Statistical Analysis of Microarray data

Based on Gonzalo, Ricardo and Sanchez-Pla, Alex (2019)

March 29, 2020

1 Summary/Abstract

Microarray data analysis has been one of the most important hits in the interaction between statistics and bioinformatics in the last two decades. The analysis of microarray data can be done in different ways using different tools. In this chapter a typical workflow for analyzing microarray data using R and Bioconductor packages is presented. The workflow starts with the raw data -binary files obtained from the hybridization process- and goes through a series of steps: Reading raw data, Quality Check, Normalization, Filtering, Selection of differentially expressed genes, Comparison of selected lists and Analysis of Biological Significance. The implementation of each step in R is described through a use

case that goes from raw data until the analysis of biological significance. Data and code for the analysis are provided in a github repository[1].

1.1 Key Words

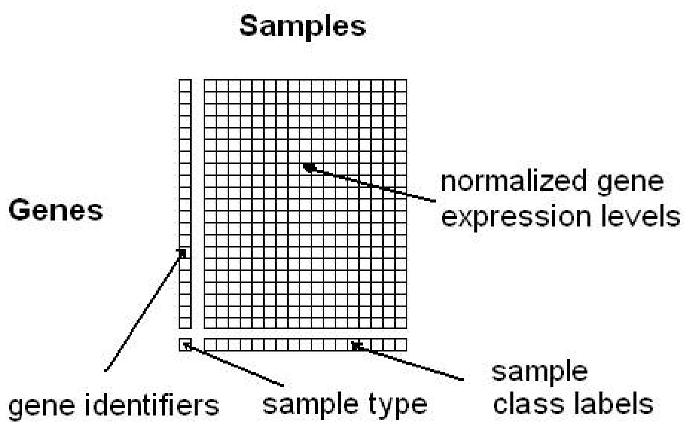
Microarrays, Bioconductor, R, Differential Expression

2 Introduction

Microarray data analysis is one of the clearest cases where interaction between bioinformatics and statistics has been highly beneficial for both disciplines. Efron Efron (2013) even calls the 21st century as the century of microarrays.

What is generically described as “microarray data analysis” is a process that starts with the design of the experiment intended to answer with one or more biological questions and ends with a tentative answer for these questions. Statistics is involved at every step of this process, for preparing, transforming visualizing or analyzing data. And, of course, every step can be done in different way that use either classical statistics or new methods developed ad-hoc for these often high dimensional problems. The detailed description of these steps is out of the scope of this chapter and the reader is assumed to be familiar with them. It is assumed that the reader is already familiar with microarrays such as they are introduced in Sánchez-Pla (2014) and also with the general ideas of microarray data analysis such as can be found in Draghici (2012). In any case, for the sake of completeness basic ideas will be briefly introduced and citations provided the first time they are discussed.

For our objectives we can assume that a microarray dataset is a matrix of continuous values that represent the expressions of a set of genes (one gene per row), in a variety of conditions or samples (one sample per column). See figure @ref(fig:dataset) for an example.



A simplified view of a gene expression matrix

Note that we have described the row contents as “genes”. Strictly speaking depending on the type of array, each row may correspond to one distinct, but related, entity, a “probeset” or a “transcript”.

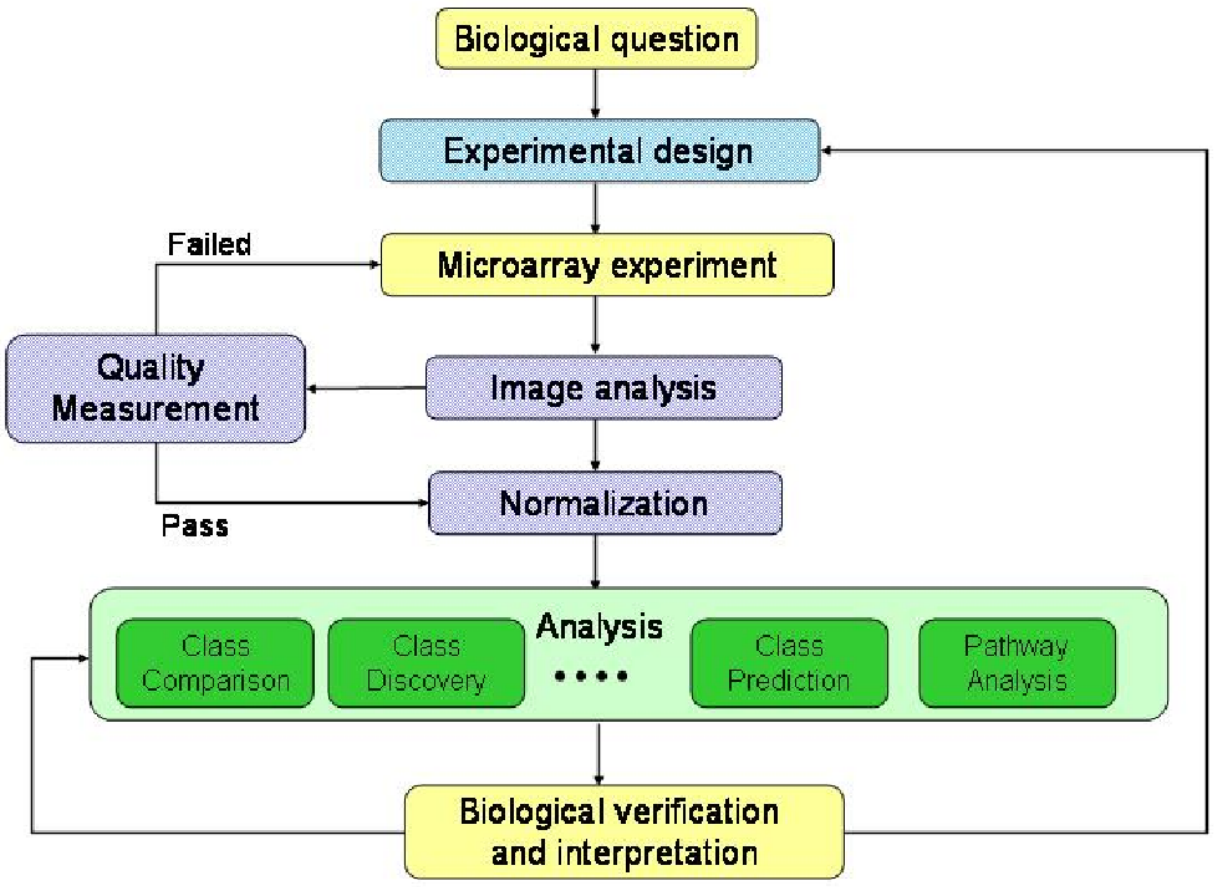
- A transcript describes how the gene has been transcribed into messenger RNA. If transcription was unique there would be a single transcript per gene. However, due to the phenomenon of alternative splicing, Sánchez-Pla et al. (2012), there may be different transcriptions of the same gene (the associated proteins are called “isoforms”). That there may be multiple transcripts per gene.
- A probeset is, as indicated by its name, a set of “probes”, which are designed to map different fragments of a given gene. Altogether it is expected that each set of probes, or probeset, uniquely characterizes one gene. However, given that this characterization is not always possible it may be convenient to have more than one probeset per gene. That is although it is common to

exchange the terms “probeset” and “gene”, it is important to be aware that there may be severalprobestes per each gene.

In practice, given that either probesets or transcripts map to genes, it is common to describe the array rows as “genes”.

Our main goal is to describe a workflow, a series of ordered steps that takes us from the raw data, the digitized images as produced by the hybridization system, to one or more lists of genes that can be used to help answering a certain biological question. This can be done in distinct ways. What we present here is an approach that has become very popular along the last decades based on analyzing the data from the images to the lists of genes, using the R Statistical language and some of the packages developed specifically for this in the Bioconductor project.

A summary of the process can be found in figure @ref(fig:MDAProcess).



The microarray data analysis process

3 Materials

In this section we list all the materials needed to perform a microarray data analysis. The document is adapted from @Gonzalo Sanz and Sánchez-Pla (2019), by the same authors.

3.1 Software

First of all it is needed to install the software to perform all the required calculations.

There are many options are available Mehta and Rani (2011) but one of the most common approaches is to use the **R statistical software**. R can be downloaded from its web page (https://cran.r-project.org/index.html) and installed following the instructions described there. The microarray analysis presented in this chapter has been performed with the latest version of R which, at the moment of writing, was 3.4.4.

R is a console based software. Its use can be facilitated with an additional interface called “R-Studio”. It can be downloaded and installed following the instructions listed in its web page: https://www.rstudio.com/. Although its use is not compulsory for reproducing the analyses in this chapter it is highly recommended to work with R using this interface.

