RNA-seq Analysis from recount3

Mateo Jiménez Sotelo

2026-02-01

RNA-seq course project

LCG22

UNAM

Dataset: SRP068976
Platform: recount3

RNA-seq Analysis from recount3

Introduction

RNA-seq allows the quantification of gene expression levels across the entire transcriptome and enables the identification of genes that are differentially expressed between biological conditions. Differential expression analysis is a key tool to understand molecular mechanisms underlying diseases.

Using edgeR for normalization and limma-voom for linear modeling, I aimed to identify genes that are differentially expressed between tumor and normal liver tissue and to look out for the global transcriptional differences between these two conditions.

Uploading data

Libraries

The specified task in the project is to perform an *RNA-seq* analysis using data from `recount3`. First I will load the necessary libraries and access the recount3 data.

```
library("recount3")
library("edgeR")
library("limma")
library("ggplot2")
library("pheatmap")
library("RColorBrewer")
```

Data access

```
human_projects <- available_projects()
```

```
2026-02-23 07:59:16.164301 caching file sra.recount_project.MD.gz.
```

```
2026-02-23 07:59:16.809742 caching file gtex.recount_project.MD.gz.
```

```
2026-02-23 07:59:17.528999 caching file tcga.recount_project.MD.gz.
```

Now I have to select a specific project (SRP*) to analyze and load its data. Thanks to the* metadata*, it is possible to search within the available projects to find one that matches the desired criteria.

Lets explore the different columns (metadata) on `human_projects` dataframe to understand what information is available to select a project for our analysis.

```
colnames(human_projects)
```

```
[1] "project"      "organism"     "file_source"  "project_home" "project_type"
[6] "n_samples"
```

Now we can properly filter projects…

```
# Search within the available projects (datafrmame notation instead of 'subset()')
sra_projects <- human_projects[
  human_projects$project_type == "data_sources" &
  human_projects$organism == "human" &
  human_projects$file_source == "sra",
]
```

And look at the different projects found:

```
head(sra_projects)
```

```
    project organism file_source     project_home project_type n_samples
1 SRP107565    human         sra data_sources/sra data_sources       216
2 SRP149665    human         sra data_sources/sra data_sources         4
3 SRP017465    human         sra data_sources/sra data_sources        23
4 SRP119165    human         sra data_sources/sra data_sources         6
5 SRP133965    human         sra data_sources/sra data_sources        12
6 SRP096765    human         sra data_sources/sra data_sources         7
```

From what we learned in class, the selected data is going to depend on:

1. At least a boolean variable to be able to separate the samples into two different groups
2. Reasonable samples in both groups in order to achieve enough statistical power for the analysis

```
head(sra_projects[order(sra_projects$n_samples, decreasing = TRUE),])
```

```
       project organism file_source     project_home project_type n_samples
3068 SRP166108    human         sra data_sources/sra data_sources      8464
5105 ERP107748    human         sra data_sources/sra data_sources      7598
310   SRP066781    human         sra data_sources/sra data_sources      3854
6731 SRP079058    human         sra data_sources/sra data_sources      3589
4458 ERP017126    human         sra data_sources/sra data_sources      3514
4024 ERP108653    human         sra data_sources/sra data_sources      2687
```

