens
a<-listDatasets((mart))
grep("sapien",a[,1], value=T)

## [1] "hsapiens_gene_ensembl"

HS<-useMart(biomart="ensembl",dataset = "hsapiens_gene_ensembl")

#Finally, we need the gene length and GC content data
grep("length",listAttributes(HS)$name,value=T)

## [1] "transcript_length" "transcript_length" "cds_length"        "transcript_length"
## [5] "cds_length"

grep("gc_content",listAttributes(HS)$name,value=T)

## [1] "percentage_gc_content" "percentage_gc_content" "percentage_gc_content"
## [4] "percentage_gc_content"

#And we need the filter in order to obtain the results using our entrezIDs

f = listFilters(HS)
grep("entrez",f$name, value = T)

## [1] "with_entrezgene"                 "with_entrezgene_transcript_name"
## [3] "entrezgene"                      "entrezgene_transcript_name"

#"getBM" function give us our annotation data
geneAnno <- getBM(
                  attributes=c("entrezgene","transcript_length", "percentage_gc_content","hgnc_symbol"),
                  filters="entrezgene",
                  values=entrezID, mart=HS)


#Now, we need to order this annotation data and match it with our count data

#Eliminate entrezID genes which don't appear in annotation data.
genOk<- intersect(entrezID, geneAnno$entrezgene)
#Match and get possitions of this genes. (entrezID object = rows from cc)
genOk<- match(genOk,entrezID)

ccOk<-cc[genOk,]  #Acces to this possitions
dim(ccOk)

## [1] 14067   162
```

```r
#We do the same over entrezID object, so this object correspond yet to the rows of cc (now, ccOk)
entrezID<-entrezID[genOk]
length(entrezID) #Check that rownames(ccOk) has the same length that entrezID.
```

```
## [1] 14067
```

```r
z<-match(entrezID, geneAnno$entrezgene) #Now, we match ccOk rows with the genes of annotation data
length(z) #Check
```

```
## [1] 14067
```

```r
anot.ok<-geneAnno[z,] #Acces to geneAnno data

dim(anot.ok)
```

```
## [1] 14067     4
```

```r
dim(ccOk)
```

```
## [1] 14067   162
```

```r
#Now, we have anot.ok and ccOk object, ccOk has count data and anot.ok has annotation data
#of the genes, both object has the same order of the genes, we refer to rows.

##Anyway, a verification is needed in order to check that our assumption is true.


#We obtain the entrezID from ccOk rownames.
b1<- unlist(strsplit(rownames(ccOk), split='|', fixed=TRUE))
m1<-seq(2,length(b1),by=2)
test<-b1[m1]
head(test)
```

```
## [1] "1"      "29974"  "2"      "144568" "51146"  "65985"
```

```r
#We check if test (contains the entrezID from ccOk) and anot.ok$entrezgene
#(contains the entrezID from anot.ok) are identical.
identical(test,as.character(anot.ok$entrezgene))
```

