Evaluation assignment for LEO Pharma A/S

Scientific Data Administrator and Visualizations Assistant

Name: PABLO IÁÑEZ PICAZO

For this assignment I'm going to use tidyverse package in order to manage data structures and plot graphical analysis as I personally think the implementation code is easier, more elegant and more intuitive.

In the beginning I realized that I had to upgrade several packages from Bioconductor as well as my current R version in order to start working on the assignment. I had in mind using DEseq2 package for differential expression analysis, as we used it in 'Bioinformatics of High Throughput Analysis' course in the first year of MSc in Bioinformatics. However, I found out that DEseq2 works only with data count, making it suitable for RNA-seq analysis but not for microarray analysis. Therefore, I installed the limma package from my Rstudio console.

while I wait for the update to finish, I start checking the next step of the assignment. I need to download GSE58558 from GEO database, and I look for information on how to do this with Bioconductor. I find out that I can install and use GEOquery package to download the experiment data and parse it into my machine. I read about the different types of GEO data structures, and more specifically about GSE data class.

I had several problems with BiocManager in order to install GEOquery, but after several tries and fighting with Rstudio console, I managed to install it.

```
library(GEOquery)
library(tidyverse)

# Parse the GSE58558 data from GEO database. If GSEMatrix=TRUE (default),
# the output will be an ExpressionSet object, needed for further analysis
gse <- getGEO('GSE58558', GSEMatrix = TRUE)
```

DATA ANNOTATION

- How many experiment files does the dataset contain?

The GSE file contains 109 GSM files, from GSM1413895 to GSM1414003, each one corresponding to a different set of probe elements (platform). Each file name is shown when parsing the GSE data into R. We can also see a summary of the experiment series using the command below.

```
show(gse)
```

```
## $GSE58558_series_matrix.txt.gz
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 54675 features, 109 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: GSM1413895 GSM1413896 ... GSM1414003 (109 total)
##   varLabels: title geo_accession ... time:ch1 (42 total)
##   varMetadata: labelDescription
## featureData
##   featureNames: 1007_s_at 1053_at ... AFFX-TrpnX-M_at (54675
##     total)
```

```
##   fvarLabels: ID GB_ACC ... Gene Ontology Molecular Function (16
##     total)
##   fvarMetadata: Column Description labelDescription
## experimentData: use 'experimentData(object)'
##   pubMedIds: 24786238
## Annotation: GPL570
```

How many samples belong to lesional, how many to the non-lesional group? Any others?

First I need to find which column from the phenotype data frame belongs to the lesion variable.

```
# Extract phenotype data
dataframe <- pData(phenoData(gse[[1]]))

# Turn phenotype data into tibble for tidyverse operations
tib <- as_tibble(dataframe)
colnames(tib)
```

```
##  [1] "title"                   "geo_accession"
##  [3] "status"                  "submission_date"
##  [5] "last_update_date"        "type"
##  [7] "channel_count"           "source_name_ch1"
##  [9] "organism_ch1"            "characteristics_ch1"
## [11] "characteristics_ch1.1"   "characteristics_ch1.2"
## [13] "characteristics_ch1.3"   "characteristics_ch1.4"
## [15] "characteristics_ch1.5"   "molecule_ch1"
## [17] "extract_protocol_ch1"    "label_ch1"
## [19] "label_protocol_ch1"      "taxid_ch1"
## [21] "hyb_protocol"            "scan_protocol"
## [23] "description"             "description.1"
## [25] "data_processing"         "platform_id"
## [27] "contact_name"            "contact_email"
## [29] "contact_institute"       "contact_address"
## [31] "contact_city"            "contact_state"
## [33] "contact_zip/postal_code" "contact_country"
## [35] "supplementary_file"      "data_row_count"
## [37] "age:ch1"                 "gender:ch1"
## [39] "lesions:ch1"             "responder:ch1"
## [41] "scorad:ch1"              "time:ch1"
```