When working with R it is often required to use some functions not available in the basic installation. This can be done by installing additional libraries, also called “packages”, developed by the scientific community. Most packages used for the analysis of high throughput genomic data are part of the Bioconductor project which started with a few packages in 2002 and has now more than one thousand (https://www.bioconductor.org/). Indeed Bioconductor has become the state-of-the-art way to analyze microarray and other omics data and it has grown from hardly a dozen packages in 2002 to the current number of more than one thousand. The analysis presented have been performed using Bioconductor version 3.6.

R and Bioconductor are open source free software. This has many advantages but it may sometimes be a problem, especially when new functionalities are not compatible with previous versions (see Note 1).

Table @ref(tab:packagesList) shows the packages needed for the analysis presented in this chapter. The table contains the name, the source and a short description of all the packages that have to be install to run the current case study. In the following section we will show the code necessary to install all of them.

3.2 Data

This protocol is applied on a dataset from a published study Li et al. (2017). The data had been uploaded into the Gene Expression Omnibus (GEO) database, an international public repository that archives and freely distributes high-throughput gene expression and other functional genomics data sets Clough and Barrett (2016). The dataset selected is identified with the accession number: **GSE100924**.

The study that generated the data investigated the function of gene `ZBTB7B` (http://www.genecards.org/cgi-bin/carddisp.pl?gene=ZBTB7B). This gene activates the thermogenic gene program during brown and beige adipocyte differentiation regulating brown fat gene expression at ambient room temperature and following cold exposure. The experiment compared 10 weeks old mice with the gene deactivated (“KO” or knockout) or not (“WT” or Wild type) at two different temperatures, ambient room temperature (RT, 22°C) or following cold exposure (COLD, 4°C) for 4 hrs. That is, it was a 2×2 factorial design (genotype and temperature) with two levels each (wild type and knock out, for genotype, and room temperature and cold, for temperature). The sample size of the experiment is 12 samples, three replicates of each group.

The microarrays used for this experiment were of type Mouse Gene 2.1 from Affymetrix, now Thermofisher, one of the most popular vendors of microarray technology.

4 Methods

4.1 Environment preparation:

In a microarray data analysis project, data analyst will have to manage a lot of files, including the files with the raw data (.CEL files) and the files generated during the analysis of them. For this reason, it is very advisable to define some folders before beginning with the analysis to try to not get lost, if all the files go to the same folder. We strongly recommend that user creates the following folders:

- A main folder that will be the "working directory" called for example "MicroarraysAnalysis"
- A folder called, for example **data** located within the working directory: Here we will save all the *.CEL* files and the *targets* file with information on the covariates, described in next section.
- A folder called, for example **results** located within the working directory: Here we will send all the results obtained in the microarray analysis.

The following commands create the desired folders. This can be made from within R as described, or using a visual file browser such as Windows File Explorer or any other (in that case you can omit this step):

```
> setwd(".")
> dir.create("data")
> dir.create("results")
```

The code for running this analysis, that can be easily adapted to run similar studies can be downloaded from a github repository specifically devoted to this chapter. In the following sections it is assumed that such code has been downloaded and copied into the working directory. The url for the repository is: https://github.com/alexsanchezpla/StatisticalAnalysisOfMicroarrayData.

Once it is saved, open R-Studio, open the file with the R code, and in R-Studio go to the menu option in the top "Session -> Set working directory -> To source file location". This action will set the folder we have set as "main" folder as our working directory

4.2 Prepare the data for the analysis

The data for the analysis will be provided as two types of files, the "CEL"" files and the "targets" file.

CEL files are the files with the "raw data" originated after microarray scanning and preprocessing with Affymetrix software. This files need to be saved into the **data** folder. Usually one expectes to have a .CEL file for each sample in the experiment.

Another file needed for the analysis is the *targets* file, which contains the information on groups and covariates. That is, this file relates the name of each .CEL file with their condition in the experiment. We can use the targets to retain all the information valuable for the analysis like other covariables.

Although the targets file need not have any fixed names it is practical to use its column names to create labels that will be used later. For example:

- Column called *FileName*: It may contain the exact name of the CEL files in the data folder
- Column called *Group*: It may summarize the conditions in the experiment for that sample.
- Column called *ShortName*: It may be used to store a short label of the sample useful for some plots.
- There may be other columns to store covariables in the study such as sex, age, etc.

For this analysis, targets file has been saved in .csv format, separated by semicolon, although any other format that works for delimited text files might have been used (see Note 2). Table @ref(tab:ReadTargets) shows the contents of the targets file used in this analysis.

```
> targets <- read.csv2("./data/targets.csv", header = TRUE, sep = ";")
> knitr::kable(
+   targets, booktabs = TRUE,
+   caption = 'Content of the targets file used for the current analysis')
```

Content of the targets file used for the current analysis

FileName	Group	Genotype	Temperature	ShortName
GSM2696488_WT_RT_1.CEL	WT.RT	WT	RT	WT.RT.1
GSM2696489_WT_RT_2.CEL	WT.RT	WT	RT	WT.RT.2
GSM2696490_WT_RT_3.CEL	WT.RT	WT	RT	WT.RT.3
GSM2696491_KO_RT_1.CEL	KO.RT	KO	RT	KO.RT.1
GSM2696492_KO_RT_2.CEL	KO.RT	KO	RT	KO.RT.2
GSM2696493_KO_RT_3.CEL	KO.RT	KO	RT	KO.RT.3
GSM2696494_WT_Cold_1.CEL	WT.COLD	WT	COLD	WT.COLD.1
GSM2696495_WT_Cold_2.CEL	WT.COLD	WT	COLD	WT.COLD.2
GSM2696496_WT_Cold_3.CEL	WT.COLD	WT	COLD	WT.COLD.3
GSM2696497_KO_Cold_1.CEL	KO.COLD	KO	COLD	KO.COLD.1
GSM2696498_KO_Cold_2.CEL	KO.COLD	KO	COLD	KO.COLD.2
GSM2696499_KO_Cold_3.CEL	KO.COLD	KO	COLD	KO.COLD.3

4.3 Packages installation in R

Packages not available in the basic R installation need to be installed before the analysis can be done (see Note 3).

As commented in *Materials* section, packages needed to do the study, may be downloaded from distinct repositories. The most common ones will be CRAN for standard packages or Bioconductor for Bioconductor packages.

Standard R packages can be downloades and installed from default repositories with the `install.packages` function. Bioconductor packages can be downloaded and installed with the function `install()` from the `BiocManager` package.

The code below will download and install the packages needed for the analysis. Note that **this code must be executed only once**. Subsequent executions of the analysis do not need to re-install the packages:

The first chunk makes a fresh install of basic bioconductor packages.

```
> if (!requireNamespace("BiocManager", quietly = TRUE))
+     install.packages("BiocManager")
> BiocManager::install()
```

The second chunk installs packages specifically needed for this study.Some packages may require compilation, so a good idea if you are not working on a linux machine, is to have `Rtools` installed. This can be downladed from https://cran.r-project.org/bin/windows/Rtools/.

```
> install.packages("knitr")
> install.packages("colorspace")
> install.packages("gplots")
> install.packages("ggplot2")
> install.packages("ggrepel")
> install.packages("htmlTable")
> install.packages("prettydoc")
> install.packages("devtools")
> install.packages("BiocManager")
> BiocManager::install("oligo")
> BiocManager::install("pd.mogene.2.1.st")
> BiocManager::install("arrayQualityMetrics")
> BiocManager::install("pvca")
> # NOT NEEDED UNTIL ANALYSES ARE PERFORMED
> BiocManager::install("limma")
> BiocManager::install("genefilter")
> BiocManager::install("mogene21sttranscriptcluster.db")
> BiocManager::install("annotate")
> BiocManager::install("org.Mm.eg.db")
> BiocManager::install("ReactomePA")
> BiocManager::install("reactome.db")
```

4.4 Read the CEL files

Next step is to read the raw data (CEL files) and to store in a variable (in this case we have called it **rawData**). First we have to load the package *oligo* with the function *library*. In this package are coded the functions to read the CEL files. Take attention to put the correct folder where the CEL files are saved when executing *list.celfiles* function.

```
> library(oligo)
> celFiles <- list.celfiles("./data", full.names = TRUE)
> library(Biobase)
> my.targets <-read.AnnotatedDataFrame(file.path("./data","targets.csv"),
+                                      header = TRUE, row.names = 1,
+                                      sep=";")
> rawData <- read.celfiles(celFiles, phenoData = my.targets)
```

Note that we have read another time the targets file, but now using another specific function: *read.AnnotatedDataFrame*, and stored in a new variable called *my.targets*. We have done that to associate the information stored in the CEL files with the targets file in on single variable with the last code’s line. This object is called *ExpressionSet* and is designed to combine several different sources of information into a single convenient structure. We could store in this object all the information available about the experiment performed (protocol used, experiment data, microarray type,...). Moreover it allow us to change the long name of the samples, for the short and more comprehensive label previously coded in *ShortName* column of the *targets*.

```
> my.targets@data$ShortName->rownames(pData(rawData))
> colnames(rawData) <-rownames(pData(rawData))
>
> head(rawData)
```