```
## [1] TRUE
```

```r
#It's true
#We check if there are any changes between both vectors, we actually check the order of the rows.
any(!test%in%as.character(anot.ok$entrezgene))
```

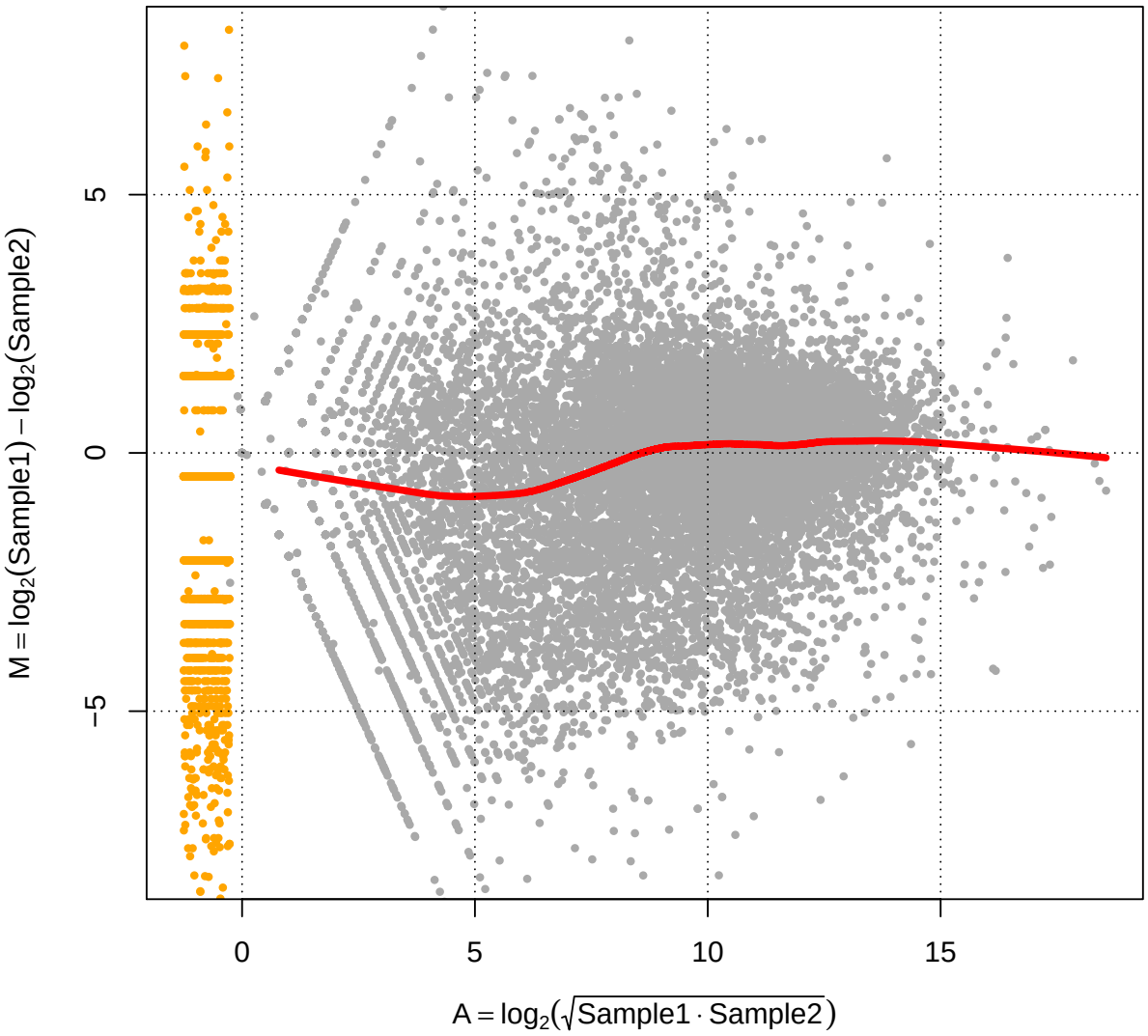
```
## [1] FALSE
```

