

# Bioinformatics for microarray analysis

Martin Grasshoff

Martin Manolov

Vanessa Klöcker

Nov 14, 2022

## Contents

<b>1 Download the data from GEO</b>	<b>2</b>
<b>2 Process the expression table for analysis</b>	<b>4</b>
<b>3 Differential Expression Analysis by Limma</b>	<b>7</b>
<b>4 PCA (Principal component analysis)</b>	<b>11</b>
<b>5 Clustering by Heatmap</b>	<b>12</b>
<b>6 GO (Gene Ontology) Enrichment Analysis</b>	<b>14</b>
<b>7 GSEA</b>	<b>15</b>
<b>8 Combining other dataset for PCA and clustering</b>	<b>17</b>

We will show here a typical pipeline for the analysis of gene expression data. This tutorial is based on Monocyte and Macrophage data from the following paper:

van de Laar L, Saelens W, De Prijck S, Martens L et al. Yolk Sac Macrophages, Fetal Liver, and Adult Monocytes Can Colonize an Empty Niche and Develop into Functional Tissue-Resident Macrophages. *Immunity* 2016 Apr 19;44(4):755-68. PMID: 26992565

This tutorial contains the following steps:

1. Download the data from GEO
2. Process the expression table for analyses
3. DE Analysis by Limma
4. PCA (Principal component analysis)
5. Clustering by Heatmap (only top 250 genes)
6. GO (Gene Ontology) Enrichment Analysis
7. GSEA (Gene Set Enrichment Analysis)
8. Combining other dataset for PCA and clustering

You can find 3 files given to you under **handout** directory:

File	Description
handout.pdf	Step-by-step explanation of the tasks.
Imun_gen.ex.Rdump	Preprocessed object from GEO for loading in the analysis.
pheno.table	A table storing the phenotype data for comparison.
mouse_hallmark_genesets.rdata	Hallmark mouse gene sets for GSEA

**Set the working directory** Before you start to work on this handout, please set your working directory to the place where you save the directory. For example,

```
setwd("/home/vkraemer/Documents/BIR_course/")  
# Please use your path instead of the above example
```

```
rm(list=ls())  
if("mogene10sttranscriptcluster.db" %in% rownames(installed.packages()) == FALSE) {  
  BiocManager::install(c("mogene10sttranscriptcluster.db"))  
}  
if("msigdb" %in% rownames(installed.packages()) == FALSE) {  
  BiocManager::install("msigdb")  
}  
  
library(Biobase)  
library(GEOquery)  
library(limma)  
library(mogene10sttranscriptcluster.db)  
library(RColorBrewer)  
library(gProfileR)  
library(genefilter)  
library(openxlsx)  
library(piano)  
library(gplots)  
library(sva)  
library(msigdb)
```

**Loading the libraries** **Biobase** contains base functions for Bioconductor; **GEOquery** is the bridge between GEO and BioConductor; **limma** is a library for the analysis of gene expression microarray data, especially the use of linear models for analysing designed experiments and the assessment of differential expression; **mogene10sttranscriptcluster.db** contains the mouse gene annotation of mapping between different systems; **RColorBrewer** can make colorful palettes; **gProfileR** is the package to perform gene enrichment analysis; **openxlsx** is for creating excel table in R; **piano** is the package for GSEA; **gplots** contains many plotting tools; and **sva** (ComBat) is used here to remove known batch effects from microarray data.

## 1 Download the data from GEO

After you find a dataset, which you want to explore, you can search in GEO using their GEO number. All the details of the protocol, platform and processing steps can be found in the GEO webpage.

Please take a look of this example: <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE76999>

First, you need to download the data from GEO by the GEO number in R. We use the function **getGEO** (from package **GEOquery**) to download a list of all the available data.

```
# Load the data from GEO for GSE76999  
# Load data from file  
GEO_List <- getGEO(filename = "/home/vkraemer/Documents/BIR_course/GSE76999_series_matrix.txt.gz", GSEMatrix = TRUE)  
# Load data via online query  
# GEO_List <- getGEO(GEO = "GSE76999", GSEMatrix = FALSE, getGPL = FALSE)  
  
# Check the type of this variable  
class(GEO_List)
```

```
## [1] "ExpressionSet"
## attr(,"package")
## [1] "Biobase"

# Check the length of this list
length(GEO_List)

## [1] 1

# An element of a list always comes together with a name, right?
names(GEO_List)

## NULL

# Since there is only one ExpressionSet, let's assign it into another variable
eset <- GEO_List
```

**ExpressionSet** (*eset* here) is a Container for high-throughput assays and experimental metadata.

In order to understand **ExpressionSet**, type:

```
eset # Get an overview

## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 35556 features, 36 samples
## element names: exprs
## protocolData: none
## phenoData
## sampleNames: GSM2042244 GSM2042245 ... GSM2042279 (36 total)
## varLabels: title geo_accession ... tissue:ch1 (40 total)
## varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## pubMedIds: 26992565
## Annotation: GPL6246

varLabels(eset) # To see what kinds of labels inside

## [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] "treatment_protocol_ch1" "growth_protocol_ch1"
## [15] "molecule_ch1" "extract_protocol_ch1"
## [17] "label_ch1" "label_protocol_ch1"
## [19] "taxid_ch1" "hyb_protocol"
## [21] "scan_protocol" "description"
## [23] "data_processing" "data_processing.1"
## [25] "data_processing.2" "platform_id"
## [27] "contact_name" "contact_email"
## [29] "contact_department" "contact_institute"
## [31] "contact_address" "contact_city"
## [33] "contact_zip/postal_code" "contact_country"
## [35] "supplementary_file" "data_row_count"
## [37] "relation" "age:ch1"
## [39] "strain:ch1" "tissue:ch1"
```

```
eset$title # To see the names of the experiments
```

```
## [1] "Monocyte extracted from adult (wk6-12) Bone Marrow, biological replicate 1"
## [2] "Monocyte extracted from adult (wk6-12) Bone Marrow, biological replicate 2"
## [3] "Monocyte extracted from adult (wk6-12) Bone Marrow, biological replicate 3"
## [4] "Monocyte extracted from adult (wk6-12) Bone Marrow, biological replicate 4"
## [5] "Monocyte extracted from E15.5 Fetal Liver, biological replicate 1"
## [6] "Monocyte extracted from E15.5 Fetal Liver, biological replicate 2"
## [7] "Monocyte extracted from E15.5 Fetal Liver, biological replicate 3"
## [8] "Monocyte extracted from E15.5 Fetal Liver, biological replicate 4"
## [9] "Macrophage extracted from E12.5 Yolk Sac, biological replicate 1"
## [10] "Macrophage extracted from E12.5 Yolk Sac, biological replicate 2"
## [11] "Macrophage extracted from E12.5 Yolk Sac, biological replicate 3"
## [12] "Macrophage extracted from E12.5 Yolk Sac, biological replicate 4"
## [13] "Alveolar Macrophage extracted from adult (wk6-12) mice via Bronchoalveolar lavage, biological replicate 1"
## [14] "Alveolar Macrophage extracted from adult (wk6-12) mice via Bronchoalveolar lavage, biological replicate 2"
## [15] "Alveolar Macrophage extracted from adult (wk6-12) mice via Bronchoalveolar lavage, biological replicate 3"
## [16] "Alveolar Macrophage extracted from adult (wk6-12) mice via Bronchoalveolar lavage, biological replicate 4"
## [17] "Alveolar Macrophage derived from transferred Bone Marrow Monocytes, 6 wks post-transfer, biological replicate 1"
## [18] "Alveolar Macrophage derived from transferred Bone Marrow Monocytes, 6 wks post-transfer, biological replicate 2"
## [19] "Alveolar Macrophage derived from transferred Bone Marrow Monocytes, 6 wks post-transfer, biological replicate 3"
## [20] "Alveolar Macrophage derived from transferred Bone Marrow Monocytes, 6 wks post-transfer, biological replicate 4"
## [21] "Alveolar Macrophage derived from transferred Fetal Liver Monocytes, 6 wks post-transfer, biological replicate 1"
## [22] "Alveolar Macrophage derived from transferred Fetal Liver Monocytes, 6 wks post-transfer, biological replicate 2"
## [23] "Alveolar Macrophage derived from transferred Fetal Liver Monocytes, 6 wks post-transfer, biological replicate 3"
## [24] "Alveolar Macrophage derived from transferred Fetal Liver Monocytes, 6 wks post-transfer, biological replicate 4"
## [25] "Alveolar Macrophage derived from transferred Yolk Sac Macrophages, 6 wks post-transfer, biological replicate 1"
## [26] "Alveolar Macrophage derived from transferred Yolk Sac Macrophages, 6 wks post-transfer, biological replicate 2"
## [27] "Alveolar Macrophage derived from transferred Yolk Sac Macrophages, 6 wks post-transfer, biological replicate 3"
## [28] "Alveolar Macrophage derived from transferred Yolk Sac Macrophages, 6 wks post-transfer, biological replicate 4"
## [29] "Alveolar Macrophage derived from transferred Alveolar macrophages, 6 wks post-transfer, biological replicate 1"
## [30] "Alveolar Macrophage derived from transferred Alveolar macrophages, 6 wks post-transfer, biological replicate 2"
## [31] "Alveolar Macrophage derived from transferred Alveolar macrophages, 6 wks post-transfer, biological replicate 3"
## [32] "Alveolar Macrophage derived from transferred Alveolar macrophages, 6 wks post-transfer, biological replicate 4"
## [33] "Alveolar Macrophage extracted from 6 wk old wild type mice via Bronchoalveolar lavage, biological replicate 1"
## [34] "Alveolar Macrophage extracted from 6 wk old wild type mice via Bronchoalveolar lavage, biological replicate 2"
## [35] "Alveolar Macrophage extracted from 6 wk old wild type mice via Bronchoalveolar lavage, biological replicate 3"
## [36] "Alveolar Macrophage extracted from 6 wk old wild type mice via Bronchoalveolar lavage, biological replicate 4"
```

```
eset$organism_ch1 # Show the organism of the experiments
```

```
## [1] "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus"
## [6] "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus"
## [11] "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus"
## [16] "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus"
## [21] "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus"
## [26] "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus"
## [31] "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus" "Mus musculus"
## [36] "Mus musculus"
```