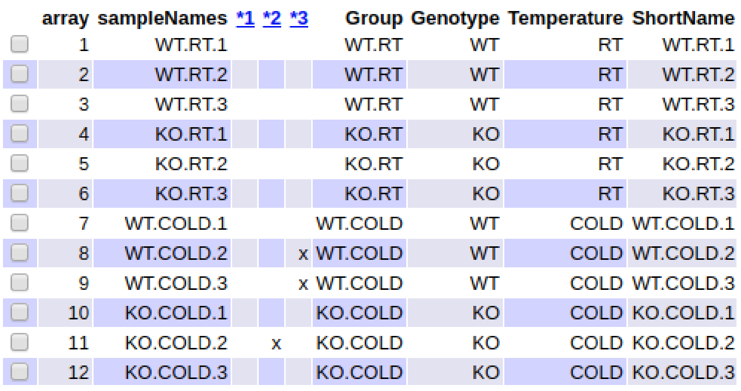
```
GeneFeatureSet (storageMode: lockedEnvironment)
assayData: 1 features, 12 samples
  element names: exprs
protocolData
  rowNames: WT.RT.1 WT.RT.2 ... KO.COLD.3 (12 total)
  varLabels: exprs dates
  varMetadata: labelDescription channel
phenoData
  rowNames: WT.RT.1 WT.RT.2 ... KO.COLD.3 (12 total)
  varLabels: Group Genotype Temperature ShortName
  varMetadata: labelDescription channel
featureData: none
experimentData: use 'experimentData(object)'
Annotation: pd.mogene.2.1.st
```

4.5 Quality control of raw data

Once the raw data is loaded it is the moment to check if the data have enough quality for normalization. This step is very important since bad quality data could introduce a lot of noise in the analysis, that normalization process could not solve. ArrayQualityMetrics package performs different quality approaches, like boxplot of the intensity of the data and Principal Component Analysis (PCA) among others. If one array is above a certain threshold defined in the function it is marked with an asterisk as an outlier. When a certain array is marked three times it should be revised carefully, perhaps this sample will have to be rejected to improve the overall quality of the experiment. First step is to load the library to gain access to the function. Be careful again to specify correctly the destination folder of the results:

```
> library(arrayQualityMetrics)
> arrayQualityMetrics(rawData)
```

We have to check the results of the quality analysis in a recently created QCDir.Raw folder inside the results folder previously created. Inside this folder we have to look for a file called *index.html*, which opens a web page from where we will be able to access a summary of the analysis performed. The image in figure @ref(fig:QCRawDataRes) shows the header of this file which contains a table with three columns indicating some quality criteria that should be verified by “good quality” arrays. In this example three samples have been marked once. Usually if there is only one mark it means that potential problems are small so we can decide to keep all the arrays in the analyis.



	array	sampleNames	*1	*2	*3	Group	Genotype	Temperature	ShortName
☐	1	WT.RT.1				WT.RT	WT	RT	WT.RT.1
☐	2	WT.RT.2				WT.RT	WT	RT	WT.RT.2
☐	3	WT.RT.3				WT.RT	WT	RT	WT.RT.3
☐	4	KO.RT.1				KO.RT	KO	RT	KO.RT.1
☐	5	KO.RT.2				KO.RT	KO	RT	KO.RT.2
☐	6	KO.RT.3				KO.RT	KO	RT	KO.RT.3
☐	7	WT.COLD.1				WT.COLD	WT	COLD	WT.COLD.1
☐	8	WT.COLD.2			x	WT.COLD	WT	COLD	WT.COLD.2
☐	9	WT.COLD.3			x	WT.COLD	WT	COLD	WT.COLD.3
☐	10	KO.COLD.1				KO.COLD	KO	COLD	KO.COLD.1
☐	11	KO.COLD.2		x		KO.COLD	KO	COLD	KO.COLD.2
☐	12	KO.COLD.3				KO.COLD	KO	COLD	KO.COLD.3

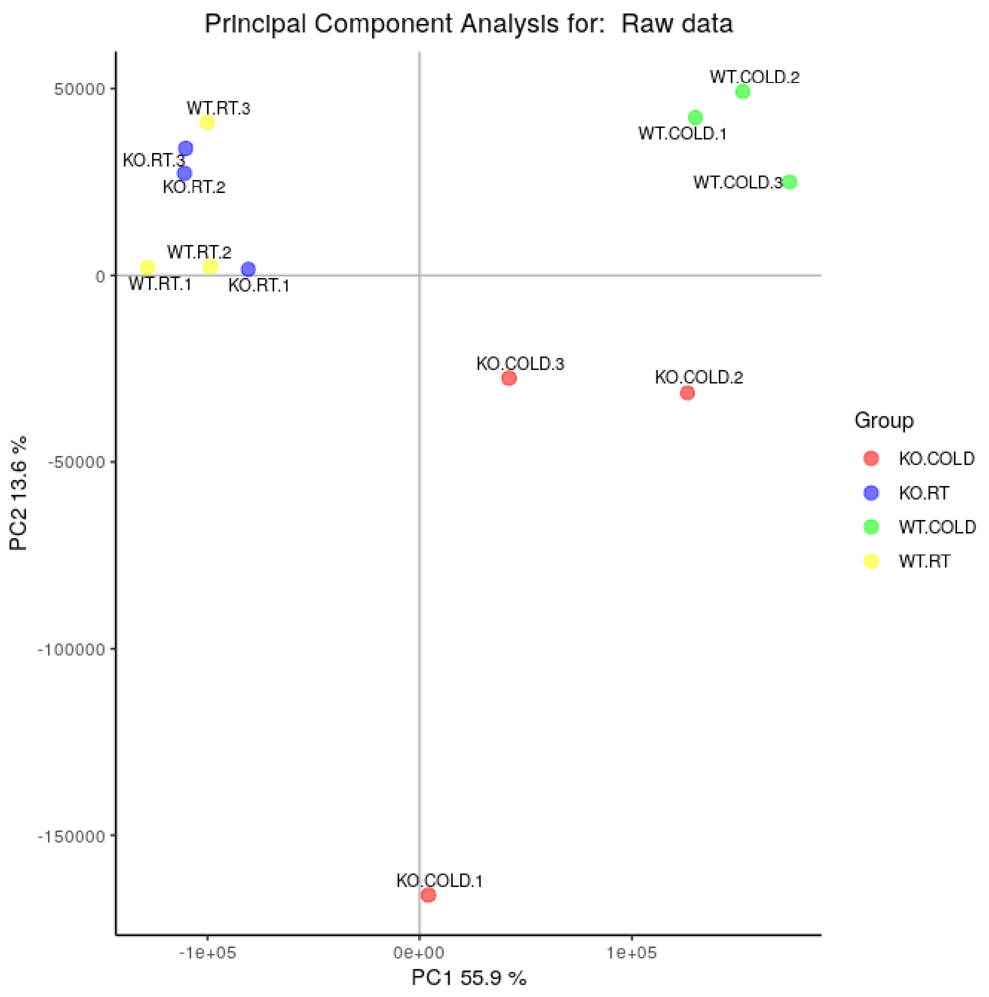
Aspect of the summary table, in the index.html file, produced by the arrayQualityMetrics package on the raw data

A more comprehensive principal component analysis can be obtained using a function specifically design for that. The code for this function is shown in the next code chunk.

```
> library(ggplot2)
> library(ggrepel)
> plotPCA3 <- function (datos, labels, factor, title, scale,colores, size = 1.5, glineas = 0.2
+   data <- prcomp(t(datos),scale=scale)
+   # plot adjustments
+   dataDf <- data.frame(data$x)
+   Group <- factor
+   loads <- round(data$sdev^2/sum(data$sdev^2)*100,1)
+   # main plot
+   p1 <- ggplot(dataDf,aes(x=PC1, y=PC2)) +
+     theme_classic() +
+     geom_hline(yintercept = 0, color = "gray70") +
+     geom_vline(xintercept = 0, color = "gray70") +
+     geom_point(aes(color = Group), alpha = 0.55, size = 3) +
+     coord_cartesian(xlim = c(min(data$x[,1])-5,max(data$x[,1])+5)) +
+     scale_fill_discrete(name = "Group")
+   # avoiding labels superposition
+   p1 + geom_text_repel(aes(y = PC2 + 0.25, label = labels),segment.size = 0.25, size = size)
+     labs(x = c(paste("PC1",loads[1],"%")),y=c(paste("PC2",loads[2],"%"))) +
+     ggtitle(paste("Principal Component Analysis for: ",title,sep=" "))+
+     theme(plot.title = element_text(hjust = 0.5)) +
+     scale_color_manual(values=colores)
+   }
```

Figure @ref(fig:PCARaw) shows the scatterplot of the first two principal components performed on the raw data.

```
> plotPCA3(exprs(rawData), labels = targets$ShortName, factor = targets$Group,
+          title="Raw data", scale = FALSE, size = 3,
+          colores = c("red", "blue", "green", "yellow"))
```



Visualization of the two first Principal Components for raw data

Note that we have defined in the function some parameters to facilitate the visualization.