```r
#And it is false, there are no changes between vectors, so our assumption is true, we prepared
#the data successfully
```

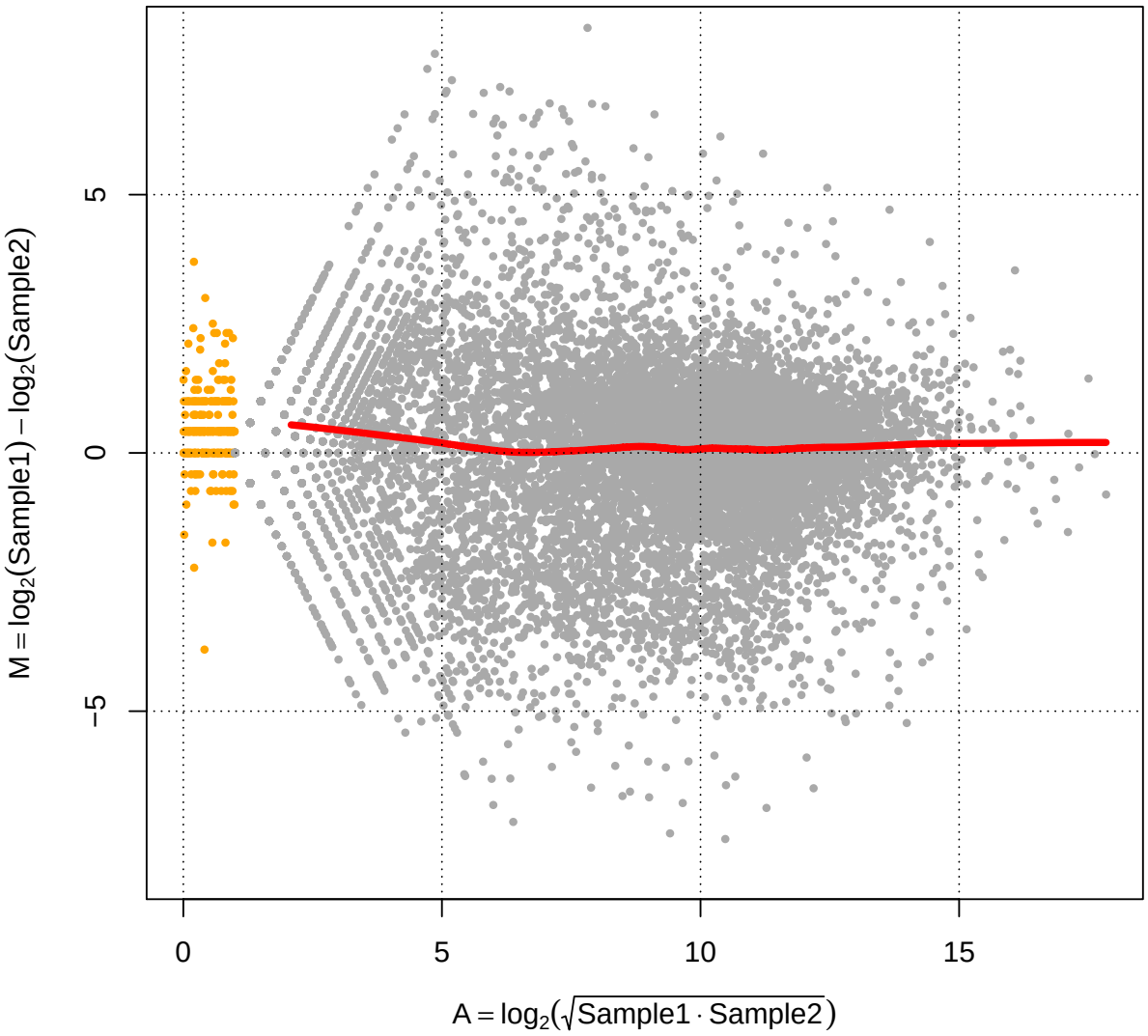
1.4 Normalization and filtering

Finally, we can normalize the data. After normalization a filtering is needed in order to eliminate those genes that do not have enought expressions levels to work with.

```r
#Preparation to use "normalizeCounts" function

annotation.ok <- anot.ok[,c("transcript_length", "percentage_gc_content")]
rownames(annotation.ok)<-entrezID
rownames(ccOk)<-entrezID

library(tweeDEseq)
library(cqn)
#Normalization by CQN method.
counts.cqn <- normalizeCounts(ccOk, method="cqn", annot=annotation.ok)

## Using cqn normalization.

## RQ fit ..........................................................................................
## SQN

## Using 'sigma' instead 'sig2' (= sigma^2) is preferred now

## .

dim(counts.cqn)

## [1] 14067   162

##We can check the normalization  by using MAplots, the first one is the MAplot before normalization,
#the second one is the MAplot after normalization.

library(edgeR)

maPlot(cc[,1], cc[,2], pch=19, cex=.5, ylim=c(-8,8),
       allCol="darkgray", lowess=TRUE,
       xlab=expression(A == log[2] (sqrt(Sample1 %.% Sample2))),
       ylab=expression(M == log[2](Sample1)-log[2](Sample2)))
grid(col="black")
```