## 2 Process the expression table for analysis

Here we select only the first 12 samples for downstream analysis and eliminate others. These 12 samples are **BM-MOs** (bone marrow monocytes), **FL-MOs** (fetal liver monocytes), and **YS-Macs** (yolk sac-derived

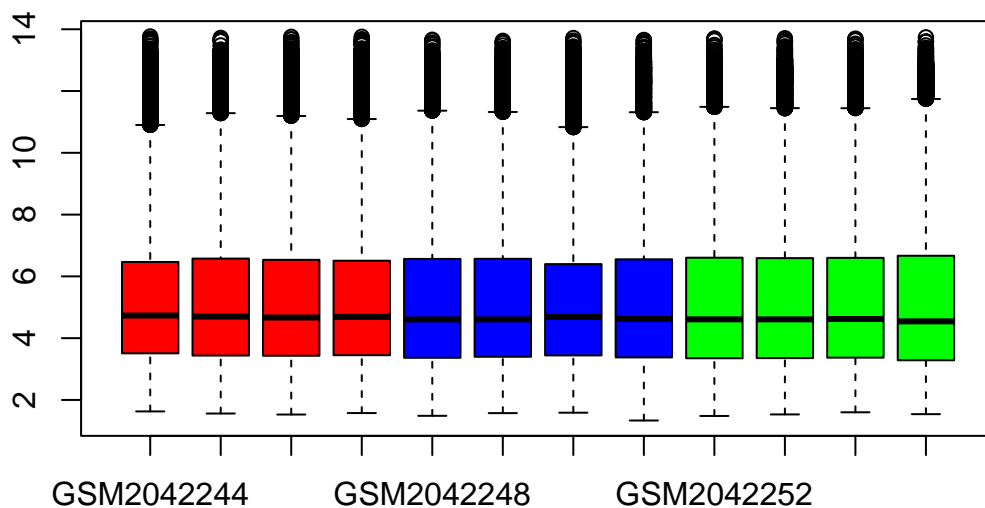
macrophages), with 4 replicates for each.

```
# Select the samples we need
sel <- 1:12
eset <- eset[,sel]
# We also need to modify its expression matrix
exprs(eset) <- exprs(eset)[,sel]
# Create labels for later steps
labels <- c("BM_MOs", "FL_MOs", "YS_Macs")
sel_groups <- factor(rep(labels, each=4))

# Lots of colors you can choose: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf
# But let's try the most simple ones
# palette function is used to set the customized colors
palette(c("red", "blue", "green"))
```

Before we do any further analysis, we need to do quality control of the data. We use boxplot to check whether we need further normalization on the data.

```
boxplot(exprs(eset), col=sel_groups)
```

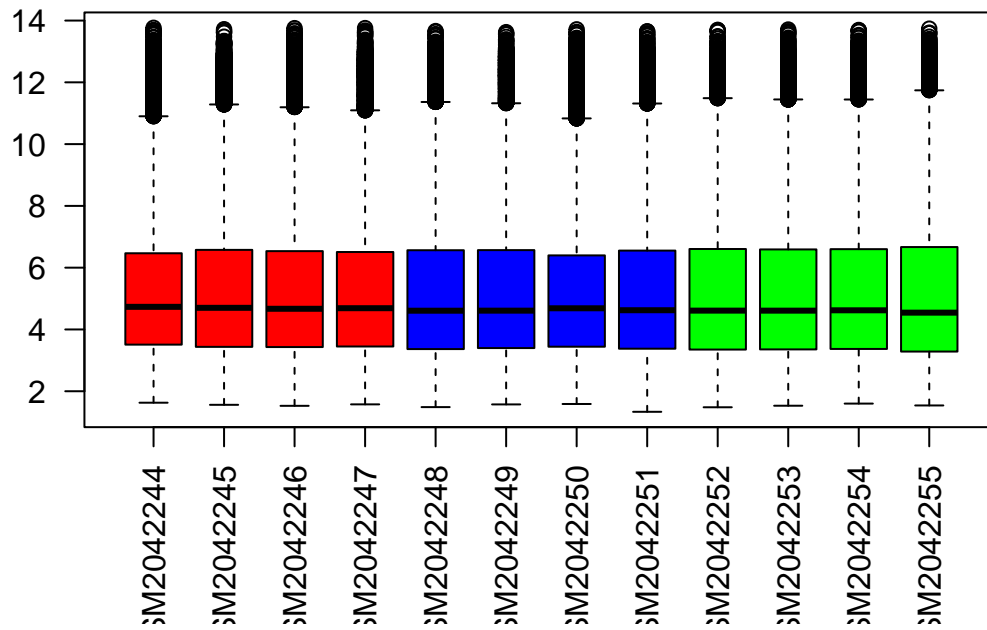


How is it? Are you satisfied? I am not. In order to learn all the parameters you can adjust for this boxplot, you can read the help message by typing:

```
?boxplot
```

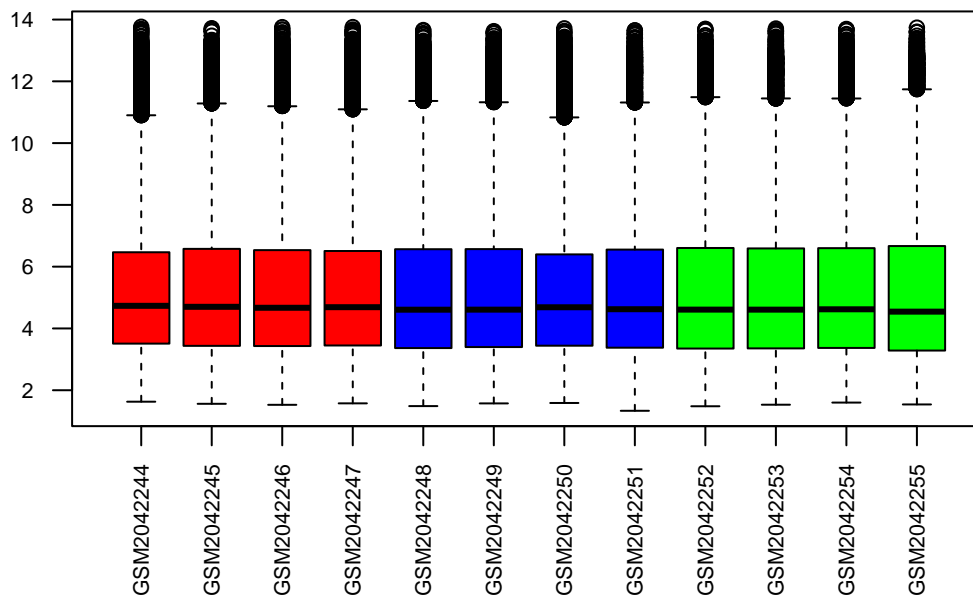
I will show a few parameters that can be used to improve this plot. Let's rotate the labels on the x-axis!!

```
boxplot(exprs(eset), col=sel_groups, las=2)
```