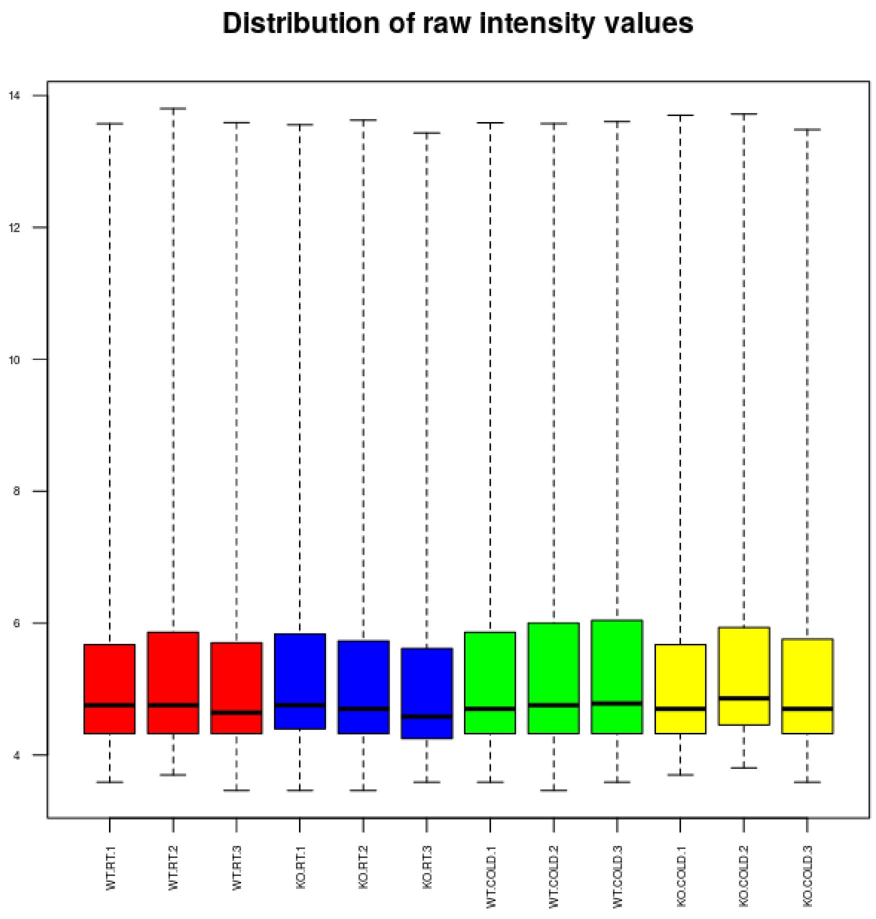
- the label of the samples, remember that it is coded in the *ShortName* column of the targets
- the characteristic to color the samples, coded in the *Group* column in targets

- the colors of each group

First component of the PCA accounts for 55.9% of the total variability of the samples, and as we can observe in the plot, this variability is mainly contributed by the *temperature* condition since samples incubated to 4 degrees are on the right and samples incubated at room temperature are on the left.

In the same way, we can easily visualize the intensity distribution of the arrays using boxplots. Figure @ref(fig:BoxplotRaw) shows a multiple boxplot depicting the distribution of the intensities along all samples.

```
> boxplot(rawData, cex.axis=0.5, las=2,  which="all",
+          col = c(rep("red", 3), rep("blue", 3), rep("green", 3), rep("yellow", 3)),
+          main="Distribution of raw intensity values")
```



Boxplot for arrays intensities (Raw Data)

A light variation of intensity among arrays is observed, but this is the expected for raw data.

4.6 Data normalization

Before beginning with differential expression analysis, it is necessary to make the arrays comparable among them and try to reduce, and if it is possible to eliminate, all the variability in the samples not owing to biological reasons.Normalization process tries to assure that intensity differences present in the array, reflects the differential expression of the genes, rather than artificial biases due to technical issues. Normalization process consists of three discrete steps: background correction, normalization, and summarization. Most commonly used method for array normalization is Robust Multichip Analysis Irizarry et al. (2003):

```
> eset_rma <- rma(rawData)
```

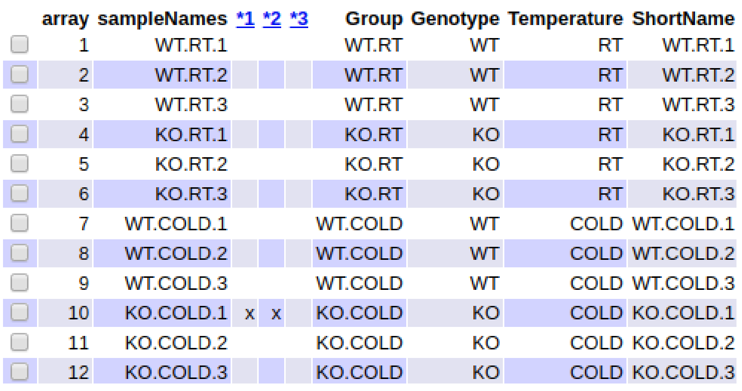
```
Background correcting
Normalizing
Calculating Expression
```

4.7 Quality control of normalized data

After performing normalization it is interesting to perform again a quality control to check how data looks. In the same way than before (look we have changed *rawData* object to *eset_rma*).

```
> arrayQualityMetrics(eset_rma, outdir = file.path("./results", "QCDir.Norm"), force=TRUE)
```

Figure @ref(fig:QCNormDataRes) shows the same summary as before, but performed on normalized data.