```
maPlot(counts.cqn[,1], counts.cqn[,2], pch=19, cex=.5, ylim=c(-8,8),
       allCol="darkgray", lowess=TRUE,
       xlab=expression(A == log[2] (sqrt(Sample1 %.% Sample2))),
       ylab=expression(M == log[2](Sample1)-log[2](Sample2)))
grid(col="black")
```



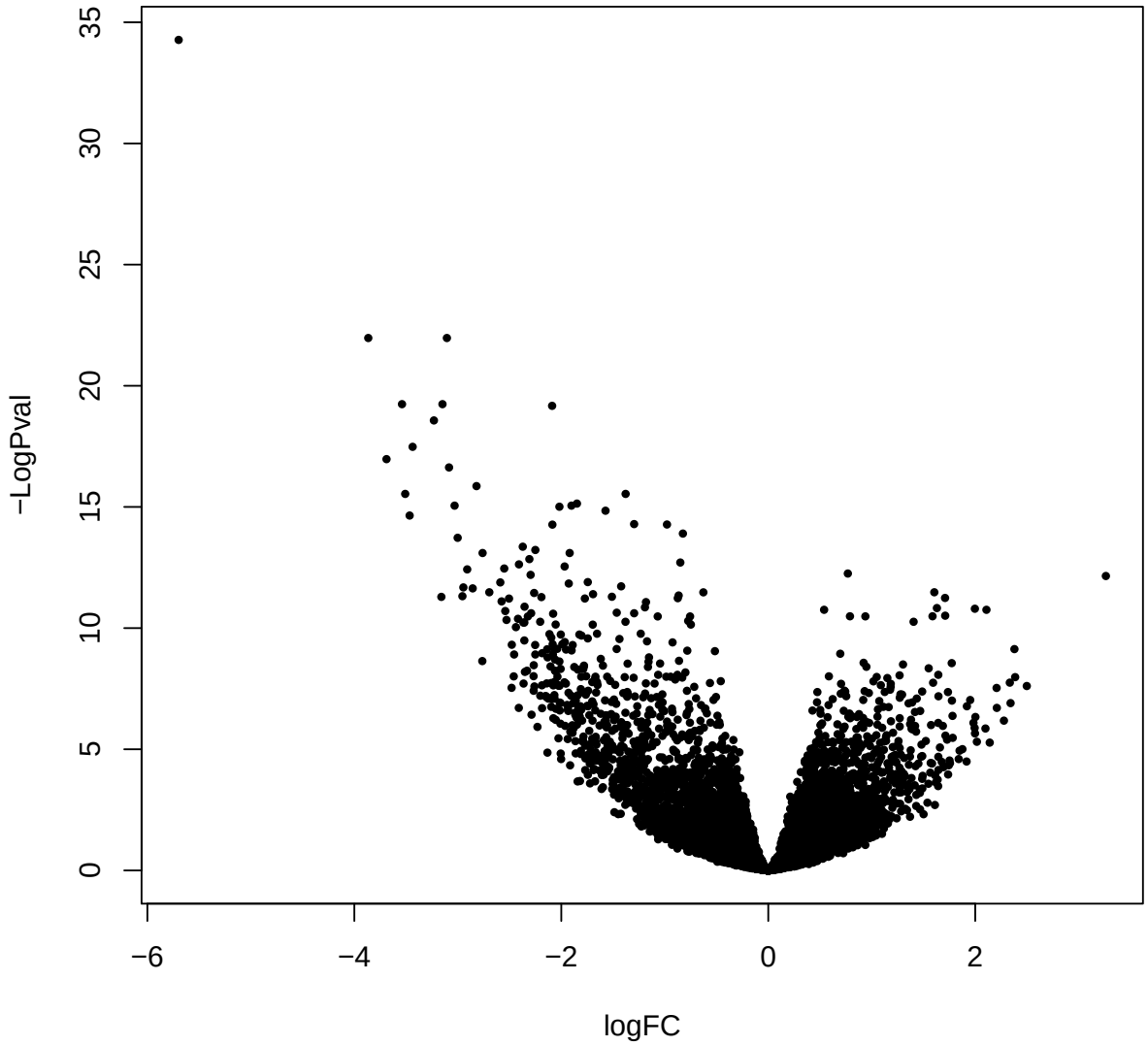

```
#"maPlot" function is used to compare the gene expression in two individuals, so we expect a line
#close to 0 and linear. We can see that we don't have this assumption before normalization,
#that's why a normalization was needed. We can see a better plot after normalization.

#Finally, the filter is done by "filterCounts" function
counts.cqn.f <- filterCounts(counts.cqn)
```

1.5 DEG with DESeq2 package

The diferential expresion gene (DEG) analysis can be performed using different packages and approaches. In this case, the DEG analysis was done by DESeq2 package.