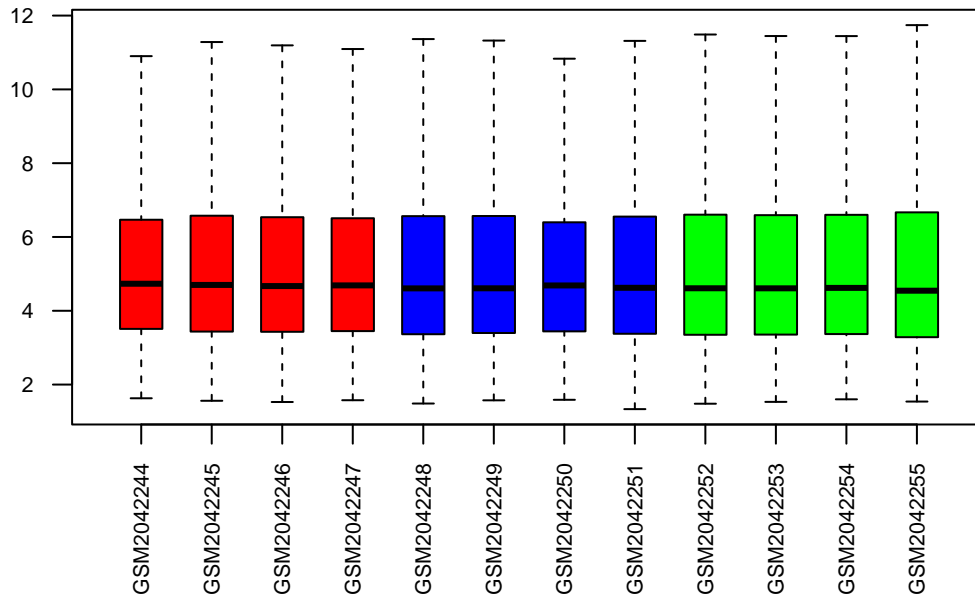
A little bit better, but not complete. Let's make the label smaller.

```
boxplot(exprs(eset), col=scl_groups, las=2, cex.axis = 0.7)
```



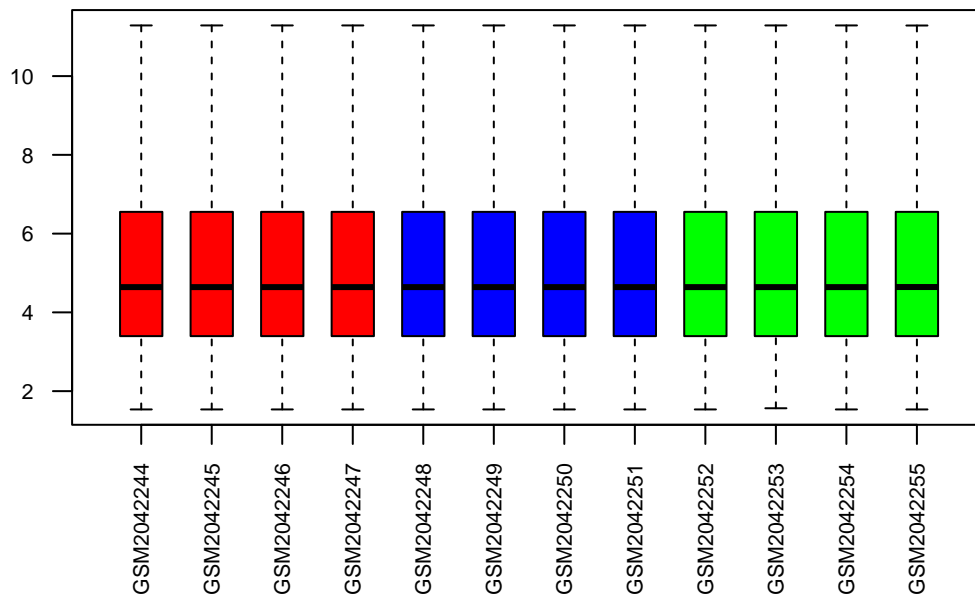
I am a little bit picky and don't like the fat box. I want to change its ratio and remove the outliers.

```
boxplot(exprs(eset), col=scl_groups, las=2, cex.axis = 0.7, boxwex=0.6, outline=FALSE)
```



The boxplot indicates all samples have the similar distribution and are normalized. This indicates there is no need to perform normalization, as normalization was already performed in the data deposited in GEO. In case you need to normalize the data, this can be done with the following commands:

```
# normalizeBetweenArrays is a function in LIMMA
exprs_matrix.norm <- normalizeBetweenArrays(exprs(eset))
boxplot(exprs_matrix.norm, col=sel_groups, las=2, cex.axis = 0.7, boxwex=0.6, outline=FALSE)
```



### 3 Differential Expression Analysis by Limma

Next, we want to find genes, which are differentially expressed between BM-MOs, FL-MOs and YS-Macs using **Limma**. First, we perform a log transformation of the data.

```
# transform into log2 scale
ex <- exprs(eset)
exprs(eset) <- log2(ex)
```

Next, we improve the class labels from the 3 distinct studies and define a design matrices, which indicates conditions to be compared in the DE analysis.

```
# set up the data and proceed with analysis
eset$description <- sel_groups # Replace the long description by short labels
# Construct design matrices by model.matrix
# model.matrix loads the columns of eset and matches to its description.
design <- model.matrix(~ description + 0, eset) # 0 defines the way we compare the samples
colnames(design) <- levels(sel_groups)
```

This is how the design table and array labels will look like.

```
eset$description

## [1] BM_MOs BM_MOs BM_MOs BM_MOs FL_MOs FL_MOs FL_MOs FL_MOs YS_Macs
## [10] YS_Macs YS_Macs YS_Macs
## Levels: BM_MOs FL_MOs YS_Macs

design

##          BM_MOs FL_MOs YS_Macs
## GSM2042244      1      0      0
## GSM2042245      1      0      0
## GSM2042246      1      0      0
## GSM2042247      1      0      0
## GSM2042248      0      1      0
## GSM2042249      0      1      0
## GSM2042250      0      1      0
## GSM2042251      0      1      0
## GSM2042252      0      0      1
## GSM2042253      0      0      1
## GSM2042254      0      0      1
## GSM2042255      0      0      1
## attr("assign")
## [1] 1 1 1
## attr("contrasts")
## attr("contrasts")$description
## [1] "contr.treatment"
```

Finally, we will fit a liner model, perform differential analysis and estimate p-values.

```
# Fit linear model for each gene given a series of arrays
fit <- lmFit(eset, design)
# Build comparison and compute the statistics
cont.matrix <- makeContrasts(BM_MOs-FL_MOs, BM_MOs-YS_Macs, FL_MOs-YS_Macs, levels=design)
# Apply the contrast matrix to the linear model
fit2 <- contrasts.fit(fit, cont.matrix)
# Generate the statistics
fit2 <- eBayes(fit2, 0.05)
```

The makecontrast function indicates which pairs of conditions should be compared (BM\_MOs vs FL\_MOs, BM\_MOs vs YS\_Macs and FL\_MOs vs. YS\_Macs). We can then use the function top table to obtain all DE genes for a given p-value (after FDR multiple test correction) for each of these 3 comparisons.

```
# Extract the sig. DE genes
# coef number represents the order of column in cont.matrix:
# coef=1 BM_MOs-FL_MOs
# coef=2 BM_MOs-YS_Macs
```



```

# coef=3 FL_MOs-YS_Macs
de_genes_BM_MOs_FL_MOs <- topTable(fit2, coef=1, adjust="fdr", p.value=0.05, number=Inf)
de_genes_BM_MOs_YS_Macs <- topTable(fit2, coef=2, adjust="fdr", p.value=0.05, number=Inf)
de_genes_FL_MOs_YS_Macs <- topTable(fit2, coef=3, adjust="fdr", p.value=0.05, number=Inf)

head(de_genes_BM_MOs_FL_MOs) # show top de genes in BM_MO vs FL_MO comparirson.

# Get all the DE genes regardless the comparisons
all_de_genes <- topTable(fit2, adjust="fdr", p.value=0.05, number=Inf)
all_genes_BM_MOs_FL_MOs <- topTable(fit2, coef=1, adjust="fdr", p.value=1, number=Inf)
# These tables will be used later

