

RNA-seq Analysis of Skin Cancer

Lidia Montes, María José Zaruma, Wenhui Zhou

Contents

| | |
|--|----------|
| 1 Abstract | 2 |
| 2 Methods | 2 |
| 2.1 Packages and tools used | 2 |
| 2.2 Data description | 3 |
| 2.3 Statistical test | 3 |
| 3 Results and Figures | 4 |
| 4 Discussion | 5 |
| 5 References | 5 |
| Appendix A: RMarkdown Script – Skin Cancer RNA-seq Analysis | 6 |

1 Abstract

Skin cancer is the most common worldwide. It results from a combination of environmental and genetic factors and is divided into two types. The first group includes those derived from the transformation of melanocytes, known as melanoma. The second group arises from epidermal-derived cells and is classified as “non-melanoma.”

The most common cause of this cancer is prolonged exposure to ultraviolet light, but there are also other risk factors. Phenotypically, individuals with light-colored eyes and skin, red hair, and freckles are more susceptible. Other environmental factors include immunosuppression and viral infections. Genetically, although most cases are sporadic, mutations in genes like *MDM2*, for instance, predispose women to develop carcinoma at an early age. Craythorne and Al-Niami (2017)

Basal cell carcinoma is the most common skin neoplasm and derives from pluripotent cells within the follicular epithelium that have developed a mutation in *p53*. However, it can also result from a mutation in *PTCH1*, leading to aberrant activation of the *sonic hedgehog* signaling cascade. On the other hand, squamous cell carcinoma arises from alterations in squamous keratinocytes. This malignant tumor has low metastatic potential, and germline mutations in *p53* predispose to its development. These two carcinomas fall under the “non-melanoma” category. Finally, melanomas account for only 2% of malignant skin tumors but are responsible for the majority of deaths, and variants in genes like *CDKN2A* have been associated with them. Craythorne and Al-Niami (2017), Linares et al. (2015)

A transcriptomic tool that allows us to study this cancer is RNA-sequencing, which helps perform differential gene expression analysis and determine quantitative changes in expression levels between tumor samples at different stages of skin cancer. Stark et al. (2019)

2 Methods

2.1 Packages and tools used

The program used was [RStudio](#) along with the following packages and tools:

- ggplot2
- ggrepel
- BiocManager

From the BiocManager package, the following tools were also installed: SummarizedExperiment, DESeq2, org.Hs.eg.db, biomaRt, edgeR, tweedEseq, GOstats, tweedEseqCountData, annotate, biomaRt

2.2 Data description

The file **rse_gene_skin.Rdata** contains the experiment data in a matrix-like format, where rows represent ranges of interest and columns contain sample data, collected in **data.frames**.

To perform the analysis, the following files must be retrieved from the **rse_gene** object:

- Variable **gdc_cases.diagnoses.tumor_stage**: contains information on tumor stage.
- *Counts*: contains information on *read counts*, and can be retrieved using the **assay()** function.
- Phenotypic data can be retrieved using the **colData()** function.
- Gene information can be retrieved using the **rowData()** function.

2.3 Statistical test

Differential expression analysis was performed using the R package **DESeq2**, which is based on a negative binomial distribution. This package uses unnormalized data as it includes its own integrated normalization method.

The analysis compares early-stage and late-stage tumors to identify the most differentially expressed genes. This analysis allows the identification of significantly up- or downregulated genes in late-stage tumors using the following filters:

- Non-significant genes with p-value >0.001 are removed.
- Genes with a *log2foldchange* of 10 are retained (i.e., values between -10 and 10, as shown in Figure 1). Using the **plotMA** function, the *log2foldchange* is visualized across the mean of normalized *counts*.

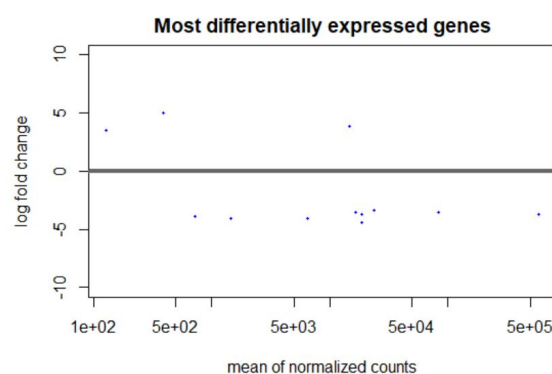


Figure 1 – Plot of the most differentially expressed genes showing their *log2foldchange* values against the mean of normalized *counts*.

3 Results and Figures

Once the RNA-seq analysis was performed, the data were visualized in a volcano plot (Figure 2). This plot displays the negative logarithm of the p-value on the y-axis, placing highly significant points at the top. The x-axis represents the logarithm of the fold change between the two conditions. This allows a quick visual identification of genes with statistically significant high-magnitude changes, located far left or right.

A gene enrichment analysis based on the functional annotation of differentially expressed genes was carried out to interpret gene expression data and determine whether those genes are associated with specific biological processes or molecular functions (Figure 3). Results show that most genes are associated with developmental processes.