```r
library(DESeq2)
#At the beginning, we eliminated those groups that we weren't interested in,
#but we didn't eliminate the levels of these groups from pheno.s$Cluster,
#so we do it now.


pheno.s$Cluster<-droplevels(pheno.s$Cluster)

#Creation of DEDeqDataSet object

dds <- DESeqDataSetFromMatrix(
  countData = counts.cqn.f,
  colData = pheno.s,
  design = ~ Cluster)

## converting counts to integer mode

#The design was pre-defined, so now we need to do the statistical analysis using "DESEq" function

dds <- DESeq(dds)

## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
## - replacing outliers and refitting for 567 genes
## - DESeq argument 'minReplicatesForReplace' = 7
## - original counts are preserved in counts(dds)
## estimating dispersions
## fitting model and testing

#Results can be visualized by executing "results"

resCP  <- results(dds, contrast= c("Cluster","Classical","Proneural"), pAdjustMethod = "fdr")

resCN <- results(dds,contrast= c("Cluster","Classical","Neural"),pAdjustMethod = "fdr")

resCM <- results(dds,contrast= c("Cluster","Classical","Mesenchymal"),pAdjustMethod = "fdr")

resCM

## log2 fold change (MAP): Cluster Classical vs Mesenchymal
## Wald test p-value: Cluster Classical vs Mesenchymal
## DataFrame with 12808 rows and 6 columns
##            baseMean log2FoldChange      lfcSE       stat     pvalue         padj
##           <numeric>      <numeric>  <numeric>  <numeric>  <numeric>    <numeric>
## 1        399.620664      0.2462723  0.2114458  1.1647065 0.24413781 0.3734524224
## 2      39754.269319     -0.6248559  0.1613813 -3.8719232 0.00010798 0.0006933422
## 144568   127.753124      0.2499915  0.2977878  0.8394956 0.40119126 0.5380020541
## 51146      4.095841     -0.3985969  0.1917136 -2.0791266 0.03760571 0.0875967469
## 65985    770.290895     -0.1107870  0.1043510 -1.0616774 0.28838217 0.4234410393
## ...             ...            ...        ...        ...        ...          ...
## 79364   1313.333568    0.180547648 0.06611383  2.7308605 0.00631692   0.02054001
```

```
## 440590     5.056987   -0.320258280 0.23659095 -1.3536370 0.17585220   0.29039647
## 7791   13995.066932    0.198104375 0.14314540  1.3839381 0.16637740   0.27899472
## 23140   1687.376265    0.049666940 0.08244754  0.6024066 0.54690351   0.67170957
## 26009   1139.318024   -0.007307875 0.05612944 -0.1301968 0.89641071   0.93297809

dim(resCP[(resCP$padj)<0.05,])

## [1] 3744    6

dim(resCN[(resCN$padj)<0.05,])

## [1] 5228    6

dim(resCM[(resCM$padj)<0.05,])

## [1] 4785    6

resCP<-resCP[order(resCP$padj),]
resCN<-resCN[order(resCN$padj),]
resCM<-resCM[order(resCM$padj),]


#If we want to plot the DEG, we can create a Volcano-plot.
#In this exercise, I think the goal is to create an enrichment analysis and write some conclusions.
#Anyway, I consider that show how to create a Volcano-plot could be interesting here,
#because in DESeq2package there is no function to generate a Volcano-plot,
#and you need to write the code.
#It's very simple, but as I said, I think it could be interesting,
#so here we are only a demostration in resCP case.

volcanoData <- cbind(resCP$log2FoldChange, -log10(resCP$padj))

colnames(volcanoData) <- c("logFC", "-LogPval")

plot(volcanoData, pch=19, cex=0.5)
```



1.6 Enrichment analysis

In this exercise, we are trying to analyze phenotype data from different subtypes of gliobastoma, so perform an enrichment analysis using DEG data is interesting. We can obtain information about Biological Process, Molecular Functions, or Cellular Components. To do the analysis we are going to use Gostats package, and org.Hs.eg.db as Homo Sapiens data base. Also, we are going to do only Biological Process GO analysis, we want to find any similitude between our analisys and the original one so checking for some related genes (we will do it later) and looking for some phenotype explanation of these genotype will be enough.