The one with the most samples has an $n = 8464$. This is a huge number and even though that is in theory a good thing, we want to avoid mashed up studies (which may have many different tissues, conditions and confounders). For the purpose of the project, I'll be looking for a more manageable number of samples with an n between 100 and 500, and with a clear variable of interest to separate the samples into two groups.

```
head(sra_projects[sra_projects$n_samples >= 50 & sra_projects$n_samples <= 100,], 20)
```

```
      project organism file_source     project_home project_type n_samples
14  SRP125965    human         sra data_sources/sra data_sources        72
18  SRP057065    human         sra data_sources/sra data_sources        51
24  SRP157965    human         sra data_sources/sra data_sources        63
47  SRP201365    human         sra data_sources/sra data_sources        79
148 SRP065812    human         sra data_sources/sra data_sources        52
177 SRP144212    human         sra data_sources/sra data_sources        56
178 SRP119312    human         sra data_sources/sra data_sources        60
208 SRP109318    human         sra data_sources/sra data_sources        75
227 ERP104818    human         sra data_sources/sra data_sources        96
229 SRP132018    human         sra data_sources/sra data_sources        56
232 SRP055918    human         sra data_sources/sra data_sources        80
235 SRP114518    human         sra data_sources/sra data_sources        91
275 SRP066118    human         sra data_sources/sra data_sources        54
305 SRP074581    human         sra data_sources/sra data_sources        50
329 SRP111481    human         sra data_sources/sra data_sources        96
345 SRP061881    human         sra data_sources/sra data_sources        56
347 SRP109781    human         sra data_sources/sra data_sources        57
364 SRP058181    human         sra data_sources/sra data_sources        73
381 SRP081020    human         sra data_sources/sra data_sources        55
399 SRP049820    human         sra data_sources/sra data_sources        83
```

After a bit of research, I decided to pick the project `SRP068976`, which has 99 samples and is titled "RNA-seq of 50 paired hepatocellular carcinoma", according to NCBI's *GEO*.

```
proj_info <- sra_projects[sra_projects$project == "SRP068976",]

rse_srpSRP068976 <- create_rse(proj_info)
```

```
2026-02-23 07:59:22.770516 downloading and reading the metadata.

2026-02-23 07:59:23.383727 caching file sra.sra.SRP068976.MD.gz.

2026-02-23 07:59:24.082421 caching file sra.recount_project.SRP068976.MD.gz.

2026-02-23 07:59:24.797881 caching file sra.recount_qc.SRP068976.MD.gz.

2026-02-23 07:59:25.411505 caching file sra.recount_seq_qc.SRP068976.MD.gz.
```

```
2026-02-23 07:59:26.127568 caching file sra.recount_pred.SRP068976.MD.gz.

2026-02-23 07:59:26.22906 downloading and reading the feature information.

2026-02-23 07:59:26.84782 caching file human.gene_sums.G026.gtf.gz.

2026-02-23 07:59:27.478516 downloading and reading the counts: 99 samples across 63856 featu

2026-02-23 07:59:28.179246 caching file sra.gene_sums.SRP068976.G026.gz.

2026-02-23 07:59:29.601117 constructing the RangedSummarizedExperiment (rse) object.
```

Exploration

Metadata

This are the general characteristics of the resulting RSE object:

```
rse_srpSRP068976
```

```
class: RangedSummarizedExperiment
dim: 63856 99
metadata(8): time_created recount3_version ... annotation recount3_url
assays(1): raw_counts
rownames(63856): ENSG00000278704.1 ENSG00000277400.1 ...
  ENSG00000182484.15_PAR_Y ENSG00000227159.8_PAR_Y
rowData names(10): source type ... havana_gene tag
colnames(99): SRR3182261 SRR3129836 ... SRR3129932 SRR3129935
colData names(175): rail_id external_id ...
  recount_pred.curated.cell_line BigWigURL
```

Format

Functions `compute_read_counts()` and `expand_sra_attributes` are used to get counts per read from the project and an insight of their different atributes (which is one metadata field)

```
# Get counts per read
assay(rse_srpSRP068976, "counts") <- compute_read_counts(rse_srpSRP068976)

# Get insight of the different attributes
rse_srpSRP068976 <- expand_sra_attributes(rse_srpSRP068976)
colData(rse_srpSRP068976)[, grepl("^sra_attribute", colnames(colData(rse_srpSRP068976)))]
```

```
DataFrame with 99 rows and 3 columns
           sra_attribute.diagnosis sra_attribute.source_name
                       <character>               <character>
SRR3182261                  Normal                     Liver
SRR3129836                  Normal                     Liver
SRR3129837                  Normal                     Liver
SRR3129838                  Normal                     Liver
SRR3129839                  Normal                     Liver
...                            ...                       ...
SRR3129927                   Tumor                     Liver
SRR3129929                   Tumor                     Liver
SRR3129931                   Tumor                     Liver
SRR3129932                   Tumor                     Liver
SRR3129935                   Tumor                     Liver
           sra_attribute.tissue
                     <character>
SRR3182261                liver
SRR3129836                liver
SRR3129837                liver
SRR3129838                liver
SRR3129839                liver
...                         ...
SRR3129927                liver
SRR3129929                liver
SRR3129931                liver
SRR3129932                liver
SRR3129935                liver
```