```

Please take a look into the content of these DE gene lists. Because the table is too big to be shown here, you can open them in Rstudio or output them to excel tables:

```

# Output in an excel file
wb <- createWorkbook()
# Create each sheet
addWorksheet(wb, "de_genes_BM_MOs_FL_MOs")
addWorksheet(wb, "de_genes_BM_MOs_YS_Macs")
addWorksheet(wb, "de_genes_FL_MOs_YS_Macs")
# Write the data into each sheet
writeData(wb, "de_genes_BM_MOs_FL_MOs", de_genes_BM_MOs_FL_MOs, rowNames = FALSE)
writeData(wb, "de_genes_BM_MOs_YS_Macs", de_genes_BM_MOs_YS_Macs, rowNames = FALSE)
writeData(wb, "de_genes_FL_MOs_YS_Macs", de_genes_FL_MOs_YS_Macs, rowNames = FALSE)
saveWorkbook(wb, "DE_genes_GSE76999.xlsx", overwrite = TRUE)

# Output CSV files if necessary
# write.table(de_genes_BM_MOs_FL_MOs, file="de_genes_BM_MOs_FL_MOs.csv", row.names=F, sep="\t")
# write.table(de_genes_BM_MOs_YS_Macs, file="de_genes_BM_MOs_YS_Macs.csv", row.names=F, sep="\t")
# write.table(de_genes_FL_MOs_YS_Macs, file="de_genes_FL_MOs_YS_Macs.csv", row.names=F, sep="\t")

```

Please open the output table *DE\_genes\_GSE76999.xlsx* and try to understand each column.

**Get gene annotation** You probably find that result is not very useful because you cannot recognize any genes by their probe ID (array-specific). What we prefer is the gene symbol, but it is not included in the data. You may think, “can we convert the probe ID to gene symbol?” Yes, this is doable.

Almost every microarray has a particular R package for this mapping task. Here we use package **mogene10sttranscriptcluster.db** which corresponds the Affymetrix mouse genes 1.0 described in GSE76999.

```

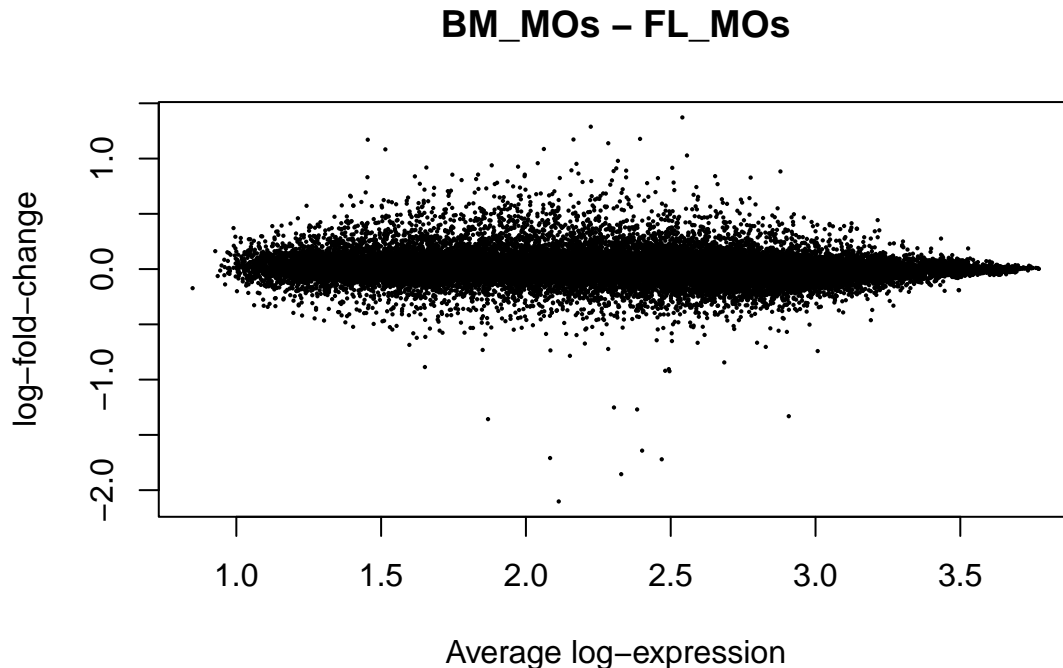
# mogene10sttranscriptcluster.db package contains the annotation of Affymetrix mogene10
# Let's see what is inside this package
columns(mogene10sttranscriptcluster.db)
# Get all the probe ID in our ExpressionSet
ID <- featureNames(eset)
# Select the symbols according to the IDs we just got
tmp <- select(mogene10sttranscriptcluster.db, ID, c("SYMBOL", "ENTREZID"))
tmp <- na.omit(tmp)
# Then we can insert a column for symbols which match the ID for each row
all_genes_BM_MOs_FL_MOs$ENTREZID <- tmp$ENTREZID[match(rownames(all_genes_BM_MOs_FL_MOs), tmp$PROBEID)]
all_de_genes$symbol <- tmp$SYMBOL[match(rownames(all_de_genes), tmp$PROBEID)]
all_de_genes <- na.omit(all_de_genes)
de_genes_BM_MOs_FL_MOs$ENTREZID <- tmp$ENTREZID[match(rownames(de_genes_BM_MOs_FL_MOs), tmp$PROBEID)]
de_genes_BM_MOs_YS_Macs$ENTREZID <- tmp$ENTREZID[match(rownames(de_genes_BM_MOs_YS_Macs), tmp$PROBEID)]
de_genes_FL_MOs_YS_Macs$ENTREZID <- tmp$ENTREZID[match(rownames(de_genes_FL_MOs_YS_Macs), tmp$PROBEID)]

```

What if I have data from a distinct array? For example, you find the array name as **Affymetrix Human Gene 2.0 ST Array [transcript (gene) version]** in the given GEO description. What you can do is google this name with **R bioconductor** to see what package is available for annotation conversion. Or, you can visit the webpage of Bioconductor: [https://bioconductor.org/packages/release/BiocViews.html#\\_\\_\\_ChipManufacturer](https://bioconductor.org/packages/release/BiocViews.html#___ChipManufacturer) where the list of Packages for all ChipManufacturer is found.

**MA plot** The MA plot visualizes the differences between measurements taken in two samples, by transforming the data into M (log ratio) and A (mean average) scales, then plotting these values. Here we take **BM-MOs** v.s. **FL\_MOs** as an example.