The information I'm looking for is in the column with name 'characteristics_ch1.1' or 'lesions:ch1'. In order to check if there are more values apart from 'lesional' and 'no lesional':

```
unique(tib$`lesions:ch1`)
```

```
## [1] "non lesional" "lesional"
```

As this output shows, there's no any other label apart from these two conditions. In order to count how many samples belong to each group I can count the number of rows with each:

```
nrow(filter(tib, `lesions:ch1` == "lesional"))
```

```
## [1] 56
```

```
nrow(filter(tib, `lesions:ch1` == "non lesional"))
```

```
## [1] 53
```

There are 56 samples belonging to the lesional group (patients with Atopic Dermatitis) and 53 samples belonging to the non-lesional group (patients with no Atopic Dermatitis). These 109 samples come from 19 different individues.

Re-annotate and filter data

After reading the scientific paper related to this experiment, I found out that the samples from untreated data are those belonging to the samples obtained on week0.

```
unique(tib$`time:ch1`)
```

```
## [1] "day 1"   "week 2"  "week 12"
```

The samples taken on week0 are annotated as 'day 1' in the 'characteristics_ch1' or 'time:ch1' column. Therefore I filter out the samples that were obtained later.

```
untreated_tib <- tib %>% filter(`time:ch1` == "day 1")
unique(untreated_tib$`time:ch1`)
```

```
## [1] "day 1"
```

After the filtering, I end up with 35 samples (35 rows), instead of 109. From here on, I apply a series of transformations as indicated in the instructions.

```
# Create a table with only the sample ID, the path to the supplementary file,
# the lesions status and the patient ID, and change column names
filt <- untreated_tib %>% select(geo_accession, supplementary_file, `lesions:ch1`, title)
colnames(filt) <- c("sample_id", "file_name", "tissue", "patient_id")

# Change 'lesional' label to 'LS' and 'non lesional' to 'NL'
# Also, trim patient_id into patient index only
filt <- filt %>%
        mutate(tissue=replace(tissue, tissue=="lesional", "LS")) %>%
        mutate(tissue=replace(tissue, tissue=="non lesional", "NL")) %>%
        separate(patient_id, c('patient_id', NA), " ")

# Save and output the resulting tibble as a .csv file
write_csv(filt, "Pablo_Anno.csv")
```