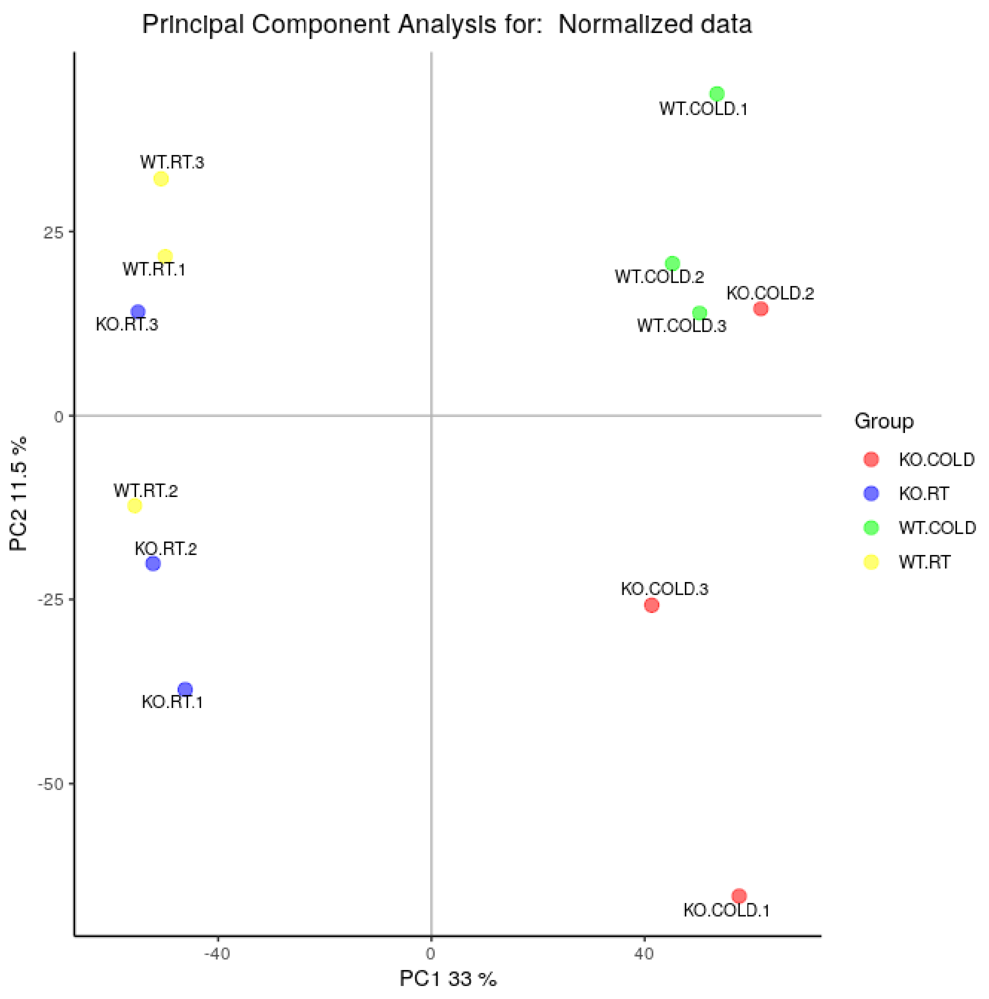


	array	sampleNames	*1	*2	*3	Group	Genotype	Temperature	ShortName
☐	1	WT.RT.1				WT.RT	WT	RT	WT.RT.1
☐	2	WT.RT.2				WT.RT	WT	RT	WT.RT.2
☐	3	WT.RT.3				WT.RT	WT	RT	WT.RT.3
☐	4	KO.RT.1				KO.RT	KO	RT	KO.RT.1
☐	5	KO.RT.2				KO.RT	KO	RT	KO.RT.2
☐	6	KO.RT.3				KO.RT	KO	RT	KO.RT.3
☐	7	WT.COLD.1				WT.COLD	WT	COLD	WT.COLD.1
☐	8	WT.COLD.2				WT.COLD	WT	COLD	WT.COLD.2
☐	9	WT.COLD.3				WT.COLD	WT	COLD	WT.COLD.3
☐	10	KO.COLD.1	x	x		KO.COLD	KO	COLD	KO.COLD.1
☐	11	KO.COLD.2				KO.COLD	KO	COLD	KO.COLD.2
☐	12	KO.COLD.3				KO.COLD	KO	COLD	KO.COLD.3

Aspect of the summary table, in the index.html file, produced by the arrayQualityMetrics package on normalized data

Figure @ref(fig:PCANorm) shows the scatterplot of the first two principal components performed on normalized data.

```
> plotPCA3(exprs(eset_rma), labels = targets$ShortName, factor = targets$Group,
+          title="Normalized data", scale = FALSE, size = 3,
+          colores = c("red", "blue", "green", "yellow"))
```

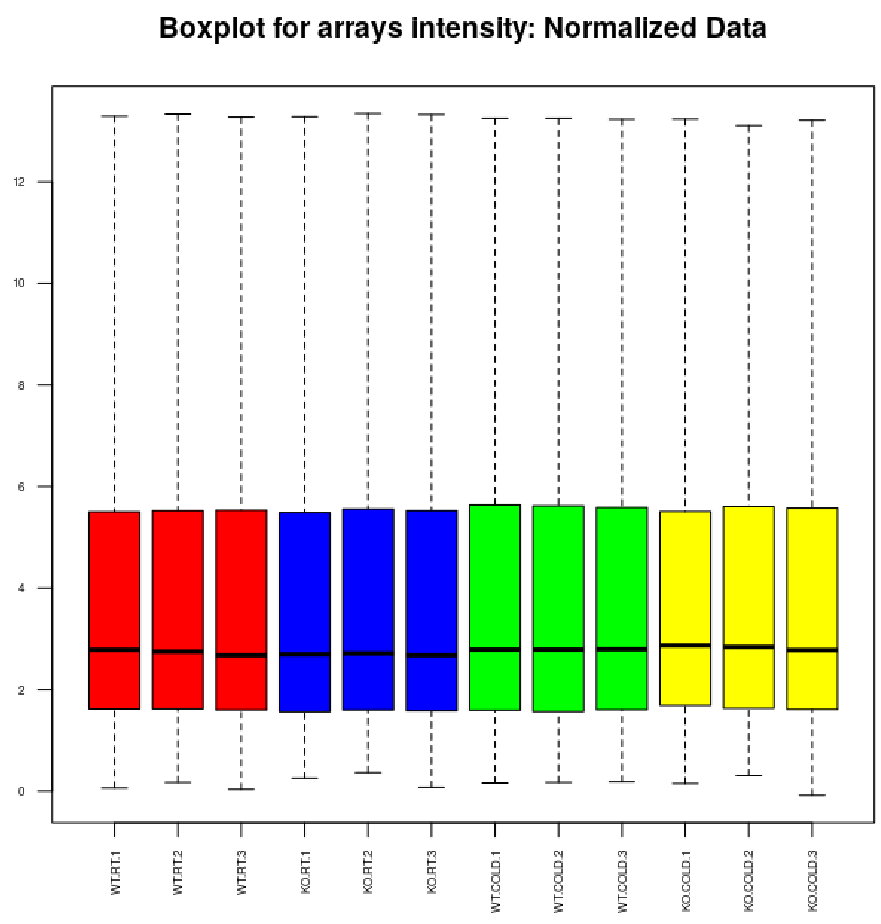


Visualization of first two principal components for normalized data

Now first component accounts for 33% of the total variability. Notice that the percentage of explained variability has decreased with respect to PCA performed on raw data. Similarly as with the PCA with raw data, it separates samples from *COLD* level of *temperature* condition on the right, and samples from *RT* level on the left. It is important to note that there are one sample from group *KO.RT* that groups near **WT.RT** and viceversa. It could be an issue of mislabeling of samples that should be checked with the laboratory that has processed the samples.

Figure @ref(fig:BoxplotNorm) shows a multiple boxplot depicting the distribution of the normalized intensities along all samples. Notice that all boxplots have the same aspect. This suggests that the normalization has worked fine. However it is important to be aware that RMA includes a step ("quantile normalization") where the empirical distribution of all the samples is set to the same values. As a consequence **it is expected that the boxplots are identical or at least very similar**.

```
> boxplot(eset_rma, cex.axis=0.5, las=2,  which="all",
+          col = c(rep("red", 3), rep("blue", 3), rep("green", 3), rep("yellow", 3)),
+          main="Boxplot for arrays intensity: Normalized Data")
```



Distribution of intensities for normalized data

4.8 Batch detection

Gene expression microarray results can be affected by minuscule differences in any number of non-biological variables like reagents from different lots, different technicians and the more usual issue the different processing date of samples from the same experiment. The cumulative error introduced by these time and place-dependent experimental variations is referred to as “batch effects”. Different approaches have been developed for identifying and removing batch effects from microarray data like surrogate variable analysis, Combat and Principal variation component analysis (PVCA).

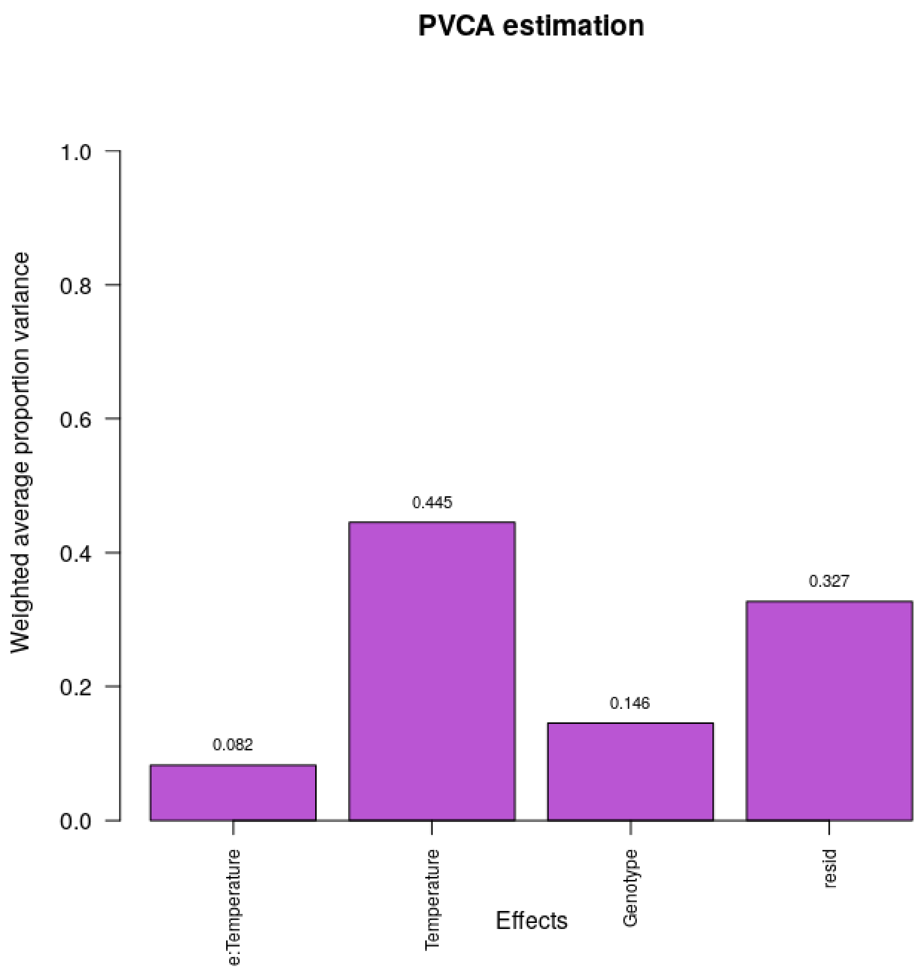
Here we will use the last one, Principal Variation Component Analysis, which estimates source and proportion of variation in two steps, principal component analysis, and variance component analysis. All samples were processed the same day (this can be seen using function `get.celfile.dates()` from package `affyio`) so a typical batch factor such as “Processing date” **needs not** be considered.

```
> #load the library
> library(pvca)
> pData(eset_rma) <- targets
> #select the threshold
> pct_threshold <- 0.6
> #select the factors to analyze
> batch.factors <- c("Genotype", "Temperature")
> #run the analysis
> pvcaObj <- pvcaBatchAssess (eset_rma, batch.factors, pct_threshold)
```

Figure @ref(fig:plotPVCA) shows a bar diagram with one bar per each source of variation included in the analysis. Their relative size indicates the percentage of variability attributable to each source. The plot shows that the main source of variation in the samples is the *Temperature* condition. This was also observed on the PCA plots on raw and normalized data in figures @ref(fig:PCARaw) and @ref(fig:PCANorm).