`sra_attribute.diagnosis` has two different values: "Normal" and "Tumor".

```
table(colData(rse_srpSRP068976)$sra_attribute.diagnosis)
```

```
Normal  Tumor
    50     49
```

Among with this variable being just ideal to separate the samples, there is enough data in both groups to succesfully perform a differential expression analysis.

Something worth to mention is that, looking at the project on *GEO*, we can see that the samples are paired, which means that we have both a sample of normal tissue and a sample of tumor tissue for each patient.

```
head(colData(rse_srpSRP068976)$sra.sample_title, 20)
```

```
 [1] "Patient 49 Normal" "Patient 1 Normal"  "Patient 2 Normal"
 [4] "Patient 3 Normal"  "Patient 4 Normal"  "Patient 5 Normal"
 [7] "Patient 6 Normal"  "Patient 7 Normal"  "Patient 10 Normal"
[10] "Patient 8 Normal"  "Patient 9 Normal"  "Patient 11 Normal"
[13] "Patient 12 Normal" "Patient 14 Normal" "Patient 13 Normal"
[16] "Patient 15 Normal" "Patient 16 Normal" "Patient 17 Normal"
[19] "Patient 18 Normal" "Patient 19 Normal"
```

Preprocessing

Clustering samples into their ID groups

Before the analysis itself, I'm going to isolate the patient ID from the `sra.sample_title` attribute:

```
rse_srpSRP068976$patient <- factor(
  sub("Patient ([0-9]+) .*", "\\1",
      colData(rse_srpSRP068976)$sra.sample_title)
)

table(rse_srpSRP068976$patient)
```

```
 1 10 11 12 13 14 15 16 17 18 19  2 20 21 22 23 24 25 26 27 28 29  3 30 31 32
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  1  2  2  2  2  2  2  2
33 34 35 36 37 38 39  4 40 41 42 43 44 45 46 47 48 49  5 50  6  7  8  9
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
```

And, indeed, there are 50 patients with 2 samples each (except for the 26th). Until now is confirmed that the data is paired.