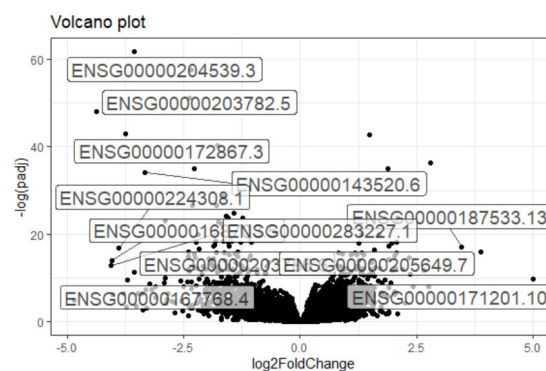


Figure 2 – RNA-seq results represented in a volcano plot. The lines indicate genes of interest with large fold changes (x-axis) and high statistical significance ($-\log_{10}$ of p-value, y-axis).

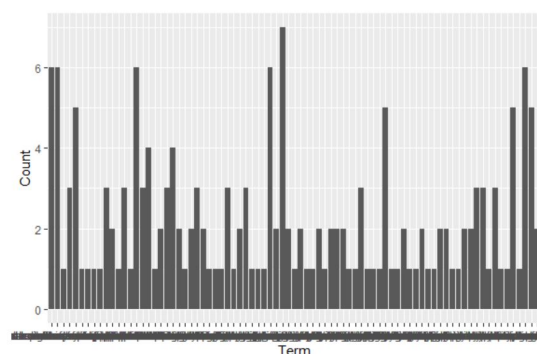


Figure 3 – GO results: x-axis shows GO terms (biological process or molecular function), y-axis shows count of significant genes.

4 Discussion

In this analysis, we performed RNA-seq to study differential gene expression between two skin cancer tumor groups. Once these genes were obtained, the enrichment analysis using *gene ontology* revealed that the genes were mainly associated with developmental processes. However, others, though fewer in number, were linked to relevant processes such as keratin activation, development and migration, epithelial differentiation, the immune system, and epidermal development. Since this is a cancer study, it is coherent that some genes are associated with developmental and immune processes. Additionally, as we are dealing with skin cancer, it is logical that genes involved in keratinocyte development, tissue differentiation, or epidermal development are implicated. Notably, no genes associated with melanocyte development were found, suggesting that the sample studied corresponds to a “non-melanoma” skin cancer.

In conclusion, the differentially expressed genes between *early* and *late* cancer are mainly involved in activation, differentiation, development, and processes related to epidermal cells.

5 References

- E. Craythorne and F. Al-Niami. Skin cancer. *Medicine*, 45(7):431–434, 2017. doi: 10.1016/j.mpmed.2017.04.003.
- M. A. Linares, A. Zakaria, and P. Nizran. Skin cancer. *Primary Care: Clinics in Office Practice*, 42(4):645–659, 2015. doi: 10.1016/j.pop.2015.07.006.
- R. Stark, M. Grzelak, and J. Hadfield. Rna sequencing: The teenage years. *Nature Reviews Genetics*, 20(11):631–656, 2019. doi: 10.1038/s41576-019-0150-2.

Appendix A: RMarkdown Script – Skin Cancer RNA-seq Analysis

This RMarkdown script is part of a group assignment for the course *Current Topics in Bioinformatics*. The instructors provided several datasets for analysis, and we chose to work with the skin cancer dataset. The base script was provided to us as a starting point. While the analysis presented in this document is our own work performed on this dataset, the original script is included here to ensure transparency and give proper credit to the original material. The preceding pages contain the report presenting the results and interpretation of the RNA-seq analysis performed on this dataset.

```

---
title: "**P4 | Solving real cases in genomics**<br><font color='#A8A8A8'
      size='5'>Finding differentially expressed genes in cancer</font>"
author: "Current Topics in Bioinformatics"
date: "1 December 2021"
output:
  html_document:
    theme: yeti
    css: https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.11.2/css/all
        .css
    self_contained: yes
---

```{r setup, include = FALSE, warning = FALSE}
knitr::opts_chunk$set(echo = TRUE, fig.align = "center", fig.width = 6, fig
 .height = 4)
#If the path directory is not the correct one. Please, uncomment the
 following line and
change "directorio" to the correct one.
#knitr::opts_knit$set(root.dir = "directorio")
library(SummarizedExperiment)
library(edgeR)
library(DESeq2)
library(tweeDEseq)
library(tweeDEseqCountData)
library(GOstats)
library(annotate)
library(org.Hs.eg.db)
library(biomaRt)
library(ggplot2)
library(ggrepel)
```

<style>
  @import url(https://fonts.googleapis.com/css?family=Fira+Sans:300,300i
    ,400,400i,500,500i,700,700i);
  @import url(https://cdn.rawgit.com/tonsky/FiraCode/1.204/distr/fira_code.
    css);
  @import url("https://use.fontawesome.com/releases/v5.10.1/css/all.css");
</style>

<div style="background-color: #86CBBB; 1px; height:3px " ></div>

```

1. Introduction

RNA-seq is a recent approach to carry out expression profiling using high-throughput sequencing (HTS) technologies. Pre-2008, microarrays were predominantly used, but because the sequencing costs have decreased, RNA-seq became the preferred option to simultaneously measure the expression of tens of thousands of genes for multiple samples.

In this tutorial we walk through a gene-level RNA-seq differential expression analysis using **Bioconductor** packages to find genes over- or under-expressed in cancer patients.

1.1 Experimental data

The **Cancer Genome Atlas** (<https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga>) (**TCGA**) is a collaboration between the National Cancer Institute (NCI) and the National Human Genome Research Institute (NHGRI) that has generated comprehensive, multi-dimensional maps of the key genomic changes in 33 types of cancer. The TCGA dataset, comprising more than two petabytes of genomic data, has been made publicly available, and this genomic information helps the cancer research community to improve the prevention, diagnosis, and treatment of cancer.