DATA ANALYSIS

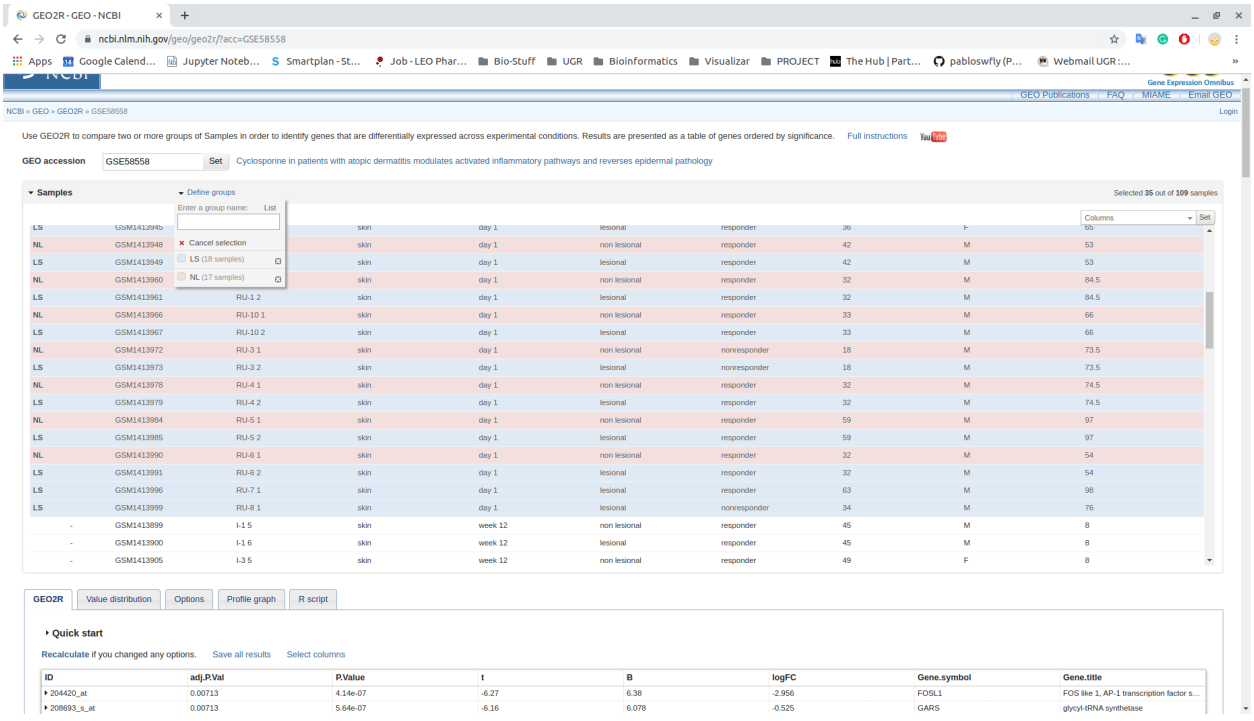
In the next code chunk I show how the Differential Expression analysis could be done with limma. However I never used this package before, as I mostly worked before with RNA-seq and DEseq2 packages. I don't know how to implement the study design we obtained previously when running the linear fit. As a result of this, all the values I obtained are the same

```
library(limma)
gene_data <- exprs(gse[[1]])
# design <- load.design(filt) and other extra steps
fit <- lmFit(gene_data)
fit <- eBayes(fit)
topTable(fit)
```

```
##                logFC  AveExpr        t       P.Value     adj.P.Val
## 236419_at    2.169565 2.169565 6920.202 4.452391e-308 7.398127e-307
## 233542_at    2.169565 2.169565 6920.202 4.452391e-308 7.398127e-307
```

```
## 234622_at    2.169565 2.169565 6920.202 4.452391e-308 7.398127e-307
## 216932_at    2.169565 2.169565 6920.202 4.452391e-308 7.398127e-307
## 1559344_at   2.169565 2.169565 6920.202 4.452391e-308 7.398127e-307
## 224100_s_at  2.169565 2.169565 6920.202 4.452391e-308 7.398127e-307
## 1553321_a_at 2.169565 2.169565 6920.202 4.452391e-308 7.398127e-307
## 1559575_a_at 2.169565 2.169565 6920.202 4.452391e-308 7.398127e-307
## 207646_s_at  2.169565 2.169565 6920.202 4.452391e-308 7.398127e-307
## 211875_x_at  2.169565 2.169565 6920.202 4.452393e-308 7.398127e-307
##                    B
## 236419_at    674.994
## 233542_at    674.994
## 234622_at    674.994
## 216932_at    674.994
## 1559344_at   674.994
## 224100_s_at  674.994
## 1553321_a_at 674.994
## 1559575_a_at 674.994
## 207646_s_at  674.994
## 211875_x_at  674.994
```

I found myself spending too much time on understanding Limma user guide (http://www.bioconductor.org/packages/devel/bioc/vignettes/limma/inst/doc/usersguide.pdf) This is why I decided to leave limma aside and perform the Differential Expression Analysis on GEO, using GEO2R. I created 2 design groups: NL corresponding to the 'non lesional' samples and LS corresponding to the 'lesional' samples. I only selected the samples which were taken on day 1, in order to work only with untreated data. The GEO browser looked as follows:



I downloaded the GEO2R analysis results into my PC and parsed the data:

```r
# Load the data from GEO2R DE analysis
DE <- as_tibble(read_delim("geo2r.txt", "\t"))

# Re-order columns and output the table as a .csv file
DE <- select(DE, ID, Gene.symbol, logFC, P.Value, everything())
write_csv(DE, "Pablo_Coefficients.csv")

# Select row corresponding to IL13 in order to obtain the log2 Fold Change
solution <- DE %>% filter(Gene.symbol == 'IL13')
solution
```

```
## # A tibble: 1 x 8
##   ID       Gene.symbol   logFC P.Value adj.P.Val      t     B Gene.title
##   <chr>    <chr>         <dbl>   <dbl>     <dbl>  <dbl> <dbl> <chr>
## 1 207844_~ IL13        -0.0636    0.33     0.755 -0.989 -6.22 interleukin ~
```