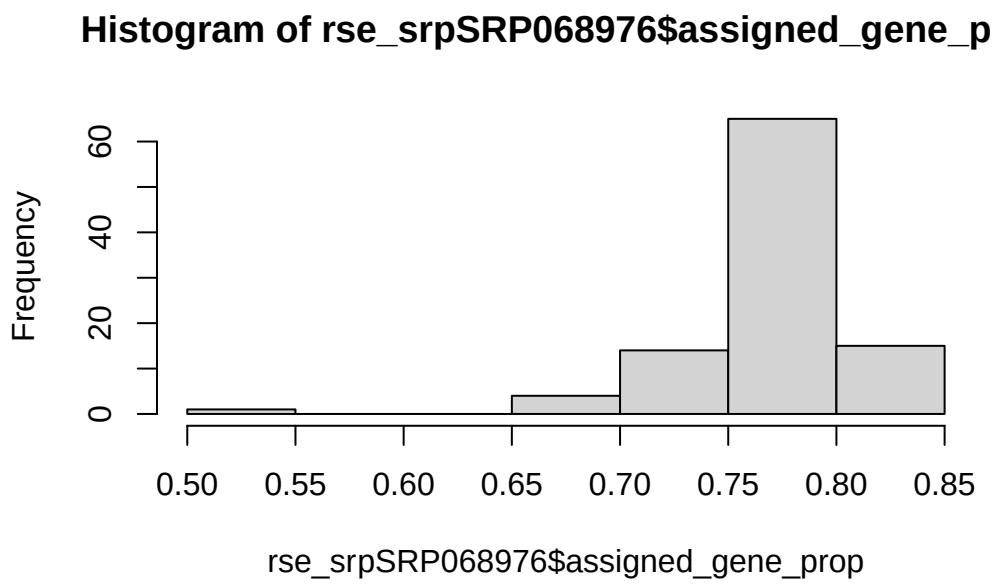
Basic QC

Basic *QC* is performed by looking at the proportion of assigned genes (genes that have at least one read assigned to them) over the total number of genes, that is, how many genes are useful and in which proportion.

```
rse_srpSRP068976$assigned_gene_prop <-
  rse_srpSRP068976$recount_qc.gene_fc_count_all.assigned /
  rse_srpSRP068976$recount_qc.gene_fc_count_all.total

summary(rse_srpSRP068976$assigned_gene_prop)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.5237  0.7621  0.7777  0.7705  0.7951  0.8169
```

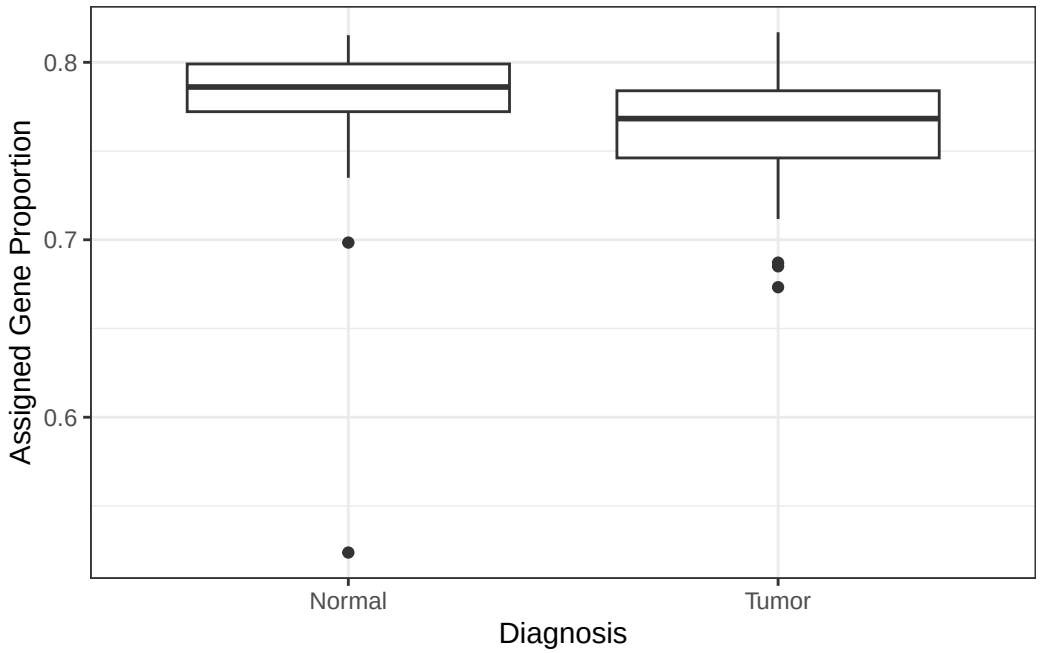
```
hist(rse_srpSRP068976$assigned_gene_prop)
```



As shown, the proportion of assigned genes is quite good, with a median of 77.77% and a mean of 77.05%. This means that most of the reads are being assigned to genes.

Box-plots are useful to determine whether the quality of the samples is independent of the diagnosis variable or not:

```
ggplot(as.data.frame(colData(rse_srpSRP068976)),
       aes(x = sra_attribute.diagnosis,
           y = assigned_gene_prop)) +
  geom_boxplot() +
  theme_bw() +
  xlab("Diagnosis") +
  ylab("Assigned Gene Proportion")
```



The quality of the samples does not seem to be confounded with the diagnosis variable.

Drop lowly expressed genes

```
# Save the original object to prevent recomputing
rse_srpSRP068976_unfiltered <- rse_srpSRP068976

# Filter out lowly expressed genes
gene_means <- rowMeans(assay(rse_srpSRP068976, "counts"))
summary(gene_means)
```

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.000e+00 1.000e-02 8.000e-01 3.391e+02 5.293e+01 1.269e+06
```

```
rse_srpSRP068976 <- rse_srpSRP068976[gene_means > 0.1, ]
```

`model.matrix()` is used to create the design matrix for the analysis, which is based on the model used in class but includes both the patient ID and the diagnosis as variables.