Recount (<https://jhubiostatistics.shinyapps.io/recount/>) is an online resource consisting of RNA-seq gene and exon counts for different studies, including TCGA data. From there, we can download the RNA-seq data of different cancer types that we will use in this practical:

| Cancer | Download |
|---------------|---|
| Bile duct | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_bile_duct.Rdata) |
| Eye | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_eye.Rdata) |
| Testis | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_testis.Rdata) |
| Pancreas | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_pancreas.Rdata) |
| Esophagus | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_esophagus.Rdata) |
| Adrenal gland | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_adrenal_gland.Rdata) |
| Liver | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_liver.Rdata) |
| Bladder | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_bladder.Rdata) |
| Stomach | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_stomach.Rdata) |
| Skin | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_skin.Rdata) |

| | |
|---------------|---|
| Bile duct | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_bile_duct.Rdata) |
| Eye | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_eye.Rdata) |
| Testis | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_testis.Rdata) |
| Pancreas | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_pancreas.Rdata) |
| Esophagus | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_esophagus.Rdata) |
| Adrenal gland | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_adrenal_gland.Rdata) |
| Liver | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_liver.Rdata) |
| Bladder | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_bladder.Rdata) |
| Stomach | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_stomach.Rdata) |
| Skin | [Link](http://duffel.rail.bio/recount/v2/TCGA/rse_gene_skin.Rdata) |

The practical consist on a worked example using data from **pleural cancer**. Each group must select a cancer type from the table to perform their own differential expression analysis.

P4 Learning outcomes

- * Data manipulation (P1)
- * Descriptive analysis of the data (P1)
- * Methods to normalize RNA-seq data
- * Perform statistical tests to find which genes are deferentially expressed **in** different human cancers
- * Enrichment analysis
- * Visualize the results (P2)

1.2 Practicals organization

In this practical, we are going to use the **RStudio**(<https://rstudio.com/>) integrated development environment (IDE) **for** R. R is a programming language **for** statistical computing and graphics.

You will see different icons through the document, the meaning of which is:

 <i class="fas fa-info-circle"></i>: additional or useful information

 <i class="fas fa-search"></i>: a worked example

 <i class="fa fa-cogs"></i>: a practical exercise

 <i class="fas fa-comment-dots"></i>: a space to answer the exercise

 <i class="fa fa-key"></i>: a hint to solve an exercise

 <i class="fa fa-rocket"></i>: a more challenging exercise

<div style="background-color: #86CBBB; 1px; height:3px " ></div>

2. Tools installation

Follow `P2.Rmd` instructions **if** you need to install **R**(<https://cran.r-project.org/>) and/or **RStudio**(<https://rstudio.com/products/rstudio/download/#download>) **for** either Windows or Linux.

2.1 Required R packages

Bioconductor has many packages supporting analysis of high-throughput sequence data, including RNA-seq. The packages that we will use **in** this tutorial include core packages maintained by the Bioconductor core team **for** importing and processing raw sequencing data and loading gene annotations.

```
```${r packages, eval = FALSE}
install.packages("ggplot2")
install.packages("ggrepel")
install.packages("BiocManager")
```



```

BiocManager::install(c("SummarizedExperiment", "DESeq2", "org.Hs.eg.db", "
 biomaRt", "edgeR", "tweeDEseq", "GOstats", "tweeDEseqCountData", "
 annotate", "biomaRt"))

<div style="background-color: #86CBBB; 1px; height:3px " ></div>

<i class = "fa fa-search"></i>Conducting an RNA-seq analysis

With this worked example, we are going to illustrate how to perform an RNA-
seq expression analysis on RNA-seq data from patients diagnosed with
mesothelioma (pleural cancer). The goal is to compare the
transcriptomic profile of patients in an early tumoral stage with ones
in an advanced stage of this disease. The objective is to find
differential expressed in this type of cancer.

The data is available in a Ranged Summarized Experiment (RSE) format.
It is a matrix-like container where rows represent ranges of interest
and columns represent samples (with sample data summarized as a `data.
frame`).

1. Prepare data

**<i class="fa fa-cogs"></i> Create a working directory named `Project4`
and create a folder named `data` and inside, `pleuralCancer` folder. **

```{r, eval = FALSE}
# bash code to create directories
mkdir Project4 | cd Project4 | mkdir data | cd data | mkdir pleuralCancer |
  cd pleuralCancer
---

**<i class="fa fa-cogs"></i> Navigate to the `pleuralCancer` folder and
download the [RSE data](http://duffel.rail.bio/recount/v2/TCGA/
rse_gene_pleura.Rdata) for pleura cancer . **

<i class="fa fa-key"></i> This time you can use the `wget` function in the
Linux terminal to download the file!

```{r, eval = FALSE}
bash code to download and uncompress the data
wget duffel.rail.bio/recount/v2/TCGA/rse_gene_pleura.Rdata

**<i class="fa fa-cogs"></i> RSE data can be loaded into `R` with the `load
()` function.**

<i class="fa fa-key"></i> Remember to change the working directory to
properly load the data if it's necessary:

```{r read-RNA-seq-data}
# getwd() to know the current working directory
# setwd("~/Descargas/Project4") will change the current directory to the

```

```

Project4 folder inside the Descargas folder

# Read RSE data of the cancer experiment
load(file = "data/pleuralCancer/rse_gene_pleura.Rdata")
```

An object called `rse_gene` will be in your R environment.