The Log2 Fold Change for IL13 is -0.0636, with a P value of 0.33. From this data, we can observe that Interleukin 13 is down-regulated in patients with Atopic Dermatitis, even though it's not very significant because the P-value is high (0.33). It would be statistically significant if the p-value was below 0.05 (for a confidence interval of 95%).

DATA VISUALIZATION

The expression values in exprs(gse) are already normalized and log2-transformed. In order to work with the raw data, I thought I had to download the supplementary files with the function getGEOSuppFiles('GSE58558'). After dealing with a huge amount of parsing data and a large waiting time, I thought there has to be a simpler way of achieving this. Even if I didn't find the GEOquery documentation clear, I understand that parsing the data with getGEO function setting the GSEMatrix variable as FALSE would return expression values from raw .CEL data. From here, I perform several operations in order to obtain a data matrix with expression values, with help from GEOquery documentation. (https://rdrr.io/bioc/GEOquery/f/vignettes/GEOquery.Rmd)

```r
raw <- getGEO('GSE58558', GSEMatrix = FALSE)

# Reference: GEOquery documentation - Converting GSE to an ExpressionSet
gsmplatforms <- lapply(GSMList(raw),function(x) {Meta(x)$platform})
Table(GSMList(raw)[[1]])[1:5,]
```

```
##      ID_REF     VALUE
## 1 1007_s_at 11.344988
## 2   1053_at  9.260948
## 3    117_at  6.016001
## 4    121_at  3.075750
## 5 1255_g_at  2.169565
```

```r
probesets <- Table(GPLList(raw)[[1]])$ID
data.matrix <- do.call('cbind',lapply(GSMList(raw),function(x)
                                      {tab <- Table(x)
                                      mymatch <- match(probesets,tab$ID_REF)
                                      return(tab$VALUE[mymatch])
                                      }))

data.matrix <- apply(data.matrix, 2, function(x) {as.numeric(as.character(x))})
rownames(data.matrix) <- probesets
colnames(data.matrix) <- names(GSMList(raw))
```

```
# Transform data frame into tibble
raw_tibble <- as_tibble(data.matrix, rownames = NA)
raw_tibble <- mutate(raw_tibble, ID = probesets)
```

I'm asked to work only with keratin 16 (KRT16) protein, so I look up it's gene ID in the Differential Expression analysis output from GEO2R.

```
krt16 <- DE %>% filter(Gene.symbol == 'KRT16')
krt16
```

```
## # A tibble: 1 x 8
##   ID        Gene.symbol logFC   P.Value adj.P.Val     t     B Gene.title
##   <chr>     <chr>       <dbl>     <dbl>     <dbl> <dbl> <dbl> <chr>
## 1 209800_at KRT16       -2.33 0.0000024    0.0119 -5.68  4.67 keratin 16
```