```
mod <- model.matrix(
  ~ patient + sra_attribute.diagnosis,
  data = colData(rse_srpSRP068976)
)
colnames(mod)
```

```
 [1] "(Intercept)"                 "patient10"
 [3] "patient11"                   "patient12"
 [5] "patient13"                   "patient14"
 [7] "patient15"                   "patient16"
 [9] "patient17"                   "patient18"
[11] "patient19"                   "patient2"
[13] "patient20"                   "patient21"
[15] "patient22"                   "patient23"
[17] "patient24"                   "patient25"
[19] "patient26"                   "patient27"
[21] "patient28"                   "patient29"
[23] "patient3"                    "patient30"
[25] "patient31"                   "patient32"
[27] "patient33"                   "patient34"
[29] "patient35"                   "patient36"
[31] "patient37"                   "patient38"
[33] "patient39"                   "patient4"
[35] "patient40"                   "patient41"
[37] "patient42"                   "patient43"
[39] "patient44"                   "patient45"
[41] "patient46"                   "patient47"
[43] "patient48"                   "patient49"
[45] "patient5"                    "patient50"
[47] "patient6"                    "patient7"
[49] "patient8"                    "patient9"
[51] "sra_attribute.diagnosisTumor"
```

There are 49 patient coefficients (one for each patient except the reference) and one coefficient for the diagnosis variable, which is the most informative one…

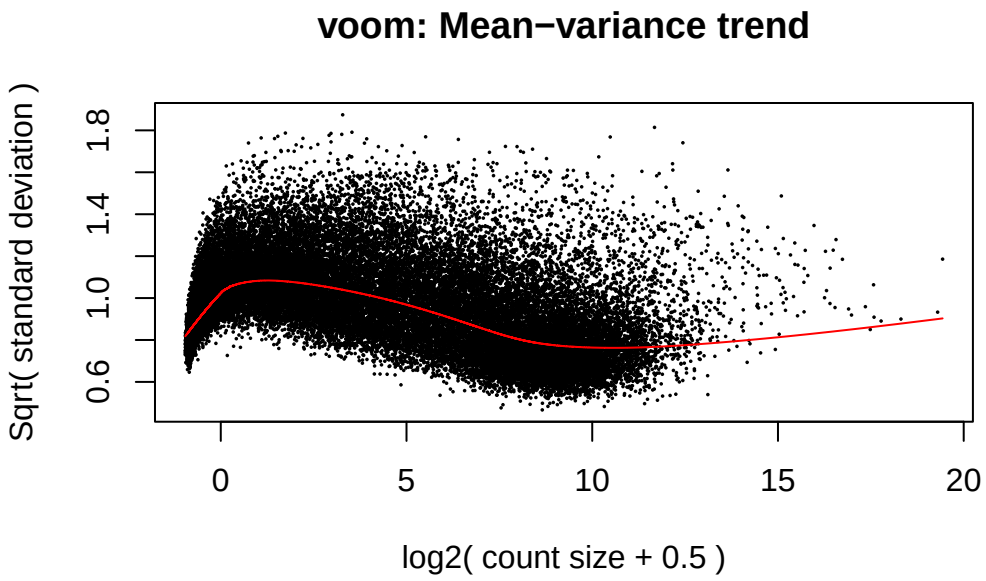
Normalization

At this point, the data is ready for normalization and differential expression analysis. The `DGEList()` function from the *edgeR* package is used to create a DGEList object from the counts and the gene information, and then `calcNormFactors()` to compute the normalization factors.

```
dge <- DGEList(
  counts = assay(rse_srpSRP068976, "counts"),
  genes = rowData(rse_srpSRP068976)
)
dge <- calcNormFactors(dge)
```

`voom()` function from the *limma* package estimates the mean-variance relationship and produces a transformed expression matrix that can be used for linear modeling.