Exploring the data using the `dim()` function, we see that there are `r dim(rse_gene)[1]` genes (number of rows) analyzed in a cohort of `r dim(rse_gene)[2]` patients with cancer (number of columns).

The variable `gdc_cases.diagnoses.tumor_stage` which is inside the `rse_gene` object contains information of the tumoral stage (i.e.: `r sort(unique(rse_gene$gdc_cases.diagnoses.tumor_stage))`). We can use this information to create a new variable in the `rse_gene` object called `GROUP` distinguishing "_early_" and "_late_" tumours. Early tumours are those in stages i and ii, while late tumours those in stages iii and iv.

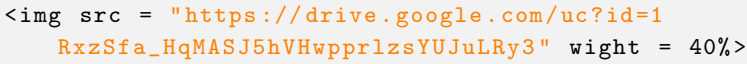
The following code will add an additional column `GROUP` to the meta-data to organize the cancer stages:

```{r create-group}
stage <- rse_gene$gdc_cases.diagnoses.tumor_stage

# id list of early tumours
ids.early <-grep(paste("stage i$", "stage ia$", "stage ib$", "stage ic$", "stage ii$", "stage iia$", "stage iib$", "stage iic$", sep="|"), stage)

# id list of late tumours
ids.late <-grep(paste("stage iii$", "stage iiia$", "stage iiib$", "stage iiic$", "stage iv$", "stage iva$", "stage ivb$", "stage ivc$", sep="|"), stage)

# create an empty column named GROUP
colData(rse_gene)$GROUP <-rep(NA, ncol(rse_gene))
# add early for those patients with tumours at stages i-ii
colData(rse_gene)$GROUP[ids.early] <- "early"
# add late for those patients with tumours at stages iii-iv
colData(rse_gene)$GROUP[ids.late] <- "late"
```

<center>


</center>

<i class="fa fa-question-circle"></i> **Can you reproduce the previous figure using the following `data.frame()`? **

```{r data-frame-fig}

```

```
# create dataframe
dataGroups <- data.frame(Stage = rse_gene$gdc_cases.diagnoses.tumor_stage,
  Group = rse_gene$GROUP)

# ggplot2 plot
ggplot(data = dataGroups, mapping = aes(x= Stage, fill = Group)) + geom_bar
  (position = 'dodge') + labs (y = 'Number of patients') + theme_bw()

```
```

It is important to remove all patients whose stage of the cancer was not recorded. We can check `if` it's necessary `in` the pleura cancer dataset using the ``table()'` `function`:

```
```{r check-na}
# Check if the status is NA
naData <- is.na(rse_gene$GROUP)
table(naData)
```
```

As we can see, all patients have information because all the results are ``FALSE'` when we check `if` something `"_is na?_"`. But `if` we are dealing with some ``TRUE'` results (i.e., there's no data `for` some patients), we can remove it as:

```
```{r remove-na}
# After removing
dim(rse_gene)

# Remove NA
rse_gene <- rse_gene[, !naData]
dim(rse_gene)
```
```

We can summarize individuals `in` each category as `"early"` and `"late"` using the ``table()'` `function`. We can see that there are more than two times patients `in` the late stage than `in` the early stage.

```
```{r summary}
# summary of the patients
table(rse_gene$GROUP)
```
```

After exploring a bit the data, we can now get the `**read counts**` data. It can be retrieved using the ``assay()'` `function`.

```
```{r save-counts}
# save the count data
counts <- assay(rse_gene, "counts")
counts[1:5, 1:2]
```
```

The data related to the `**phenotype**` of the patients can also be retrieved

```

using the `colData()` function:

```{r save-pheno}
# save the phenotype data
phenotype <- colData(rse_gene)
phenotype[1:5, 1:2]
```

We need to check that the same individuals are found both in the `counts`
dataset and in the `phenotype` dataset.

```{r check-ind}
# check if the same individuals are found in the datasets
identical(colnames(counts), rownames(phenotype))
```

Because the result is `TRUE`, it means that we have the same individuals in
both datasets. If the result were `FALSE`, we would need to only keep
those individuals in common in both datasets. For that, we can use the
`intersect` function.

```{r keep-same-individuals}
# save a vector with the id of the individuals in common in both datasets
individualsCommon <- intersect(colnames(counts), rownames(phenotype))
# filter the count dataset to keep only the individuals in the vector
counts <- counts[, individualsCommon]
# filter the phenotype dataset to keep only the individuals in the vector
phenotype <- phenotype[individualsCommon,]
```

In this case, we still have the `r length(individualsCommon)` individuals.

Finally, the gene information can be retrieved using the `rowData()`
function:

```{r annotation}
# save the annotation data
annotation <- rowData(rse_gene)
```

We have information for a total of `r length(annotation$gene_id)` genes.

<i class="fa fa-question-circle"></i> Can you explore which information
is available for each gene?

<div style="background-color:#F0F0F0">
<i class="fas fa-comment-dots"></i> Answer: id del gen,
longitud, s mbolo (nombre)
</div>

<i class="fa fa-question-circle"></i> Write a short summary of the

```

```

information you have available (e.g., total number of individuals,
filtered individuals, individuals by stage, number of genes, ...)**

<div style="background-color:#F0F0F0">
 <i class="fas fa-comment-dots"></i> Answer: tenemos
informaci n de cada persona sobre su fenotipo y genotipo. Tambi n
sobre el estad o del cancer, la cantidad de genes que tenemos y sus
respectivas caracter sicas, y se han filtrado los individuos en los
que no se ha determinado su estad o de cancer o les faltan otros datos
.