KRT16 is listed as 209800_at. I extract the information related to this entry from the raw data tibble, as well as selecting only those samples from day 1. Reminder: The samples ID from time:day 1 are stored in filt$sample_id variable.

```
day1_krt16 <- raw_tibble[colnames(raw_tibble) %in% filt$sample_id] %>%
              mutate(ID = probesets) %>%
              filter(ID == '209800_at')

day1_krt16
```

```
## # A tibble: 1 x 36
##   GSM1413895 GSM1413896 GSM1413901 GSM1413902 GSM1413907 GSM1413908
##        <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
## 1       10.8       13.3       9.80       12.3       9.49       12.7
## # ... with 30 more variables: GSM1413913 <dbl>, GSM1413914 <dbl>,
## #   GSM1413919 <dbl>, GSM1413922 <dbl>, GSM1413923 <dbl>,
## #   GSM1413928 <dbl>, GSM1413929 <dbl>, GSM1413934 <dbl>,
## #   GSM1413935 <dbl>, GSM1413940 <dbl>, GSM1413941 <dbl>,
## #   GSM1413944 <dbl>, GSM1413945 <dbl>, GSM1413948 <dbl>,
## #   GSM1413949 <dbl>, GSM1413960 <dbl>, GSM1413961 <dbl>,
## #   GSM1413966 <dbl>, GSM1413967 <dbl>, GSM1413972 <dbl>,
## #   GSM1413973 <dbl>, GSM1413978 <dbl>, GSM1413979 <dbl>,
## #   GSM1413984 <dbl>, GSM1413985 <dbl>, GSM1413990 <dbl>,
## #   GSM1413991 <dbl>, GSM1413996 <dbl>, GSM1413999 <dbl>, ID <chr>
```