```
vGene <- voom(dge, mod, plot = TRUE)
```




```
eb_results <- eBayes(lmFit(vGene))

de_results <- topTable(
  eb_results,
```

```
  coef = "sra_attribute.diagnosisTumor",
  number = Inf
)
dim(de_results); head(de_results)
```

```
[1] 41003    16
```

```
                  source type bp_length phase            gene_id
ENSG00000134690.10 HAVANA gene      2469    NA ENSG00000134690.10
ENSG00000100526.19 HAVANA gene      1836    NA ENSG00000100526.19
ENSG00000237649.7  HAVANA gene      3345    NA  ENSG00000237649.7
ENSG00000112984.11 HAVANA gene      3958    NA ENSG00000112984.11
ENSG00000143228.12 HAVANA gene      3362    NA ENSG00000143228.12
ENSG00000164611.12 HAVANA gene      2013    NA ENSG00000164611.12
                       gene_type gene_name level          havana_gene  tag
ENSG00000134690.10 protein_coding     CDCA8     2 OTTHUMG00000004320.1 <NA>
ENSG00000100526.19 protein_coding     CDKN3     1 OTTHUMG00000140302.3 <NA>
ENSG00000237649.7  protein_coding     KIFC1     2 OTTHUMG00000031209.4 <NA>
ENSG00000112984.11 protein_coding    KIF20A     2 OTTHUMG00000129195.3 <NA>
ENSG00000143228.12 protein_coding      NUF2     1 OTTHUMG00000034275.3 <NA>
ENSG00000164611.12 protein_coding     PTTG1     2 OTTHUMG00000130328.5 <NA>
                      logFC   AveExpr        t      P.Value    adj.P.Val
ENSG00000134690.10 4.043978 1.2304986 28.16270 8.453577e-34 3.466220e-29
ENSG00000100526.19 4.625529 0.7976569 27.26575 4.287480e-33 8.789978e-29
ENSG00000237649.7  4.132802 2.0841661 25.92257 5.329825e-32 7.284627e-28
ENSG00000112984.11 4.044777 1.5456917 25.32955 1.680714e-31 1.722858e-27
ENSG00000143228.12 4.589461 0.8598324 25.01408 3.125257e-31 2.562898e-27
ENSG00000164611.12 4.181153 1.7461350 24.76174 5.157552e-31 3.003476e-27
                         B
ENSG00000134690.10 66.37806
ENSG00000100526.19 64.76176
ENSG00000237649.7  62.50351
ENSG00000112984.11 61.33286
ENSG00000143228.12 60.65325
ENSG00000164611.12 60.26603
```

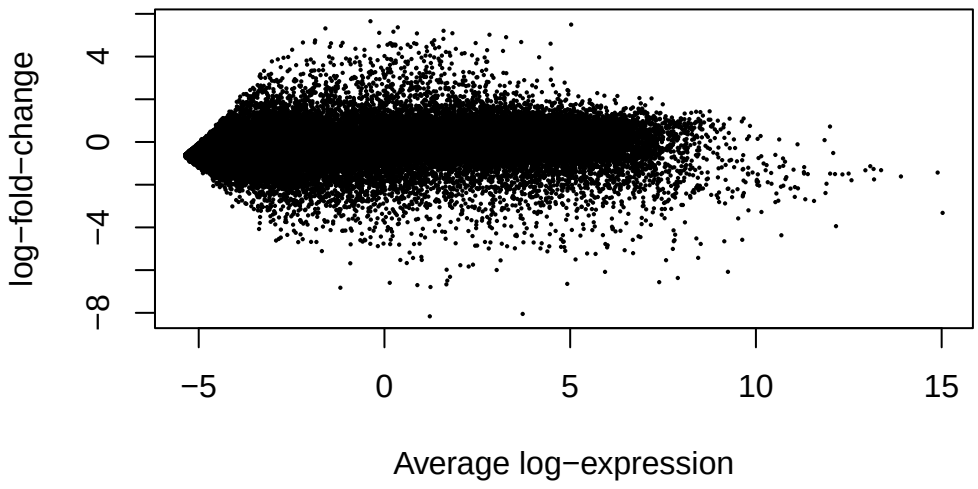
Now we have the results of the differential expression analysis, with the log fold change, average expression, t-statistic, p-value and adjusted p-value for each gene. We can filter the results to find the differentially expressed genes based on a threshold for the adjusted p-value.