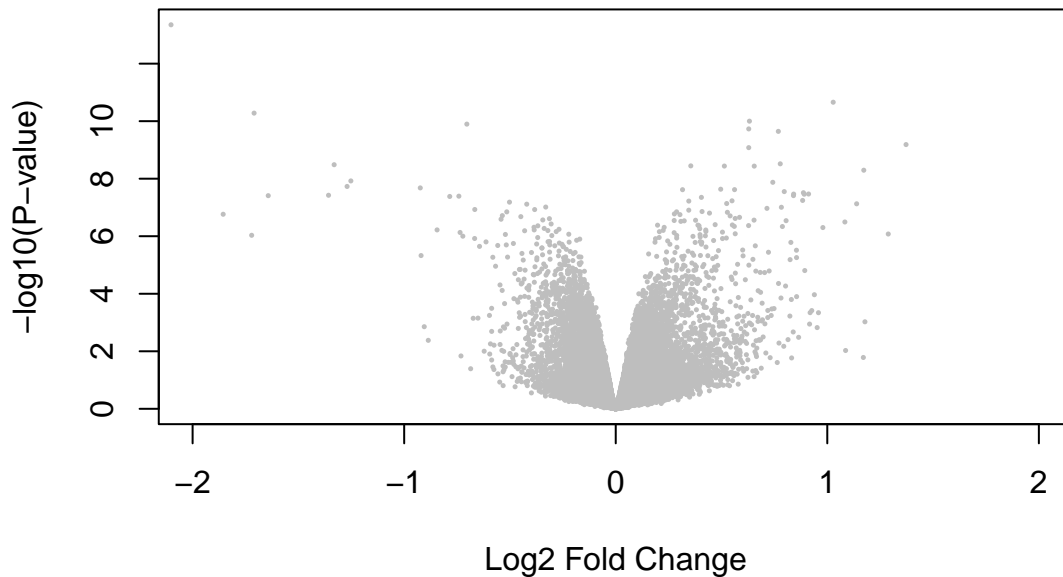
```
# By Limma build-in function
ma <- fit2[, "BM_MOs - FL_MOs"]
limma::plotMA(ma)
```



**Volcano plot** A volcano plot is a type of scatter-plot that is used to quickly identify the significance of changes in large data sets. It plots p-values versus fold-change.

```
{volcanoplot(fit2, coeff=1, xlim=c(-2,2), col="gray")
title("BM_MOs - FL_MOs")}
```

## BM\_MOs – FL\_MOs



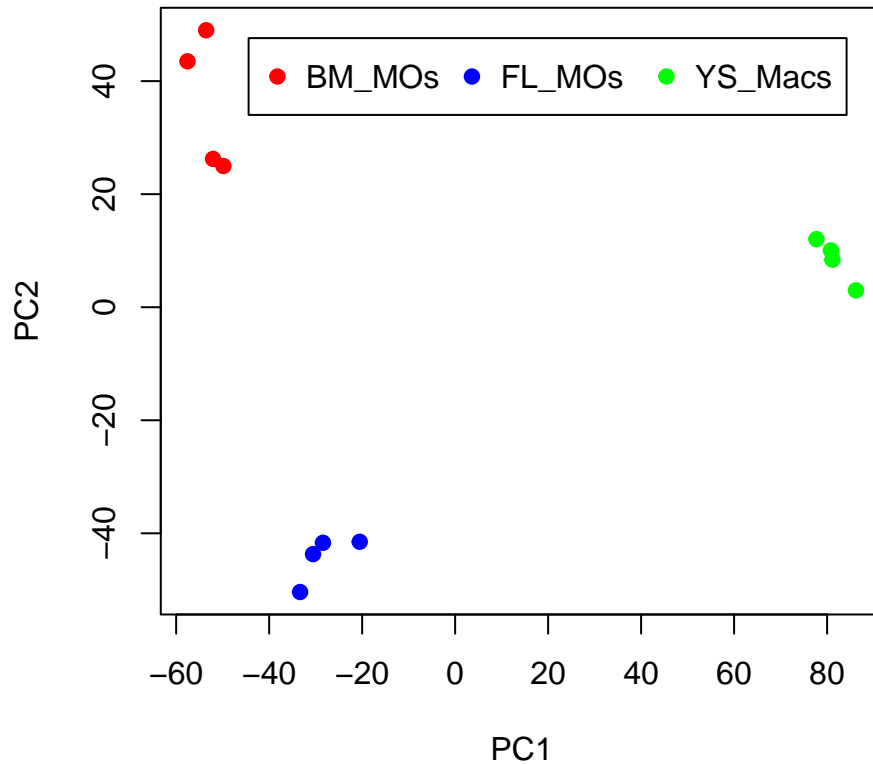
You can also save any figure into PDF file by the codes below:

```
pdf("volcanoplot.pdf", width=4, height=4) # Save the plot into PDF format
{volcanoplot(fit2, coeff=1, xlim=c(-2,2), col="gray")
title("BM_MOs – FL_MOs")}
dev.off() # close the graphical device
```

## 4 PCA (Principal component analysis)

Principal component analysis (PCA) is a mathematical algorithm that reduces the dimensionality of the data while retaining most of the variation in the data set<sup>1</sup>. It accomplishes this reduction by identifying directions, called principal components, along which the variation in the data is maximal. By using a few components, each sample can be represented by relatively few numbers instead of by values for thousands of variables. Samples can then be plotted, making it possible to visually assess similarities and differences between samples and determine whether samples can be grouped (see Nature Biotechnology 26, 303 - 304 (2008) “What is principal component analysis?”).

```
# Performs a principal components analysis on the given data matrix
pca <- prcomp(t(ex))
# Take PC1 and PC2 for the plot
{plot(pca$x[,1:2], col=scl_groups, pch=19)
# include a legend for points
legend("topright", inset=.05, labels, pch=19, col=1:3, horiz=TRUE)}
```



## 5 Clustering by Heatmap

Another way to visualize the similarity and the difference of the samples is to perform hierarchical clustering and plot results as a heatmap.

First, we will create a color maps for generating values from blue (low expression) to red (high expression). Try out this command (`display.brewer.all()`) to check other pallets. We also restrict the matrix to only show differentially expressed genes.

```
# ex is the matrix for all the expression values
# all_de_genes is all the DE genes
# Let's filter ex by the list in all_de_genes
sel <- match(rownames(all_de_genes), rownames(ex)) # Get the index of DE genes in ex
sel_ex <- ex[sel,] # Select those rows only
```

Next, we create a few functions defining the use of Pearson correlation as a distance matrix and complete linkage for cluster.

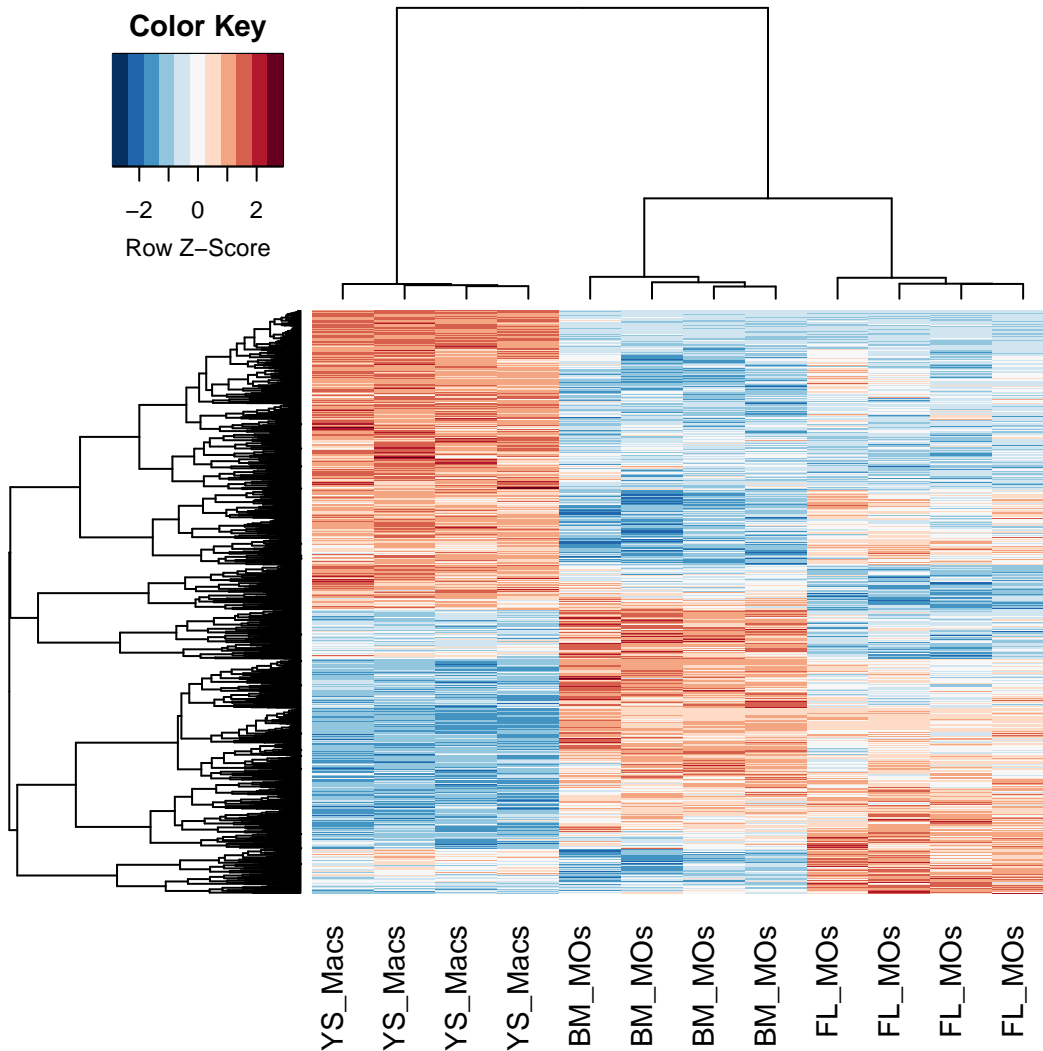
```
clust <- function(x) hclust(x, method="complete") # perform complete linkage clustering
dist <- function(x) as.dist((1-cor(t(x)))/2) # use the inverse of correlation as distance.
hmcol <- rev(brewer.pal(11, "RdBu"))

# heatmap.2 is a powerful and complicated function, please read its help message
hp <- heatmap.2(sel_ex, # input matrix
  col=hmcol, # define the customized colormap
  scale="row", # scale the values for each gene (row)
  labRow=F, # Not showing row labels
  density.info="none", # Not showing density info
  trace="none", # Not showing trace line
  margins = c(8,3), # modify the margin to fit the labels
```

```

labCol = rep(labels,each=4), # add the column labels
# define hierarchical clustering method
hclust = clust,
# Using correlation coefficient for distance function
distfun = dist
)