</div>

2. Normalization

Why is it necessary to **normalize** RNA-seq data?

1. The number of counts is related to sequencing depth

_Accounting for sequencing depth is necessary for comparison of gene
expression between samples. In the example below, each gene appears to
have doubled in expression in Sample A relative to Sample B, however
this is a consequence of Sample A having double the sequencing depth._

<center>
<img src = "https://hbctraining.github.io/DGE_workshop/img/
normalization_methods_depth.png" width = 60%>
</center>

2. The number of counts is related to transcript length

_Accounting for gene length is necessary for comparing expression between
different genes within the same sample. In the example, Gene X and Gene
Y have similar levels of expression, but the number of reads mapped to
Gene X would be many more than the number mapped to Gene Y because
Gene X is longer._

<center>
<img src = "https://hbctraining.github.io/DGE_workshop/img/
normalization_methods_length.png" width = 30%>
</center>

3. The number of counts is proportional to the mRNA expression level

_A few highly diferentially expressed genes between samples, differences in
the number of genes expressed between samples, or presence of
contamination can skew some types of normalization methods. Accounting
for RNA composition is recommended for accurate comparison of
expression between samples, and is particularly important when
performing differential expression analyses._

<center>
<img src = "https://hbctraining.github.io/DGE_workshop/img/

```

```

normalization_methods_composition.png" width = 60%>
</center>

Information retrieved from [Introduction to DGE](https://hbctraining.github
.io/DGE_workshop/lessons/02_DGE_count_normalization.html).

_The aim of normalization is to remove systematic technical effects that
 occur in the data to ensure that technical bias has minimal impact on
 the results_ ([Robinson and Oshlack, 2010](https://genomebiology.
 biomedcentral.com/articles/10.1186/gb-2010-11-3-r25)).

There are different methods to normalize the counts (for a review see [
 Evans et al., 2017](https://academic.oup.com/bib/article
 /19/5/776/3056951)). We are going to focus in **two types**:

1. **RPKM** ([Mortazavi et al., 2008](https://www.nature.com/articles/nmeth
 .1226)) -- Reads Per Kilobase Million. This method corrects for the
 sequencing depth and the gene length. It's a non-sophisticated
 normalization method. Counts are divided by the transcript length (kb)
 times the total number of millions of mapped reads:

$$RPKM = \frac{\frac{\text{number of reads in region}}{\text{region length} \times 10^3}}{\text{total reads} \times 10^6}$$

2. **TMM** ([Robinson and Oshlack, 2010](https://genomebiology.
 biomedcentral.com/articles/10.1186/gb-2010-11-3-r25)) -- Trimmed Mean
 of M values. This method was developed to address this issue: _The
 proportion of reads attributed to a given gene in a library depends on
 the expression properties of the whole sample rather than just the
 expression level of that gene._ Therefore, the method accounts for
 sequencing depth, RNA composition, and gene length.

<div style = "background-color:#FFDAA1">
 <i class="fa fa-info-circle"></i> Both methods allows comparisons between
 genes within a sample, but only TMM is recommended for between
 sample comparisons and differential expression analyses.
</div>

The **MA-plots** are used to check whether normalization is needed or not.
 There is a function in the `edgeR` package designed to create such a
 plot, the `maPlot()` function:


```

`{r maPlot-row}
maPlot(counts[,1],
       counts[,2],
       pch=19,
       cex=.5,
       ylim=c(-8,8),
       allCol="darkgray",
       lowess=TRUE,
       xlab=expression(A == log[2] (sqrt(S1/N %.% S2/N))),
       ylab=expression(M == log[2] (S1/N)-log[2] (S2/N)))
grid(col="black")

```


```

```

title("Raw data")
```

This plot represents the log-fold change (M-values, i.e. the log of the ratio of level counts for each gene between two samples) against the log-average (A-values, i.e. the average level counts for each gene across the two samples). From a MA-plot one can see if normalization is needed or not.

One expects that the vast majority of genes are not differentially expressed between individual, thus having a symmetrical distribution on the plot with most of the genes at the 0 ( $y = 0$ ). A lowess fit (red line) is plotted underlying a possible trend in the bias related to the mean expression.

#### 2.1 RPKM normalization

We can normalize read counts using the RPKM normalization simply applying the previous formula.

```{r rpkm}
length of the gene
geneLength <- annotation$bp_length

RPKM normalization method
counts.rpkm <- t(t(counts/geneLength*1000)/colSums(counts)*1e6)

counts.rpkm[1:5,1:2]
```

```{r rpkm-visualization}
maPlot(counts.rpkm[,1],
 counts.rpkm[,2],
 pch=19,
 cex=.5,
 ylim=c(-8,8),
 allCol="darkgray",
 lowess=TRUE,
 xlab=expression(A==log[2] (sqrt(S1/N%.%S2/N))),
 ylab=expression(M==log[2] (S1/N)-log[2] (S2/N)))
grid(col="black")
title("RPKM")
```

#### 2.2 TMM normalization

TMM normalization method is implemented in the twedEseq Bioconductor package, inside the normalizeCounts() function.