```
table(de_results$adj.P.Val < 0.05)
```

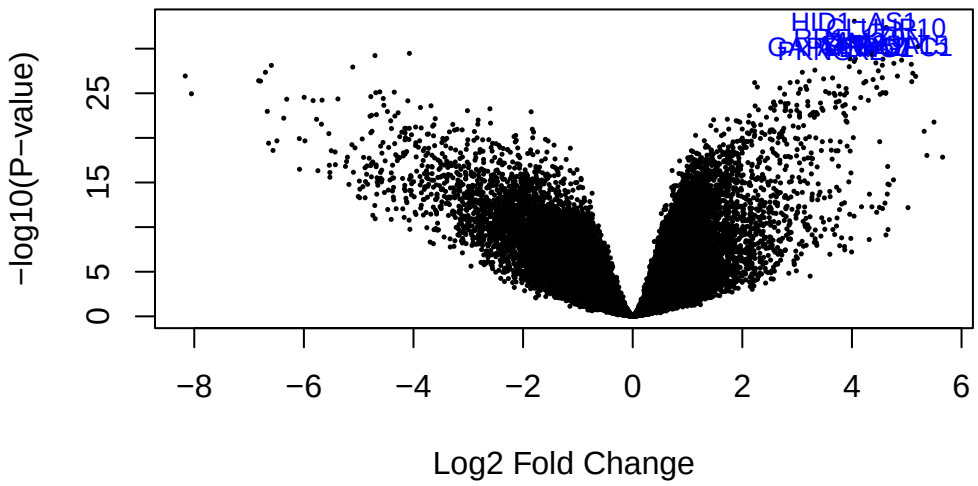
```
FALSE  TRUE
13008 27995
```

And, as well as in class, MA and a volcano plots can be maked to visualize the results:

```
plotMA(eb_results, coef = "sra_attribute.diagnosisTumor")
```

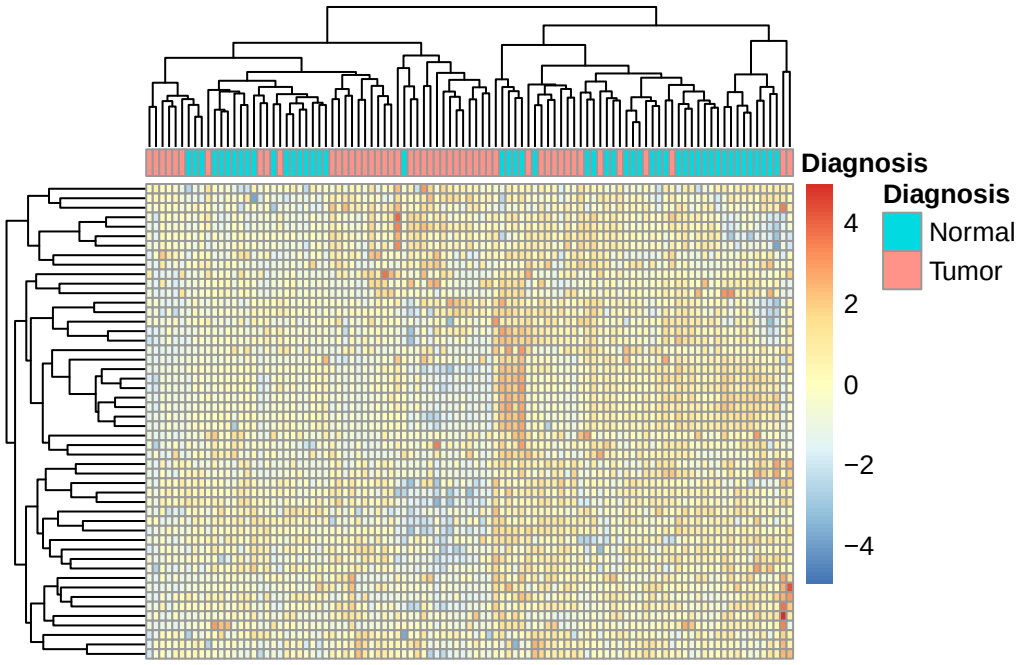