Notice that temperature **is not** a batch factor. It is an experimental factor that was included in the experimental desiogn.

```
> #plot the results
> bp <- barplot(pvcaObj$dat, xlab = "Effects",
+   ylab = "Weighted average proportion variance",
+   ylim= c(0,1.1),col = c("mediumorchid"), las=2,
+   main="PVCA estimation")
> axis(1, at = bp, labels = pvcaObj$label, cex.axis = 0.75, las=2)
> values = pvcaObj$dat
> new_values = round(values , 3)
> text(bp,pvcaObj$dat,labels = new_values, pos=3, cex = 0.7)
```



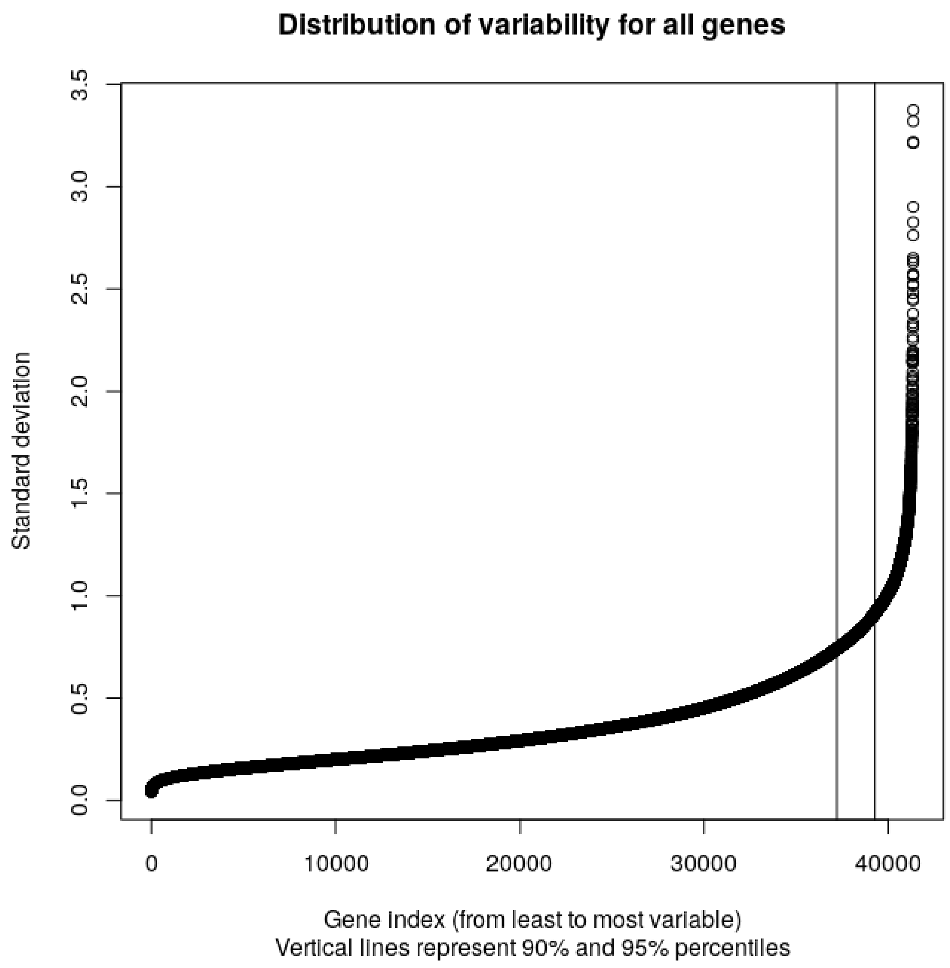
Relative importance of the different factors -genotype, temperature and interaction- affecting gene expression

4.9 Detecting most variable genes

Selection of differentially expressed genes is affected by the number of genes on which we make it. The higher the number, the greater the necessary adjustment of p-values (as will be seen below), which will lead us to end up miscarrying more genes.

If a gene is differentially expressed, it is expected that there is a certain difference between the groups, and therefore the overall variance of the gene will be greater than that of those that do not have differential expression. Plotting the overall variability of all genes is useful to decide which percentage of genes shows a variability that can be attributed to other causes than random variation. Figure @ref(fig:SDplot) depicts the standard deviations of all genes sorted from smallest to biggest values. The plot shows that the most variable genes are those with a standard deviation above 90-95% of all standard deviations.

```
> sds <- apply (exprs(eset_rma), 1, sd)
> sdsO<- sort(sds)
> plot(1:length(sdsO), sdsO, main="Distribution of variability for all genes",
+      sub="Vertical lines represent 90% and 95% percentiles",
+      xlab="Gene index (from least to most variable)", ylab="Standard deviation")
> abline(v=length(sds)*c(0.9,0.95))
```



Values of standard deviations allong all samples for all genes ordered from smallest to biggest

4.10 Filtering least variable genes

Filtering out those genes whose variability can be attributed to random variation, that is the genes that are, reasonably, not expected to be differential expressed, has proven to be useful to reduce the number of tests to be performed with the corresponding increase in power Hackstadt and Hess (2009).

Function `nsFilter` from the bioconductor package `genefilter` can be used to remove genes based on a variability threshold. If an annotation package -associating probesets identifiers and gene identifiers from different databases- is available it can also be used to remove probesets which do not have a gene identifier associated.

```
> library(genefilter)
> library(mogene21sttranscriptcluster.db)
> annotation(eset_rma) <- "mogene21sttranscriptcluster.db"
> filtered <- nsFilter(eset_rma,
+                      require.entrez = TRUE, remove.dupEntrez = TRUE,
+                      var.filter=TRUE, var.func=IQR, var.cutoff=0.75,
+                      filterByQuantile=TRUE, feature.exclude = "^AFFX")
```

Function `nsFilter` returns the filtered values and a report of the filtering results.

```
> print(filtered$filter.log)
```

```
$numDupsRemoved
[1] 671

$numLowVar
[1] 17981

$numRemoved.ENTREZID
[1] 16699
```

```
> eset_filtered <-filtered$eset
```

After filtering there are 5994 genes left. Note that we have stored the genes left in the variable *eset_filtered*

4.11 Saving normalized and filtered data

Normalized filtered data are the starting point for further analyses but we may want to go back to them, for example to review specific gene expression values. It is usual to save the binary objects but also to write expression values into text or excel files. Writing to Excel from R is not a trivial task -for strange it may seem- because different packages work differently depending of the operating system, so it is omitted from the code.

```
> write.csv(exprs(eset_rma), file="./results/normalized.Data.csv")
> write.csv(exprs(eset_filtered), file="./results/normalized.Filtered.Data.csv")
> save(eset_rma, eset_filtered, file="./results/normalized.Data.Rda")
```

4.12 Defining the experimental setup: The design matrix

Selection of differential expressed genes basically consists of doing some type of test, usually on a gene-wise basis, to compare gene expression between groups. This can be done using many different approaches (see Chrominski and Tkacz (2015)). There is a general agreement that using standard statistical tests such as t-tests is not appropriate Jeanmougin et al. (2010) and that better options are methods that perform some type of variance shrinking Allison et al. (2006). Techniques specifically developed for microarrays such as SAM Tusher, Tibshirani, and Chu (2001) or Linear Models for Microarrays Smyth (2004) have proved to produce much better results Chrominski and Tkacz (2015).

In this protocol the Linear Models for Microarrays method, implemented in the limma package Smyth (2005) is used to select differential expressed genes.

The first step for the analysis based on linear models is to create the **design matrix**. Basically it is a table that describes the allocation of each sample to a group or experimental condition. It has as many rows as samples and as many columns as groups (if only one factor is considered). Each row contains a one in the column of the group to which the sample belongs and a zero in the others.