```{r tmm}
TMM normalization method
counts.tmm <- normalizeCounts(counts, method = "TMM")

```

```

counts.tmm[1:5,1:2]
```

```{r tmm-visualization}
maPlot(counts.tmm[,1],
 counts.tmm[,2],
 pch=19, cex=.5,
 ylim=c(-8,8),
 allCol="darkgray",
 lowess=TRUE,
 xlab=expression(A==log[2] (sqrt(S1/N%.%S2/N))),
 ylab=expression(M==log[2] (S1/N)-log[2] (S2/N)))
grid(col="black")
title("TMM")
```

**Discuss with a colleague the following: **

- **Do you observe any improvement when comparing the Raw data against the RPKM normalization?**
- **And when comparing the Raw data and the TMM method?**
- **Which is the best normalization method in this case?**

<center>


```


normalization method.

This `function` requires the ``SummarizedExperiment`` object and the design, which `in` this case is given `in` the variable ``GROUP`` because we want to compare early vs. late tumor stages.

```
```{r diffgen}
stage of each patient
pheno.stage <- subset(phenotype, select=GROUP)

recreate the counts in a new matrix
counts.adj <- matrix((as.vector(as.integer(counts))), nrow=nrow(counts),
 ncol=ncol(counts))

rownames(counts.adj) = rownames(counts)
colnames(counts.adj) = colnames(counts)

check information
identical(colnames(counts.adj), rownames(pheno.stage))

transform the group variable to factor
pheno.stage$GROUP <- as.factor(pheno.stage$GROUP)

create the DESeqDataSet input
DEs <- DESeqDataSetFromMatrix(countData = counts.adj,
 colData = pheno.stage,
 design = ~ GROUP)

differential expression analysis
dds <- DESeq(DEs)

results extracts a result table from a DESeq analysis
res <- results(dds, pAdjustMethod = "fdr")

head(res)
```
```

The ``plotMA`` `function` allows the graphical representation of the \log_2 fold-change over the mean of normalized counts `for` all the samples `in` the ``DESeqDataSet``. Points will be colored red `if` the adjusted p-value is less than 0.1. Points which fall out of the window are plotted as open triangles pointing either up or down.

```
```{r visualization}
plotMA(res, ylim=c(-20,20), main='Differentially expressed genes in early
 vs late pleura cancer')
```
```

With this analysis it is possible to find the genes that are significantly overexpressed or underexpressed `in` late tumours. We will apply the following filters:

1. Keep genes whose adjusted p-value is lower than 0.001

```

```{r result1}
res.padj <- res[(res$padj < 0.001 & !is.na(res$padj)),]
nrow(res.padj)
```

We keep `r nrow(res.padj)` genes.

2. The other variable that will be useful for the filtering is the log<sub>2</sub>fold-change. We will keep genes that have a 10 log<sub>2</sub>fold-change.

```{r top}
res.padj_4fold <- res.padj[abs(res.padj$log2FoldChange) > log2(10),]
nrow(res.padj_4fold)
```

We see that this is a more stringent criteria, because only `r nrow(res.padj_4fold)` are kept.

```{r visualize-top}
plotMA(res.padj_4fold, ylim=c(-10,10), main='Most differentially expressed genes')
```

When we represent only the most differentially expressed genes we can see that almost all of them are overexpressed.

<i class="fa fa-question-circle"></i> **Write a short summary of the result of the DE analysis. How many genes are overexpressed? And underexpressed? Fill the table**



| Underexpressed | Not differentially expressed | Overexpressed |
|----------------|------------------------------|---------------|
| 1              | 58007                        | 29            |



#### &nbsp;<i class="fas fa-comment-dots"></i> Answer: underexpressed=1, overexpressed=29, not differentially expressed=58007


</div>

### 4. Post RNA-seq analysis: visualization

In statistics, a volcano plot is a type of scatter-plot that is used to identify changes in large data sets. It plots significance vs. fold-change on the `y` and `x` axes, respectively.

More specifically, a volcano plot is essentially an scatter plot, constructed by plotting the negative log of the P-value on the `y`-axis (usually base 10). This results in data points with low P-values (highly significant) appearing toward the top of the plot. The `x`-axis

```

```

    is the **log of the fold change** between the two conditions (usually
    base 2). Each point (gene) will be colored based on the filtering (`
    filter`).

    ```{r df}
 # we first create the dataframe with the results of the DE
 resDF <- as.data.frame(res)
 # new column with the gene names
 resDF$gene <- rownames(resDF)
 # new column with TRUE/FALSE representing if a gene is deferentially
 # expressed or not
 resDF$filter <- abs(res$log2FoldChange) > log2(10) & res$padj < 0.001
 table(resDF$filter)
    ```

    1. With the previous information, build the basic `ggplot()` layer (i.e., `
    ggplot() + geom_*()`)

    ```{r ggplot1}
 plot <- ggplot(data = resDF, mapping = aes(x=log2FoldChange, y=-log(padj)))
 + geom_point()
 plot
    ```

    2. Improving the graph with publication-quality details
    + Use a nice theme (`+ theme_*`)
    + Colourblind palettes (you can use a manual palette using `
    scale_color_manual()`)
    + Use informative labels and titles (`+ labs()`)
    + Etc.

    ```{r ggplot2}
 plot + theme_bw() + scale_color_manual(values = 'blue') + labs (title= '
 Volcano plot')
    ```

    3. One finally thing we can do is to annotate the name of the DE genes. For
    that we can use the `geom_label_repel()` function from the `ggrepel`
    library.

    ```{r ggplot3}

 plot + geom_label_repel(data=resDF[abs(resDF$log2FoldChange) > log2(10) &
 resDF$padj < 0.001 ,], aes(label = as.factor(gene)), alpha = 0.7, size
 = 5, force = 1.3)

    ```

    The resulting figure could look like something like this:

    <center>
    <img src = "https://drive.google.com/uc?id=15
    MHvYftkNtE4Qn6X0qI_e99SszqcnrPmZ" width = 80%>

```

```
</center>
```

5. Post RNA-seq analysis: enrichment analysis

A common final step to interpret gene expression data is to perform a **gene set enrichment analysis** based on the functional annotation of the differentially expressed genes. This is useful **for** finding out **if** the differentially expressed genes are associated with a certain biological process or molecular **function**.