Now I will prepare the data in order to plot the boxplot.

```
# Get rows relative to lesional (LS) and non lesional (NL) samples
lesion <- filter(filt, tissue == 'LS')
no_lesion <- filter(filt, tissue == 'NL')

# Select and prepare raw data for KTR16-DAY1-LS
day1_lesion <- day1_krt16[colnames(day1_krt16) %in% lesion$sample_id] %>%
               gather(key = "sample", value = "value") %>%
               mutate(lesion = "lesional")

# Select and prepare the raw data for KTR16-DAY1-NL
day1_no_lesion <- day1_krt16[colnames(day1_krt16) %in% no_lesion$sample_id] %>%
                  gather(key = "sample", value = "value") %>%
                  mutate(lesion = "non lesional")

# Bind both data tibbles together again
```

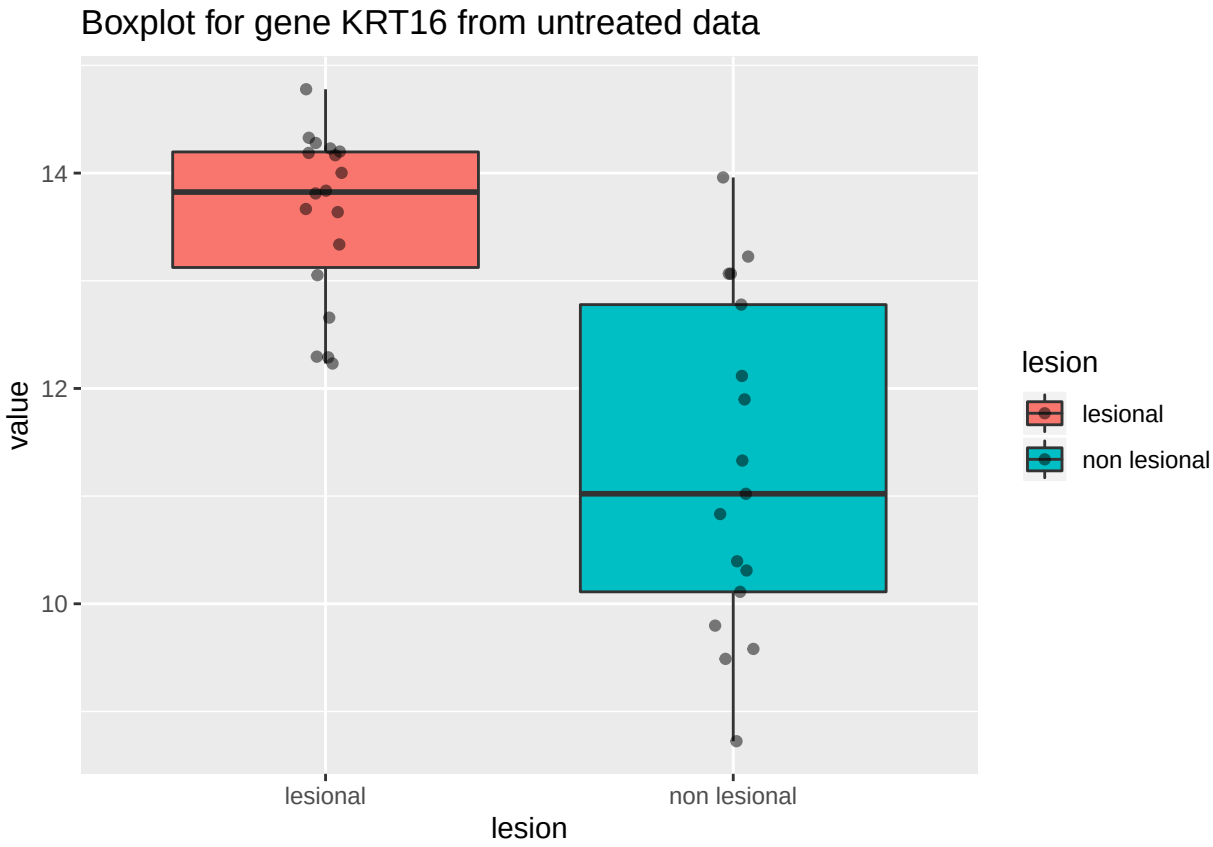
```
boxdata <- rbind(day1_lesion, day1_no_lesion)

# Output statistics summary
boxdata %>% group_by(lesion) %>% summarise(mean = mean(value),
            median = median(value), var = var(value))
```

```
## # A tibble: 2 x 4
##   lesion        mean median   var
##   <chr>        <dbl>  <dbl> <dbl>
## 1 lesional      13.6   13.8 0.625
## 2 non lesional  11.3   11.0 2.43
```