```
volcanoplot(eb_results,
            coef = "sra_attribute.diagnosisTumor",
            highlight = 10,
            names = de_results$gene_name)
```



Lastly, this are the top 50 differentially expressed genes:

```
exprs_heatmap <- vGene$E[rank(de_results$adj.P.Val) <= 50, ]

df <- as.data.frame(
  colData(rse_srpSRP068976)[, "sra_attribute.diagnosis", drop = FALSE]
)

colnames(df) <- "Diagnosis"

pheatmap::pheatmap(
  exprs_heatmap,
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  show_rownames = FALSE,
  show_colnames = FALSE,
  annotation_col = df,
  scale = "row"
)
```

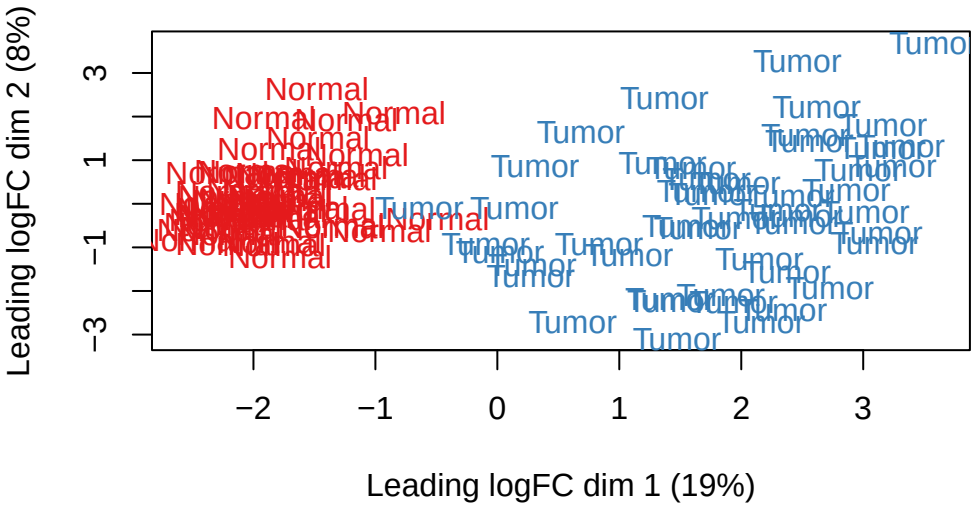


Tumor and normal samples separate into distinct clusters, indicating that the strongest differentially expressed genes are sufficient to discriminate between the two biological conditions.

```
# Extraemos el grupo biológico
col.group <- rse_srpSRP068976$sra_attribute.diagnosis

# Convertimos a factor (por seguridad)
col.group <- factor(col.group)

# Asignamos colores
levels(col.group) <- brewer.pal(nlevels(col.group), "Set1")
```

```
Warning in brewer.pal(nlevels(col.group), "Set1"): minimal value for n is 3, returning reques
```

```
# Convertimos a vector de colores
col.group <- as.character(col.group)

# MDS
plotMDS(vGene$E,
        labels = rse_srpSRP068976$sra_attribute.diagnosis,
        col = col.group)
```



There is a clear separation between the two conditions in the MDS plot, which indicates that the global transcriptional variation is strongly associated with the diagnosis variable.

Resultados

Results

This *differential expression analysis* identified a large number of genes significantly altered between tumor and normal liver tissue (FDR < 0.05). The MA and volcano plots shows that many genes exhibit both strong fold changes and high statistical significance, indicating some sort of transcriptional dysregulation in *hepatocellular carcinoma*. Following, the heatmap of the top 50 differentially expressed genes shows a clustering by diagnosis, suggesting tumor and normal samples have different expression *forms*. Moreover, when looking to the MDS plot, the way the two conditions are clustered reinforces that diagnosis is fundamental on global transcriptional variation.

References