The design matrix can be defined manually or from a factor variable that may have been introduced in the "targets" file with this aim created specifically for it. In this study that "Group" variable is a combination of the two experimental conditions, "KO/Wild" and "RT/COLD" which are jointly represented as one factor with 4 levels.

```
> if (!exists("eset_filtered")) load (file="./results/normalized.Data.Rda")
```

```
> library(limma)
> designMat<- model.matrix(~0+Group, pData(eset_filtered))
> colnames(designMat) <- c("KO.COLD", "KO.RT", "WT.COLD", "WT.RT")
> print(designMat)
```

```
   KO.COLD KO.RT WT.COLD WT.RT
1        0     0       0     1
2        0     0       0     1
3        0     0       0     1
4        0     1       0     0
5        0     1       0     0
6        0     1       0     0
7        0     0       1     0
8        0     0       1     0
9        0     0       1     0
10       1     0       0     0
11       1     0       0     0
12       1     0       0     0
attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$Group
[1] "contr.treatment"
```

4.13 Defining comparisons with the Contrasts Matrix

The contrasts matrix is used to describe the comparisons between groups It consists of as many columns as comparisons and as many rows as groups (that is, as columns of the design matrix). A comparison between groups - called "contrast" - is represented by a "1" and a "-1" in the rows of groups to compare and zeros in the rest. If several groups intervened in the comparison would have as many coefficients as groups with the only restriction that its sum would be zero.

In this example we want to check the effect of knocking out a gene ("KO vs WT") separately for cold and RT temperature. Also we want to test if there is interaction between knocking out the gene and temperature. This can be done by doing three comparisons described below:

```
> cont.matrix <- makeContrasts (KOvsWT.COLD = KO.COLD-WT.COLD,
+                               KOvsWT.RT = KO.RT-WT.RT,
+                               INT = (KO.COLD-WT.COLD) - (KO.RT-WT.RT),
+                                levels=designMat)
> print(cont.matrix)
```

```
         Contrasts
Levels     KOvsWT.COLD KOvsWT.RT INT
  KO.COLD            1         0   1
  KO.RT              0         1  -1
  WT.COLD           -1         0  -1
  WT.RT              0        -1   1
```

The contrast matrix is defined to perform three comparisons: Effect of KO in Cold temperature, Effect of KO in RT temperature and interaction between KO and temperature.

4.14 Model estimation and gene selection

Once the design matrix and the contrasts have been defined, we can proceed to estimate the model, estimate the contrasts and perform the significance tests that will lead to the decision, for each gene and each comparison, if they can be considered differential expressed.

The method implemented in the `limma` package extends the traditional analysis using Empirical Bayes models to combine an estimate of variability based on the entire matrix with individual estimates based on each individual values providing improved error estimates Smyth (2004).

The analysis provides the usual test statistics such as Fold-change t-moderated or adjusted p-values that are used to order the genes from more unless differential expressed.

In order to control the percentage of false positives that may result from high number of contrasts made simultaneously the p-values are adjusted so that we have control over the false positive rate using the Benjamini and Hochberg method Benjamini and Hochberg (1995).

All relevant information for further exploration of the results is stored in an R object of class `MArrayLM` defined in the `limma` package. Here it is named as `fit.main`.

```
> library(limma)
> fit<-lmFit(eset_filtered, designMat)
> fit.main<-contrasts.fit(fit, cont.matrix)
> fit.main<-eBayes(fit.main)
> class(fit.main)
```

```
[1] "MArrayLM"
attr(,"package")
[1] "limma"
```

4.15 Obtaining lists of differentially expressed genes

The `limma` package implements function `topTable` which contains, for a given contrast a list of genes ordered from smallest to biggest p–value which can be considered to be most to least differential expressed. For each gene the following statistics are provided:

- `logFC`: Mean difference between groups.
- `AveExpr`: Average expression of all genes in the comparison.
- `t` : Moderated t-statistic (t-test-like statistic for the comparison).
- `P.Value`: Test p–value.
- `adj.P.Val`: Adjusted p–value following Benjamini and Hochberg (1995)

- `B` : B-statistic: Posterior log odds of the gene of being vs non being differential expressed.

We can have a look at the first lines of each `topTable`.

For comparison 1 (KOvsWT.COLD): Genes that change their expression between KO and WT in cold temperature:

```
> topTab_KOvsWT.COLD <- topTable (fit.main, number=nrow(fit.main), coef="KOvsWT.COLD", adjust=
> head(topTab_KOvsWT.COLD)
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
17497829	2.474954	6.706683	15.73570	0e+00	0.0000301	10.278310
17407049	3.676524	4.697071	14.66069	0e+00	0.0000323	9.724106
17529307	-3.628616	3.493648	-10.96870	2e-07	0.0003523	7.268699
17373930	2.071352	5.412865	10.79258	3e-07	0.0003523	7.125248
17251565	1.991109	2.955199	10.72249	3e-07	0.0003523	7.067379
17291821	-2.434786	5.348719	-10.24570	5e-07	0.0004035	6.661420

For comparison 2 (KOvsWT.RT): Genes that change their expression between KO and WT in room temperature:

```
> topTab_KOvsWT.RT <- topTable (fit.main, number=nrow(fit.main), coef="KOvsWT.RT", adjust="fdr
> head(topTab_KOvsWT.RT)
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
17407049	3.675590	4.697071	14.65697	0e+00	0.0000242	9.814462
17282970	-3.565316	9.093793	-14.60544	0e+00	0.0000242	9.785700
17497829	2.280728	6.706683	14.50082	0e+00	0.0000242	9.726841
17425609	2.207391	2.464984	11.92132	1e-07	0.0001202	8.056504
17472497	-3.366076	7.132534	-11.88752	1e-07	0.0001202	8.031485
17399411	-1.816331	4.457054	-11.15264	2e-07	0.0001886	7.464482

For comparison 3 (INT): Genes that behave differently between comparison 1 and 2:

```
> topTab_INT  <- topTable (fit.main, number=nrow(fit.main), coef="INT", adjust="fdr")
> head(topTab_INT)
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
17282970	3.369838	9.093793	9.761369	8.00e-07	0.0046174	5.718379
17494820	2.023908	1.704030	8.449033	3.30e-06	0.0098079	4.583532
17251565	2.127194	2.955199	8.100148	4.90e-06	0.0098846	4.246054

	logFC	AveExpr	t	P.Value	adj.P.Val	B
17274184	-1.800778	8.001305	-7.331232	1.29e-05	0.0180141	3.443058
17543045	2.012296	9.748289	7.213657	1.50e-05	0.0180141	3.312709
17432247	2.798819	4.218885	7.055025	1.85e-05	0.0184941	3.133530

First column of each topTable contains the manufacturer's (Affymetrix) ID for each probeset. Next step is to guess which gene correspond to each Affymetrix ID. This process is called **annotation**.

4.16 Gene Annotation

Once we have the top table it is useful to provide additional information on the features that have been selected. This process is called "annotation" and essentially what it does is to look for information to associate identifiers that appear in the top table, usually corresponding to probesets or transcripts depending of the array type, with more familiar names such as the Gene Symbol, the Entrez Gene identifier or the Gene description.

For simplicity, because there are three toptables, a function annotating one topTable with a given package is prepared and used.

```
> annotatedTopTable <- function(topTab, anotPackage)
+ {
+   topTab <- cbind(PROBEID=rownames(topTab), topTab)
+   myProbes <- rownames(topTab)
+   thePackage <- eval(parse(text = anotPackage))
+   geneAnots <- select(thePackage, myProbes, c("SYMBOL", "ENTREZID", "GENENAME"))
+   annotatedTopTab<- merge(x=geneAnots, y=topTab, by.x="PROBEID", by.y="PROBEID")
+ return(annotatedTopTab)
+ }
```