```



This is a pretty heatmap, or? As you noticed, this involves calling a complex function with several parameters. Play around with parameters: changing the colormap, linkage function of the clustering and see what you get. Also notice that computing this takes a few minutes and obtaining a nice looking heatmap can involve a lot of work on parameters.

**Get genes from clustering** If we want to get the clusters from the dendrogram on the left side, we can use the script below:

```

# Extract the hierarchical cluster from heatmap to class "hclust"
hc <- as.hclust(hp$rowDendrogram)
# Cut the tree by the dedired number of groups
clusters <- cutree(hc, k=5)
# Converting the ID into gene symbol into a list. Please try to understand this code step by step...
clustered_genes <- list(all_de_genes$symbol[match(names(clusters)[clusters==1],all_de_genes$ID)],

```

```

all_de_genes$symbol[match(names(clusters)[clusters==2],all_de_genes$ID)],
all_de_genes$symbol[match(names(clusters)[clusters==3],all_de_genes$ID)],
all_de_genes$symbol[match(names(clusters)[clusters==4],all_de_genes$ID)],
all_de_genes$symbol[match(names(clusters)[clusters==5],all_de_genes$ID)])

# Alternative way to do it
clustered_genes <- list(all_de_genes[names(clusters[clusters==1]),]$symbol,
                        all_de_genes[names(clusters[clusters==2]),]$symbol,
                        all_de_genes[names(clusters[clusters==3]),]$symbol,
                        all_de_genes[names(clusters[clusters==4]),]$symbol,
                        all_de_genes[names(clusters[clusters==5]),]$symbol)

# Output in an excel file
wb <- createWorkbook()
addWorksheet(wb, "cluster 1")
addWorksheet(wb, "cluster 2")
addWorksheet(wb, "cluster 3")
addWorksheet(wb, "cluster 4")
addWorksheet(wb, "cluster 5")
writeData(wb, "cluster 1", clustered_genes[[1]], rowNames = FALSE)
writeData(wb, "cluster 2", clustered_genes[[2]], rowNames = FALSE)
writeData(wb, "cluster 3", clustered_genes[[3]], rowNames = FALSE)
writeData(wb, "cluster 4", clustered_genes[[4]], rowNames = FALSE)
writeData(wb, "cluster 5", clustered_genes[[5]], rowNames = FALSE)
saveWorkbook(wb, "gene_clusters.xlsx", overwrite = TRUE)

```

**Exercise Idea:** try to reimplement the previous code with a loop so that you can also evaluate results with distinct number of clusters.

## 6 GO (Gene Ontology) Enrichment Analysis

Next, we will do a GO analysis to check GO terms associated to each of the DE gene sets. Here we will look at genes up- or down-regulated in the comparison **BM-MOs** (G0) v.s. **YS-Macs** (G2).

```

go_enrich_up <- gprofiler(as.vector(all_de_genes$symbol[all_de_genes$BM_MOs...YS_Macs > 0]),
                        organism="mmusculus")
go_enrich_down <- gprofiler(as.vector(all_de_genes$symbol[all_de_genes$BM_MOs...YS_Macs < 0]),
                        organism="mmusculus")

head(go_enrich_up)

```

```

##  query.number significant  p.value term.size query.size overlap.size precision
## 1             1         TRUE 5.64e-03      78      1903         21      0.011
## 2             1         TRUE 8.19e-03     126      1903         28      0.015
## 3             1         TRUE 3.06e-03      94      1903         24      0.013
## 4             1         TRUE 2.09e-02    2913      1903        316      0.166
## 5             1         TRUE 9.90e-09     661      1903        112      0.059
## 6             1         TRUE 4.31e-07     593      1903         99      0.052
##  recall      term.id domain subgraph.number
## 1  0.269 GO:0050764    BP         127
## 2  0.222 GO:0001906    BP          81
## 3  0.255 GO:0001909    BP          81
## 4  0.108 GO:0051239    BP         139
## 5  0.169 GO:0001816    BP         139
## 6  0.167 GO:0001817    BP         139

```

```
##               term.name relative.depth
## 1      regulation of phagocytosis         1
## 2                cell killing         1
## 3      leukocyte mediated cytotoxicity     2
## 4 regulation of multicellular organismal process 1
## 5                cytokine production     1
## 6      regulation of cytokine production     2
##
## 1
## 2
## 3
## 4 CLEC2G, ITGB2, BCL11A, RARG, DEF8, TIAM1, AXL, IL17RA, APOC2, CDKN1B, KLF4, PTPN6, LY9, MAP2K1, INPP5K, SULF2, RHO
## 5
## 6
```

Then, of course, we can save these tables into Excel files.

```
# Output in an excel file
wb <- createWorkbook()
addWorksheet(wb, "GO_BM_MOs_YS_Macs_UP")
addWorksheet(wb, "GO_BM_MOs_YS_Macs_DOWN")
writeData(wb, "GO_BM_MOs_YS_Macs_UP", go_enrich_up, rowNames = FALSE)
writeData(wb, "GO_BM_MOs_YS_Macs_DOWN", go_enrich_down, rowNames = FALSE)
saveWorkbook(wb, "GO_analysis.xlsx", overwrite = TRUE)
```

## 7 GSEA

What should we do if we wish to test a specific (maybe self-defined) gene set, to see whether they are up-regulated or down-regulated compared to other genes in terms of differential expression?

Then here is the Gene set enrichment GSEA.

We assume that the first 20 genes are a set of genes interested. We use camera from limma to see whether it is going up or down.

```
# assume the set of first 20 genes is the interested gene set
interested_gene_set <- rownames(exprs(eset))[1:20]

# Do the GSEA on limma
res.gsea <- camera(exprs(eset), list(set1 = interested_gene_set), design)
print(res.gsea)
```

```
##      NGenes Direction    PValue
## set1      20         Up 0.2180146
```