```
#######enriquecimiento

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

```r
#We can create a threshold in order to be more or less restrictives on which genes
#we consider differentialy expressed. In this case we will consider an adjuts p-value
#equal to 0.05 and a fold change equal to log2(2) as threshold.

maskCP<- resCP$padj < 0.05 &
  abs(resCP$log2FoldChange) > log2(2)


maskCN<- resCN$padj < 0.05 &
  abs(resCN$log2FoldChange) > log2(2)

maskCM<- resCM$padj < 0.05 &
  abs(resCM$log2FoldChange) > log2(2)

#Apply the mask to the results

deGenesCP<-rownames(resCP[maskCP,])
deGenesCN<-rownames(resCN[maskCN,])
deGenesCM<-rownames(resCM[maskCM,])

#Create the univers
geneUniverseCP <- rownames(resCP)
geneUniverseCN <- rownames(resCN)
geneUniverseCM <- rownames(resCM)


#Comparisons
paramsCP <- new("GOHyperGParams", geneIds=deGenesCP,
              universeGeneIds=geneUniverseCP,
              annotation="org.Hs.eg.db", ontology="BP",
              pvalueCutoff=0.05, conditional=FALSE,
              testDirection="over")


paramsCN <- new("GOHyperGParams", geneIds=deGenesCN,
              universeGeneIds=geneUniverseCN,
              annotation="org.Hs.eg.db", ontology="BP",
              pvalueCutoff=0.05, conditional=FALSE,
              testDirection="over")


paramsCM <- new("GOHyperGParams", geneIds=deGenesCM,
              universeGeneIds=geneUniverseCM,
              annotation="org.Hs.eg.db", ontology="BP",
              pvalueCutoff=0.05, conditional=FALSE,
              testDirection="over")


#Hypergeometric test
hgOverCP <- hyperGTest(paramsCP)

hgOverCN <- hyperGTest(paramsCN)
```

```r
hgOverCM <- hyperGTest(paramsCM)



#Creation of the reports as .html files.
htmlReport(hgOverCP, file="res.html")
htmlReport(hgOverCN, file="res.html")
htmlReport(hgOverCM, file="res.html")

#The .html files generated are into the .zip archive where this document is found.
```

1.7 Conclusions. Genes of interest.

We did an ontology analysis of the biological procces of each DEG data resulting of the comparisons.

In the first one (Classical-Proneural), we can see many neural proccess that looks normal taking account we are talking about nervious samples. Also, we can find some growth procces differentialy expressed, maybe related to the especific expression of IDH1 and PDGFRA genes.

The second one (Classical-Neural): the Neural group was characterized by the expression of several gene types that are also typical of the brainâĂŹs normal, that's why we find a lot of normal biological procces of the brain in this report.

The third one (Classical-Mesenchymal): frequent mutations in the PTEN and TP53 tumor suppressor genes also occurred in the group, increasing survival after agressive treatment, we can see in the report so many inmunology procces diferentialy expressed, maybe the increase survival is due to these inmunology procces.

As we mentioned before, at the original paper some specific genes glioblastoma subtypes were described, NF1, TP53, PTEN, IDH1 and PDGFRA. In the reports generated, we can see related behaviour with these genes.