With this summary we can already know what to expect from the boxplot. The lesional sample group has a higher average expression value and a smaller variance, while the non lesional sample group has lower average expression value and a higher variance.

```
boxdata %>%
  ggplot(aes(x=lesion, y=value, fill=lesion)) +
  geom_boxplot() +
  geom_jitter(width = 0.05, alpha = 0.5 ) +
  ggtitle("Boxplot for gene KRT16 from untreated data")
```



For the last bonus task of the evaluation test, I'm going to plot the heatmap for the asked genes. In the beginning I had problems when calling the pheatmap() function. After exploring different alternatives, I found out that I had a column of string variables for the gene ID that I had to get rid off in order to construct the heatmap. I do this transformation using select(data, -ID).

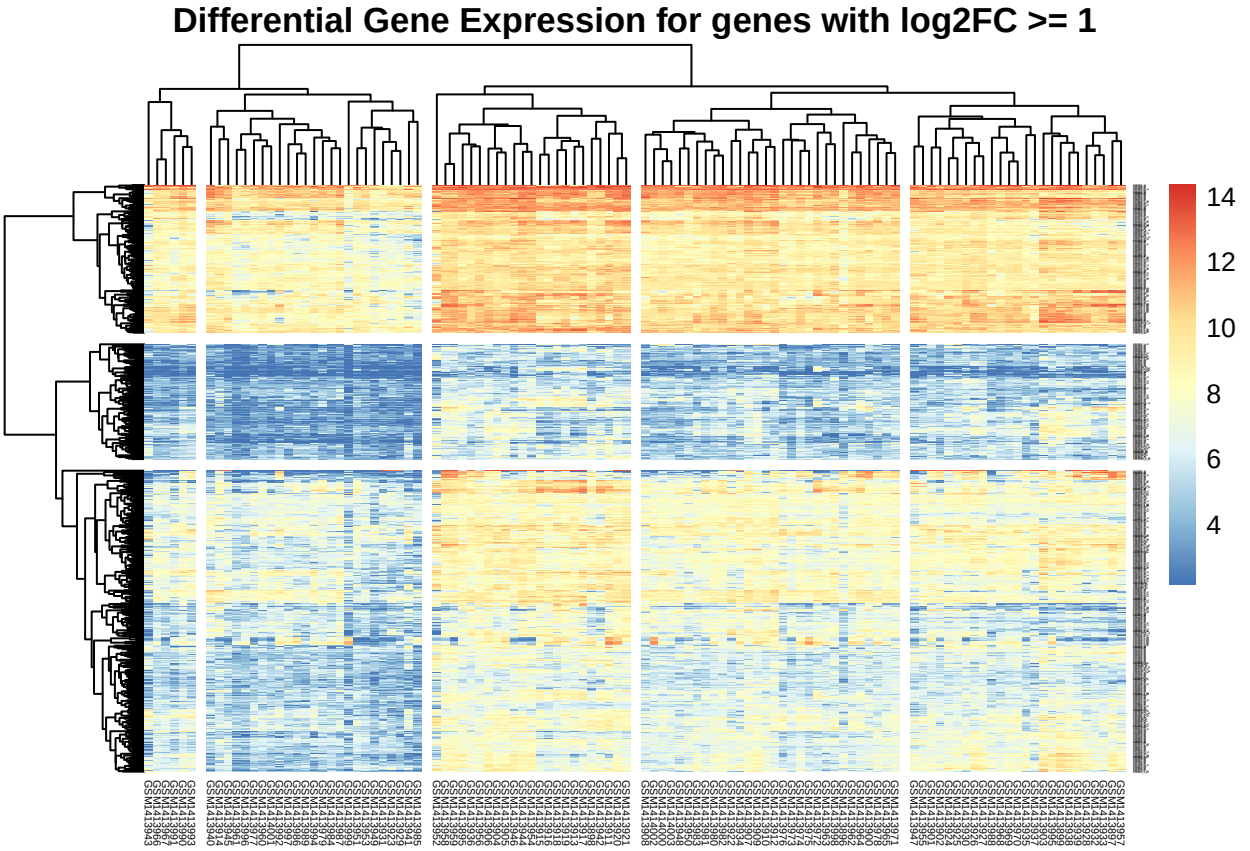
```
# Parse in the data from the previous Differential Expression analysis
DE <- read_csv("Pablo_Coefficients.csv")
```

```
# Extract the gene rows which a log2 Fold Change higher than 1 (up-regulated)
valid <- filter(DE, logFC >= 1)

# Prepare some variables to draw the heatmap
FC1_tibb <- raw_tibble[raw_tibble$ID %in% valid$ID,]
row_names <- pull(FC1_tibb, ID)
FC1_tibb <- select(FC1_tibb, -ID)
col_names <- colnames(FC1_tibb)

# Libraries required to plot the heatmap
library(pheatmap)
library(ggrepel)

# Draw the graphic with pheatmap()
pheatmap(FC1_tibb, fontsize_row = 1, fontsize_col = 4, cutree_rows = 3,
         cutree_cols = 5, cluster_rows = TRUE, cluster_cols = TRUE,
         labels_row = row_names, labels_col = col_names,
         main="Differential Gene Expression for genes with log2FC >= 1")
```



The output plot was saved as pheatmap.pdf with higher resolution and size. The file is also compressed in the zip file.

```
dim(FC1_tibb)
```

```
## [1] 602 109
```

The heatmap have dimensions 602x109. Even though it's possible to read the column names (sampleID), the genesIDs are unreadable (there are 602 rows). I believed I could avoid this problem with ggrepel package, as

it performs good on the majority of ggplot graphics. However, I wasn't able to fix this issue in the heatmap. An improvement on this heatmap would be making the geneID labels legible.

The function pheatmap() performs by default clustering on both columns and rows. I played a little with the kmeans_k argument value to see the best k value in which to cluster the data. In order to make the visualization clearer, I added rows and columns gaps with cutree_rows and cutree_col, dividing the heatmap into different observable clusters.

Adding tissue group information is possible using pheatmap annotation_col argument. This argument would take a data frame with (lesional/non lesional) values. This way, the heatmap would include an extra row of cells on top with a different color depending on the lesion value. The data frame should look more or less like this:

```
#annotation_col = data.frame(lesionClass = rep(c("lesional", "non lesional"),
#                                               a_value_vector))
```