The first step is to find some information of the differentially expressed genes, using the ``biomaRt`` package.

```
```{r saveGenes}
we save a list of DE genes, and we remove the last part of the id
deGenes <- gsub("\\..*", "", rownames(res.padj_4fold))
head(deGenes)
```
```

The first step to use the ``biomaRt`` package is to select a database and a dataset. For that we use the ``useMart`` **function**. We will use as database the ``ensembl`` and as a dataset the ``hsapiens_gene_ensembl``.

```
```{r biomaRt}
biomaRt_uses <- useMart(biomaRt="ensembl", dataset="hsapiens_gene_ensembl")
biomaRt_uses
```
```

The second step is to build a `biomaRt` query. For that we use the ``getBM()`` **function**, that needs three arguments:

1. ``filter``: define a restriction
2. ``attributes``: define the values we are interested **in** to retrieve
3. ``values``: the vector of elements we want to use **for** the query

```
```{r query}
Entrez_BM <- getBM(attributes=c("ensembl_gene_id", "entrezgene_id"),
 filter="ensembl_gene_id",
 values=deGenes,
 mart=biomaRt_uses,
 uniqueRows=TRUE)

dim(Entrez_BM)
head(Entrez_BM)
```
```

We obtain a ``data.set`` with two columns, one with the gene ids **in** the Ensembl format and another with the Entrez gene format.

How many genes are there **in** the new dataset? Why?

```

<div style="background-color:#F0F0F0">
##### &emsp;<i class="fas fa-comment-dots"></i> Answer: en el nuevo dataset
      hay 34 genes, porque estos son los que estan diferencialmente
      expresados y se asocian a un proceso biol gico o una funci n
      molecular.

</div>

We can see that there are some NA values. We can clean them using the `na.
omit` function and keep those Entrez gene ids for which we have the
information

```{r clean}
Entrez_ids <- as.character(na.omit(Entrez_BM$entrezgene_id))
length(Entrez_ids)
```

<i class="fa fa-question-circle"></i> **How many genes are we keeping? Use
the `length()` function to find it out.**

<div style="background-color:#F0F0F0">
##### &emsp;<i class="fas fa-comment-dots"></i> Answer: conservamos 31
genes.

</div>

The `G0stats` R package has extensive facilities for testing the
association of Gene Ontology (GO) terms to genes in a gene list.

To perform a GO enrichment analysis, one needs to define a **gene
universe** dataset and a list of the interesting genes (i.e., the DE
expressed genes).

```{r G0}
universe <- mappedkeys(org.Hs.egGO)
count.mappedkeys(org.Hs.egGO)
```

We have information for a total of `r count.mappedkeys(org.Hs.egGO)` human
genes.

```{r testGO}
Set the threshold for the p value to 5%
G0test <- new("GOHyperGParams",
 geneIds = Entrez_ids,
 universeGeneIds=universe,
 annotation = "org.Hs.eg.db",
 ontology = "BP",
 pvalueCutoff= 0.05,
 conditional = FALSE,
 testDirection = "over")
hypergeometric test

```

```

G0testOver <- hyperGTest(G0test)
G0testOver

G0result <- summary(G0testOver)
head(G0result)

...

<i class="fa fa-question-circle"></i> **Which is the best plot to represent
the previous data? Discuss it with your colleagues and try to create a
ggplot2 graph.**

```{r data-frame-fig2}
# ggplot2 plot

plot2 <- ggplot(data = G0result, mapping = aes(x = Term, y = Count)) +
  geom_col() + labs (title= 'Gene Ontology results', x = 'GO enriched', y
= 'Number of enriched genes') + theme (axis.text.x=element_text (angle
=90, hjust=1))
#El mejor gr fico es uno de barras en el que se muestran en el eje X los
grupos de gene ontology enriquecidos y en el eje Y el n mero de genes
que hay por cada grupo

plot2
...

<i class="fa fa-key"></i> **Hint**: The plot is very similar to the ones
performed in practical P2!

<i class="fa fa-question-circle"></i> **Try to interpret the GO result.**

<div style="background-color:#F0F0F0">
#### &nbsp;<i class="fas fa-comment-dots"></i> Answer: en el gr fico se
muestran los procesos biol gicos en los que participan los genes que
hemos identificado. La mayor a de genes enriqrucidos forman parte de
los siguientes procesos biol gicos: multicellular organismal process
(>15 genes), system process (>10 genes), muscle contraction (>7) y
muscle system process (>7).

</div>

<div style="background-color: #86CBBB; 1px; height:3px " ></div>

# <i class="fa fa-cogs"></i> Perform a RNA-seq analysis

**Following the obesity pleura cancer example, perform a RNA-seq analysis
using one of the tumour data provided in the table (one cancer per
group).**

```{r data-frame-fig2}
#Environment
load('C:/Users/Usuario/Desktop/Gen tica/4rt/TAB/P4/environments/

```

```

 rse_gene_skin_env.RData')
 ...

  ```{r data-frame-fig2}
  #Grafico de expresion diferencial
  plotMA(res, ylim=c(-20,20), main='Differentially expressed genes in early
    vs late skin cancer')
  ...