Finally, we can check if our asumptions could be possible checking if these genes are found on our data.

```r
#Looking fore some genes in our data

#TP53 can be found at Mesenchymal and Proneural group
  #Mesenchymal

    grep("\\bTP53\\b",anot.ok$hgnc_symbol)

## [1] 12582

    anot.ok[10591,] #EntrezID 7157

##       entrezgene transcript_length percentage_gc_content hgnc_symbol
## 71274       6323              8533                 34.85       SCN1A

    #We do not found it
    any(deGenesCM=="7157")

## [1] FALSE

    #But this is because of our threshold, actually we can found it in resCM,
    #and the adjust p-value is close to 0.05, so we cannot be totally sure to
    #say whether this gene is not differentialy expressed
    resCM["7157",]
```

```
## log2 fold change (MAP): Cluster Classical vs Mesenchymal
## Wald test p-value: Cluster Classical vs Mesenchymal
## DataFrame with 1 row and 6 columns
##       baseMean log2FoldChange     lfcSE      stat     pvalue       padj
##      <numeric>      <numeric> <numeric> <numeric>  <numeric>  <numeric>
## 7157  2684.334      0.3352282 0.1456706  2.301275 0.02137609 0.05587057
```

```
  #Proneural
    #We do not found it
    any(deGenesCP=="7157")
```

```
## [1] FALSE
```

```
    #It's the same like Mesenchymal, but now the adjust p-value is higher.
    resCP["7157",]
```

```
## log2 fold change (MAP): Cluster Classical vs Proneural
## Wald test p-value: Cluster Classical vs Proneural
## DataFrame with 1 row and 6 columns
##       baseMean log2FoldChange     lfcSE      stat     pvalue       padj
##      <numeric>      <numeric> <numeric> <numeric>  <numeric>  <numeric>
## 7157  2684.334      0.3769816 0.1659983  2.270996 0.02314722 0.07094271
```

```
    #This result isn't strange, because TP53 is only differentialy expressed in 53% of proneural tumors.

#PTEN, especific of Mesenchymal

    grep("\\PTEN\\b",anot.ok$hgnc_symbol)
```

```
## [1] 9709
```

```
    anot.ok[8198,] #entrezID=5728
```

```
##      entrezgene transcript_length percentage_gc_content hgnc_symbol
## 6490      10762              5233                 43.62       NUP50
```

```
    #We do not found it
    any(deGenesCM=="5728")
```

```
## [1] FALSE
```

```
    #But, we have it in our data, again, the threshold didn't let the gene stay into the deGenes
    #used in the enrichment analysis. Now due to the fold change threshold.
    resCM["5728",]
```

```
## log2 fold change (MAP): Cluster Classical vs Mesenchymal
## Wald test p-value: Cluster Classical vs Mesenchymal
## DataFrame with 1 row and 6 columns
##       baseMean log2FoldChange     lfcSE      stat     pvalue      padj
##      <numeric>      <numeric> <numeric> <numeric>  <numeric> <numeric>
## 5728  2055.285     -0.2773733 0.1149015  -2.41401 0.01577801 0.0434124
```

```
#NF1, especific of Mesenchymal subtype
    grep("\\NF1\\b",anot.ok$hgnc_symbol)
```

```
## [1] 1030 1398 7883
```

```
    anot.ok[1187,] #entrezID=114897
```

```
##       entrezgene transcript_length percentage_gc_content hgnc_symbol
## 96864      84446              2977                 54.48       BRSK1
```

```
    any(deGenesCM=="114897")
```

```
## [1] FALSE
```

```
    #Again, the adjust p-value didn't let the gene cross the threshold
    resCM["114897",]
```

```
## log2 fold change (MAP): Cluster Classical vs Mesenchymal
## Wald test p-value: Cluster Classical vs Mesenchymal
## DataFrame with 1 row and 6 columns
##         baseMean log2FoldChange     lfcSE      stat     pvalue       padj
##        <numeric>      <numeric> <numeric> <numeric>  <numeric>  <numeric>
## 114897  3011.228     -0.4756136 0.2359527 -2.015715 0.04382975 0.09900731
```

```
#IDH1, specific Proneural gene.
    grep("\\IDH1\\b",anot.ok$hgnc_symbol)
```

```
## [1] 5761
```

```
    anot.ok[5004,] #entrezID=3417
```

```
##       entrezgene transcript_length percentage_gc_content hgnc_symbol
## 88374      64841              3860                 37.88     GNPNAT1
```

```
    any(deGenesCM=="3417")
```

```
## [1] FALSE
```

```
    #The same case.
    resCP["3417",]
```

```
## log2 fold change (MAP): Cluster Classical vs Proneural
## Wald test p-value: Cluster Classical vs Proneural
## DataFrame with 1 row and 6 columns
##       baseMean log2FoldChange     lfcSE      stat     pvalue       padj
##      <numeric>      <numeric> <numeric> <numeric>  <numeric>  <numeric>
## 3417  5946.494      0.2962718 0.1405528  2.107904 0.03503932 0.09750363
```

```
#PDGFRA, specific Proneural gene.
    grep("\\PDGFRA\\b",anot.ok$hgnc_symbol)
```

```
## [1] 8887
```

```
    anot.ok[7538,] #entrezID=5156
```

```
##       entrezgene transcript_length percentage_gc_content hgnc_symbol
## 85362       4482              1536                 43.55        MSRA
```

```
    any(deGenesCP=="5156")
```

```
## [1] TRUE
```

```
    #This one is true, anyway we are going to check it in resCP
    resCP["5156",]
```

```
## log2 fold change (MAP): Cluster Classical vs Proneural
## Wald test p-value: Cluster Classical vs Proneural
## DataFrame with 1 row and 6 columns
##       baseMean log2FoldChange     lfcSE      stat       pvalue         padj
##      <numeric>      <numeric> <numeric> <numeric>    <numeric>    <numeric>
## 5156  8961.346      -2.065419 0.3421916 -6.035855 1.581231e-09 8.068686e-08

#As we can see, all the genes are differentialy expressed, but not all of these crossed
#the threshold, maybe with a less restrict threshold the results of the enrichment
#analysis will be more clear, anyway, we saw a correlation between our analysis and
#the results given by the authors in the original paper.
```