We can also take use of previous found gene set from Database MSigDB. (see MSigDB homepage). Now we import the NABA geneset from MSigDB.

```
m_df = msigdbr(species = "Mus musculus", category = "C2", subcategory = "CP")

naba <- m_df[grep("NABA", m_df$gs_name), ]

naba.1 <- list()
for(i in unique(naba$gs_name)){
  sub <- naba[naba$gs_name == i, ]
  sub <- as.data.frame(sub)
  naba.1[[i]] <- as.character(sub$gene_symbol)
```

```
}
```

```
# Now we show what genes are inside the NABA_CORE_MATRISOME geneset  
naba.l$NABA_CORE_MATRISOME
```

```
## [1] "Abi3bp" "Acan" "Adipoq" "Aebp1" "Agrn" "Ambn"  
## [7] "Amelx" "Amelx" "Aspn" "Bcan" "Bglap" "Bglap2"  
## [13] "Bglap3" "Bgn" "Bmper" "Bsph1" "Ccn1" "Ccn2"  
## [19] "Ccn3" "Ccn4" "Ccn5" "Ccn6" "Cdc2" "Chad"  
## [25] "Chad1" "Cilp" "Cilp2" "Coch" "Col10a1" "Col11a1"  
## [31] "Col11a2" "Col12a1" "Col13a1" "Col14a1" "Col15a1" "Col16a1"  
## [37] "Col17a1" "Col18a1" "Col19a1" "Col1a1" "Col1a2" "Col20a1"  
## [43] "Col22a1" "Col23a1" "Col24a1" "Col25a1" "Col26a1" "Col27a1"  
## [49] "Col28a1" "Col2a1" "Col3a1" "Col4a1" "Col4a2" "Col4a3"  
## [55] "Col4a4" "Col4a5" "Col4a6" "Col5a1" "Col5a2" "Col5a3"  
## [61] "Col6a1" "Col6a2" "Col6a3" "Col6a5" "Col6a6" "Col7a1"  
## [67] "Col8a1" "Col8a2" "Col9a1" "Col9a2" "Col9a3" "Colq"  
## [73] "Comp" "Creld1" "Creld2" "Crim1" "Crispld1" "Crispld2"  
## [79] "Cthrc1" "Dcn" "Dmbt1" "Dmp1" "Dpt" "Dspp"  
## [85] "Ecm1" "Ecm2" "Edil3" "Efemp1" "Efemp2" "Egflam"  
## [91] "Eln" "Emid1" "Emilin1" "Emilin2" "Emilin3" "Epyc"  
## [97] "Esm1" "Fbln1" "Fbln2" "Fbln5" "Fbln7" "Fbn1"  
## [103] "Fbn2" "Fga" "Fgb" "Fgg" "Fgl1" "Fgl2"  
## [109] "Fmod" "Fn1" "Fndc1" "Fndc7" "Fndc8" "Fras1"  
## [115] "Gas6" "Gldn" "Hapln1" "Hapln2" "Hapln3" "Hapln4"  
## [121] "Hmcn1" "Hmcn2" "Hspg2" "Ibsp" "Igfals" "Igfbp1"  
## [127] "Igfbp2" "Igfbp3" "Igfbp4" "Igfbp5" "Igfbp6" "Igfbp7"  
## [133] "Igfbp11" "Igsf10" "Impg1" "Impg2" "Ints14" "Ints61"  
## [139] "Kcp" "Kera" "Lama1" "Lama2" "Lama3" "Lama4"  
## [145] "Lama5" "Lamb1" "Lamb2" "Lamb3" "Lamc1" "Lamc2"  
## [151] "Lamc3" "Lgi1" "Lgi2" "Lgi3" "Lgi4" "Lrg1"  
## [157] "Ltbp1" "Ltbp2" "Ltbp3" "Ltbp4" "Lum" "Matn1"  
## [163] "Matn2" "Matn3" "Matn4" "Mepe" "Mfap1a" "Mfap2"  
## [169] "Mfap3" "Mfap4" "Mfap5" "Mfge8" "Mgp" "Mmrn1"  
## [175] "Mmrn2" "Ncan" "Ndnf" "Nell1" "Nell2" "Nid1"  
## [181] "Nid2" "Npnt" "Ntn1" "Ntn3" "Ntn4" "Ntn5"  
## [187] "Ntng1" "Ntng2" "Nyx" "Ogn" "Oit3" "Omd"  
## [193] "Optc" "Otog" "Otol1" "Papln" "Pcolce" "Pcolce2"  
## [199] "Podn" "Podnl1" "Zp3" "Postn" "Prep" "Prg2"  
## [205] "Prg3" "Prg4" "Pxdn" "Reln" "Rspo1" "Rspo2"  
## [211] "Rspo3" "Rspo4" "Sbspon" "Slit1" "Slit2" "Slit3"  
## [217] "Smoc1" "Smoc2" "Sned1" "Sparc" "Sparcl1" "Spock1"  
## [223] "Spock2" "Spock3" "Spon1" "Spon2" "Spp1" "Srgn"  
## [229] "Srp" "Srp2" "Sspo" "Svep1" "Tecta" "Tectb"  
## [235] "Tgfbi" "Thbs1" "Thbs2" "Thbs3" "Thbs4" "Thsd4"  
## [241] "Tinag" "Tinagl1" "Tnc" "Tnfaip6" "Tnn" "Tnr"  
## [247] "Tnxb" "Tsku" "Tspear" "Ush2a" "Vcan" "Vit"  
## [253] "Vtn" "Vwa1" "Vwa2" "Vwa3a" "Vwa3b" "Vwa5a"  
## [259] "Vwa5b1" "Vwa5b2" "Vwa7" "Vwce" "Vwde" "Vwf"  
## [265] "Zp1" "Zp2" "Zp3" "Zpld1"
```

Quiz:

1. How to get the full geneset of HALLMARK\_GLYCOLYSIS?
2. How to test all gene sets from MSigDB



## 8 Combining other dataset for PCA and clustering

Finally, as an advanced example of data combination, we have selected Monocyte and Macrophage populations from the ImmGenn data (<https://www.immgenn.org/>).

For simplicity, we will load the pre-processed data and a table containing details on the experiments (pheno data). To make these data comparable, we need to remove the batch effects caused by different labs, by using a tool called combat. Here, we are particularly interested in contrasting the expression patterns of Monocytes and Macrophages from distinct organs.

```
# get imm gene data from saved object
ex_imun_gen <- dget("/home/vkraemer/Documents/BIR_course/Imun_gen.ex.Rdump")
# get all pheno data from tab sep. file
phenoData_table <- read.table(file="/home/vkraemer/Documents/BIR_course/pheno.table.csv", header=T, sep="
")

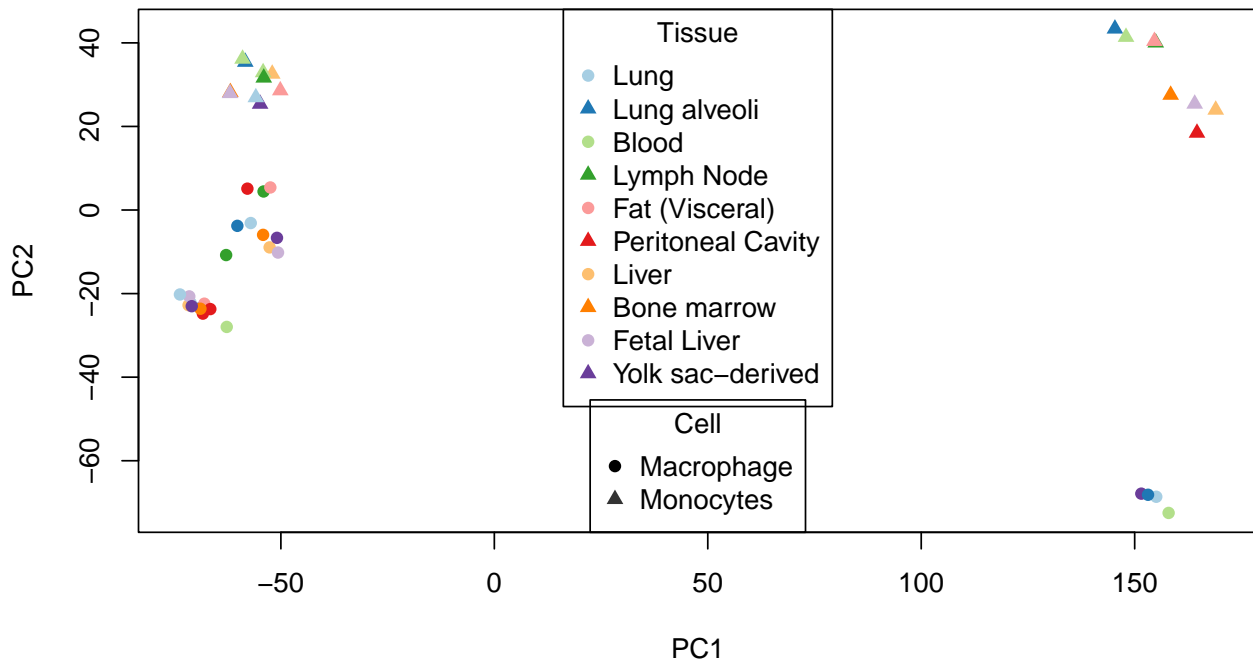
phenoData_table # see the content of the phenotype table.

# combine two expression matrices
ex_combined.raw <- merge(ex_imun_gen, ex, by="row.names")
rownames(ex_combined.raw) <- ex_combined.raw$"Row.names"
ex_combined.raw$"Row.names" <- NULL
ex_combined.raw <- as.matrix(ex_combined.raw)
```