  ```{r data-frame-fig2}
 #Genes que se ajustan al p-valor menor de 0.001
 res.padj <- res[(res$padj < 0.001 & !is.na(res$padj)),]
 nrow(res.padj)
 #m s diferencialmente expresados
 res.padj_4fold <- res.padj[abs(res.padj$log2FoldChange) > log2(10),]
 nrow(res.padj_4fold)
 #Total de genes:
 dim(rse_gene)
 ...

  ```{r data-frame-fig2}
  plotMA(res.padj_4fold, ylim=c(-10,10), main='Most differentially expressed
    genes')
  ...

  | Underexpressed | Not differentially expressed | Overexpressed |
  |-----|-----|-----|
  |          9          |          58025          |          3          |

  ...

  ```{r data-frame-fig2}
 ##VISUALIZATION

 # we first create the dataframe with the results of the DE
 resDF <- as.data.frame(res)
 # new column with the gene names
 resDF$gene <- rownames(resDF)
 # new column with TRUE/FALSE representing if a gene is deferentially
 expressed or not
 resDF$filter <- abs(res$log2FoldChange) > log2(10) & res$padj < 0.001
 table(resDF$filter)

 #guardamos los datos de resDF para poder realizar a parte el gr fico ya
 que en nuestro ordenador parece que el environment impide que este se
 ejecute corresctamente
 save(resDF, file = "resDF.RData")
 ...

  ```{r data-frame-fig2}
  plot <- ggplot(data = resDF, mapping = aes(x=log2FoldChange, y=-log(padj)))
    + geom_point() + theme_bw() + scale_color_manual(values = 'blue') +
    labs (title= 'Volcano plot') + geom_label_repel(data=resDF[abs(
    resDF$log2FoldChange) > log2(10) & resDF$padj < 0.001 ,], aes(label =

```

```

    as.factor(gene)), alpha = 0.7, size = 5, force = 1.3)
plot
```

```{r data-frame-fig2}
##POST RNA-SEQ ANALYSIS: ENRICHMENT ANALYSIS
# we save a list of DE genes, and we remove the last part of the id
deGenes <- gsub("\\..*", "", rownames(res.padj_4fold))
head(deGenes)
biomart_uses <- useMart(biomart="ensembl", dataset="hsapiens_gene_ensembl")
biomart_uses
```

```{r data-frame-fig2}
Entrez_BM <- getBM(attributes=c("ensembl_gene_id", "entrezgene_id"),
                    filter="ensembl_gene_id",
                    values=deGenes,
                    mart=biomart_uses,
                    uniqueRows=TRUE)

dim(Entrez_BM)
```

```{r data-frame-fig2}
head(Entrez_BM)

data.frame(Entrez_BM)
#En el nuevo dataset hay 12 genes, porque estos son los que estan
diferencialmente expresados y se asocian a un proceso biol gico o una
funci n molecular.
```

```{r data-frame-fig2}
Entrez_ids <- as.character(na.omit(Entrez_BM$entrezgene_id))

length(Entrez_ids)
#Conservamos 10 genes
```

```{r data-frame-fig2}
universe <- mappedkeys(org.Hs.egG0)
count.mappedkeys(org.Hs.egG0)
```

```{r data-frame-fig2}
# Set the threshold for the p value to 5%
G0test <- new("GOHyperGParams",
              geneIds = Entrez_ids,
              universeGeneIds=universe,
              annotation = "org.Hs.eg.db",
              ontology = "BP",
              pvalueCutoff= 0.05,

```



```

        conditional = FALSE,
        testDirection = "over")
# hypergeometric test
G0testOver <- hyperGTest(G0test)
G0testOver
```

```{r data-frame-fig2}
G0result <- summary(G0testOver)
head(G0result)
```

```{r data-frame-fig2}
G0result
#guardamos los datos de G0result para poder realizar a parte el gr fico ya
#que en nuestro ordenador parece que el environment impide que este se
#ejecute correctamente
save (G0result, file = "Goresults.RData")
```

```{r data-frame-fig2}
ggplot2 <- ggplot(data = G0result, mapping = aes(x = Term, y = Count)) +
  geom_col() + labs (title= 'Gene Ontology Results Skin Cancer', x = 'GO
  enriched', y = 'Number of enriched genes') + theme (axis.text.x=
  element_text (angle=90, hjust=1))

ggplot2

#En el gr fico se muestran los procesos biol gicos en los que participan
#los genes que hemos identificado. La mayor a de genes enriquecidos
#forman parte de procesos de activaci n, desarrollo y migraci n de
#queratina, diferenciaci n epitelial, del sistema inmune y del
#desarrollo epid rmico.

```


25


```

```
proper objects names)

<div style="background-color: #86CBBB; 1px; height:3px " ></div>

4. Upload your results to your GitHub

Upload this `Rmd` document and the figures you have generated to your
GitHub repository.

<div style="background-color: #86CBBB; 1px; height:3px " ></div>

5. References

Practical based on Juan Ram n Gonz lez ' material available at [GitHub](
 https://github.com/isglobal-brge/TeachingMaterials/tree/master/
 Master_Bioinformatics).

<div style="background-color: #86CBBB; 1px; height:3px " ></div>


```