**PCA on merged data** Once the data has been combined, we can perform PCA and clustering analysis in both data sets. Here tissues are represented by distinct colors and cell types by distinct symbols.

```
pca_im <- prcomp(t(ex_combined.raw)) # perform PCA
exp_names <- paste0(phenoData_table$cell, "_", phenoData_table$Tissue) # combine cell and tissue data.
# Take PC1 and PC2 for the plot
#par(mai = c(1, 1, .25, 2))
{plot(pca_im$x[,1:2],
      pch = ifelse(phenoData_table$cell == "Macrophage", 16, 17), #c(16, 17)[as.numeric(phenoData_table$cell)],
      col = brewer.pal(length(unique(phenoData_table$Tissue)), "Paired")) # brewer.pal(10, "Set1")[as.numeric(phenoData_table$Tissue)]
# Legend outside the plot on right
legend("top", # Location of legend
      xpd = TRUE, # Allow drawing outside plot area
      xjust = 0, # Left justify legend box on x
      yjust = 1, # Center legend box on y
      legend = unique(phenoData_table$Tissue),
      col = brewer.pal(length(unique(phenoData_table$Tissue)), "Paired"), # Legend Element Colors
      pch = c(16,17), # Legend Element Styles
      title = "Tissue") # Legend Title

# Legend outside the plot on bottom
legend("bottom", # Location of legend
      xpd = TRUE, # Allow drawing outside plot area
      xjust = 0, # Left justify legend box on x
      yjust = 0, # Center legend box on y
      legend = unique(phenoData_table$cell),
      col = c("black", gray(.2)), # Legend Element Colors
      pch = c(16, 17), # Symbol styles for legend elements
      title = "Cell") # Legend Title
}
```



This PCA plot is clearly separated into 2 parts, each representing the two data sets. This is known as batch effect. Because we combine two different datasets, their different background, condition, or procedure may result in this separation. In order to remove it, we use **ComBat** function from package **sva** to remove it.

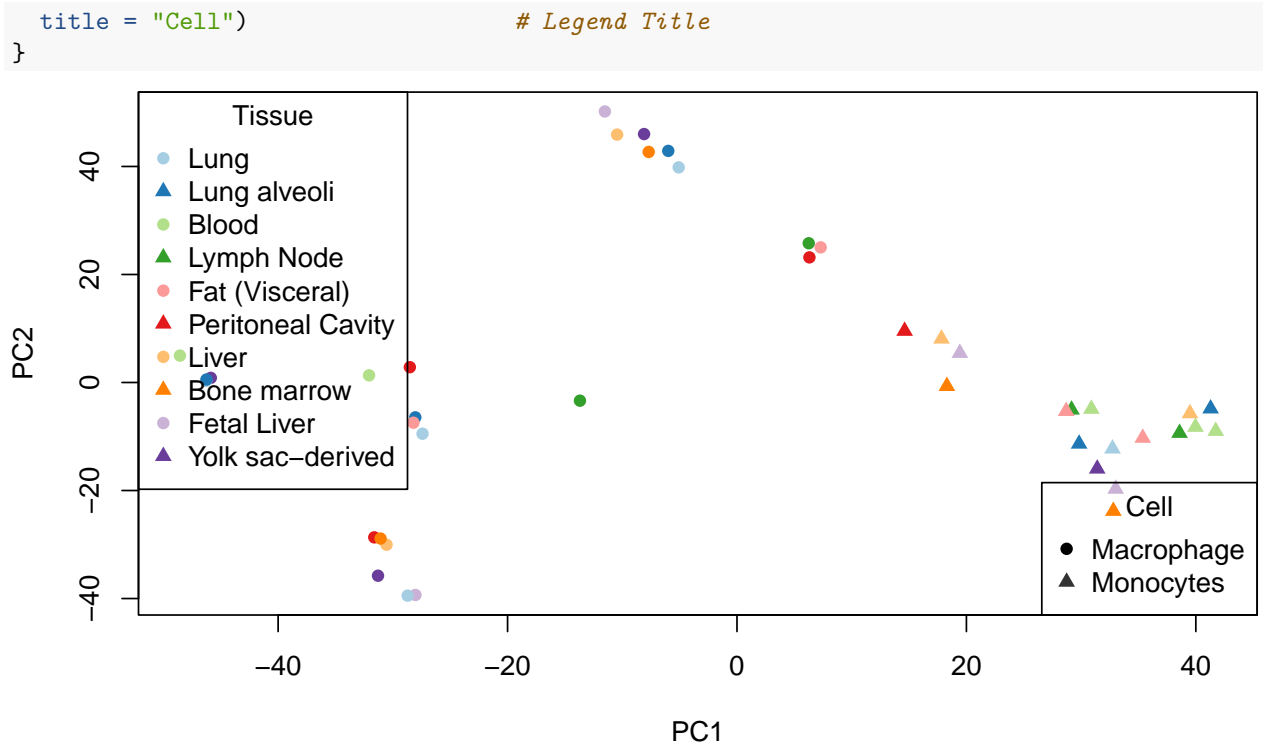
```
modcombat = model.matrix(~1, data=phenoData_table)
ex_combined.batch_removed = ComBat(dat=ex_combined.raw,
                                   batch=phenoData_table$batch, mod=modcombat, par.prior=TRUE)
```

Let's check how the PCA looks like after this:

```
pca_im <- prcomp(t(ex_combined.batch_removed)) # pca analysis of batch removed matrix

# Take PC1 and PC2 for the plot
plot(pca_im$x[,1:2],
     pch = ifelse(phenoData_table$cell == "Macrophage", 16, 17), #c(16, 17)[as.numeric(phenoData_table$cell)],
     col = brewer.pal(length(unique(phenoData_table$Tissue)), "Paired")) # brewer.pal(10, "Set1")[as.numeric(unique(phenoData_table$Tissue))],
# Legend outside the plot on right
legend("topleft", # Location of legend
      xpd = TRUE, # Allow drawing outside plot area
      xjust = 0, # Left justify legend box on x
      yjust = 1, # Center legend box on y
      legend = unique(phenoData_table$Tissue),
      col = brewer.pal(length(unique(phenoData_table$Tissue)), "Paired"), # Legend Element Colors
      pch = c(16, 17), # Legend Element Styles
      title = "Tissue") # Legend Title

# Legend outside the plot on bottom
legend("bottomright", # Location of legend
      xpd = TRUE, # Allow drawing outside plot area
      xjust = 0, # Left justify legend box on x
      yjust = 0, # Center legend box on y
      legend = unique(phenoData_table$cell),
      col = c("black", gray(.2)), # Legend Element Colors
      pch = c(16, 17), # Symbol styles for legend elements
```



Now, we see that similar cells (and tissue of origin) cluster apart.

**Clustering of merged data** Alternatively, we can also cluster the gene expression profiles.

```

# Select the de genes from the expression matrix after batch effect removal
sel <- match(rownames(all_de_genes), rownames(ex_combined.batch_removed))
sel <- sel[!is.na(sel)] # remove the NAs

clust <- function(x) hclust(x, method="complete") # perform complete linkage clustering
dist <- function(x) as.dist((1-cor(t(x)))/2) # use the inverse of correlation as distance.
hmcol <- rev(brewer.pal(11, "RdBu"))

# heatmap.2 is a powerful and complicated function, please read its help message
hp <- heatmap.2(ex_combined.batch_removed[sel,], # input matrix
  col=hmcol, # define the customized colormap
  scale="row", # scale the values for each gene (row)
  labRow=F, # Not showing row labels
  density.info="none", # Not showing density info
  trace="none", # Not showing trace line
  margins = c(11,3), # modify the margin to fit the labels
  labCol=paste0(phenoData_table$cell, " / ", phenoData_table$Tissue), # add the column labels
  # define hierarchical clustering method
  hclust = clust,
  # Using correlation coefficient for distance function
  distfun = dist
)

```