```
> topAnnotated_KOvsWT.COLD <- annotatedTopTable(topTab_KOvsWT.COLD,
+ anotPackage="mogene21sttranscriptcluster.db")
> topAnnotated_KOvsWT.RT <- annotatedTopTable(topTab_KOvsWT.RT,
+ anotPackage="mogene21sttranscriptcluster.db")
> topAnnotated_INT <- annotatedTopTable(topTab_INT,
+ anotPackage="mogene21sttranscriptcluster.db")
> write.csv(topAnnotated_KOvsWT.COLD, file="./results/topAnnotated_KOvsWT_COLD.csv")
> write.csv(topAnnotated_KOvsWT.RT, file="./results/topAnnotated_KOvsWT_RT.csv")
> write.csv(topAnnotated_INT, file="./results/topAnnotated_INT.csv")
```

Annotation makes the tables more comprehensible. Table @ref(tab:annotatedTop) shows the annotations added to results "topTable" for the comparison "KOvsWT.COLD" (only the first four columns are shown).

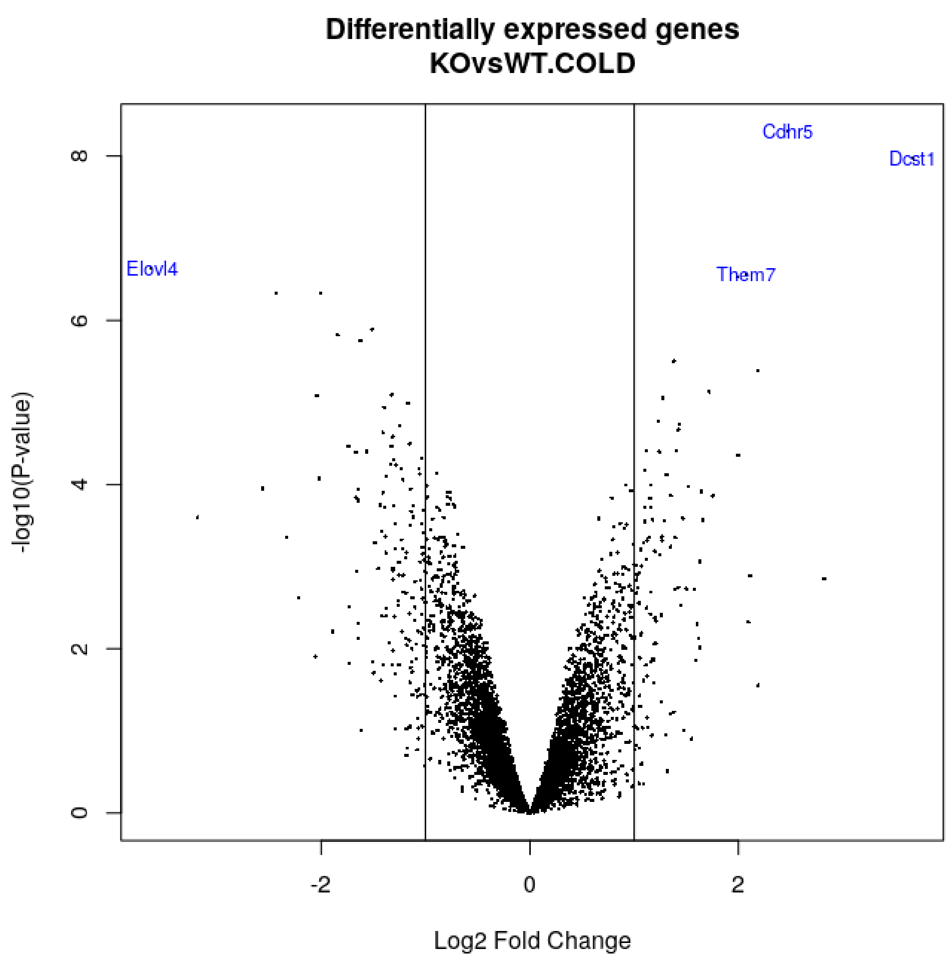
```
   PROBEID        SYMBOL ENTREZID
1 17210887       Atp6v1h   108664
2 17210984        Pcmtd1   319263
3 17211000          Rrs1    59014
4 17211131         Prex2   109294
5 17211174 A830018L16Rik   320492
                                                              GENENAME
1                              ATPase, H+ transporting, lysosomal V1 subunit H
2 protein-L-isoaspartate (D-aspartate) O-methyltransferase domain containing 1
3                                             ribosome biogenesis regulator 1
4    phosphatidylinositol-3,4,5-trisphosphate-dependent Rac exchange factor 2
```

```
5                                                    RIKEN cDNA A830018L16 gene
```

4.17 Visualizing differential expression

A visualization of the overall differential expression can be obtained using volcano-plots. These plots show if there are many or few genes with a large fold-change and significantly expressed or if this number is low. These graphs represent in the X-axis the changes of expression in logarithmic scale ("biological effect") and in the Y-axis the "minus logarithm" of the p-value or alternatively the `B` statistic ("Statistical effect"). Figure @ref(fig:volcanoPlot) shows a volcano plot for the comparison between KO and WT in COLD temperature. The names of the top 4 genes (i.e. the first four genes in the topTable) are shown in the plot.

```
> library(mogene21sttranscriptcluster.db)
> geneSymbols <- select(mogene21sttranscriptcluster.db, rownames(fit.main), c("SYMBOL"))
> SYMBOLS<- geneSymbols$SYMBOL
> volcanoplot(fit.main, coef=1, highlight=4, names=SYMBOLS,
+             main=paste("Differentially expressed genes", colnames(cont.matrix)[1], sep="\n")
>   abline(v=c(-1,1))
```



Volcano plot for the comparison between KO and WT in COLD temperature. The names of the top 4 genes (i.e. the first four genes in the topTable) are shown in the plot

4.18 Multiple Comparisons

When one selects genes in several comparisons it is usually interesting to know which genes have been selected in each comparison. Sometimes biologically relevant genes will be those that are selected in one of them but not in others. In other occasions he interest will lie in genes that are selected in all comparisons.

Functions `decideTests` and `VennDiagram` from package limma can be used to annotate and count the genes selected in every comparison.

```
> library(limma)
> res<-decideTests(fit.main, method="separate", adjust.method="fdr", p.value=0.1, lfc=1)
```

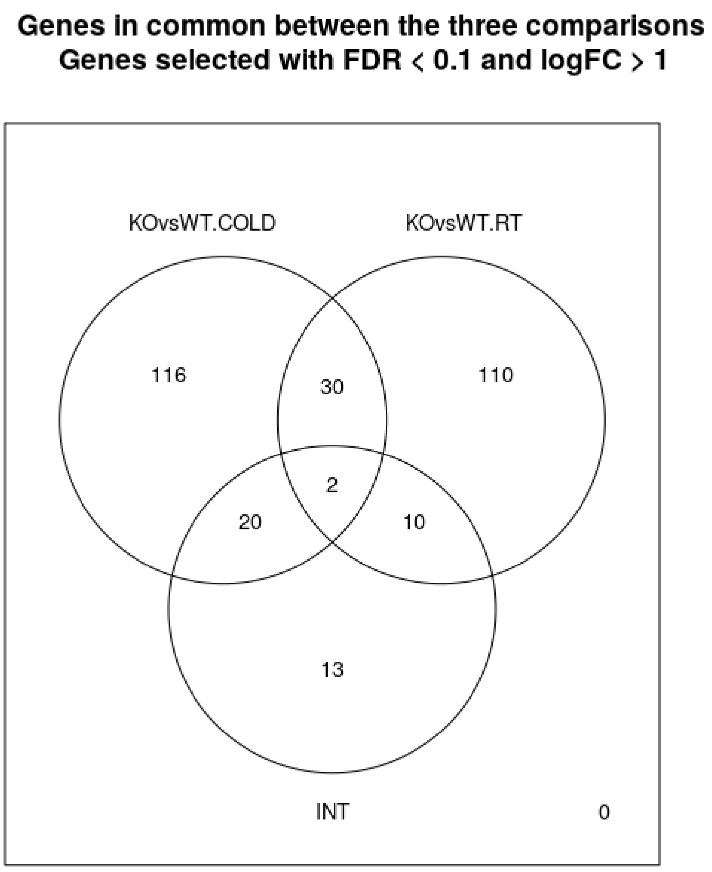
This object has as many columns as comparisons and as many rows as genes. Per each gene and comparison a "+1" denotes significantly up-regulated (t-test values > 0, FDR < selected cutoff), a "-1" significantly down-regulated (t-test values < 0, FDR < selected cutoff) and a "0" non significant difference (FDR > selected cutoff).

```
> sum.res.rows<-apply(abs(res),1,sum)
> res.selected<-res[sum.res.rows!=0,]
> print(summary(res))
```

```
       KOvsWT.COLD KOvsWT.RT  INT
Down            99        72   24
NotSig        5826      5842 5949
Up              69        80   21
```

This can be visualized in a Venn Diagram. Figure @ref(fig:vennDiagram) shows a Venn diagram depicting the number of genes that have been called differentially expressed in each comparison with a given cutoff (here the cutoff is defined by "FDR < 0.1" and "logFC > 1" The figure shows how many of these genes are shared by one or more selections.

```
> vennDiagram (res.selected[,1:3], cex=0.9)
> title("Genes in common between the three comparisons\n Genes selected with FDR < 0.1 and log
```



Venn diagram showing the genes in common between the three comparisons performed

4.19 Heatmaps

Genes that have been selected as differential expressed may be visualized using a heatmap. These plots use color palettes to highlight distinct values –here positive (up-regulation) or negative (down-regulation) significantly differential expressions.

Heatmaps can be used to visualize the expression values of differential expressed genes with no specific order, but it is usually preferred to plot them doing a hierarchical clustering on genes (rows) or columns(samples) in order to find groups of genes with common patterns of variation which can eventually be associated to the different groups being compared.

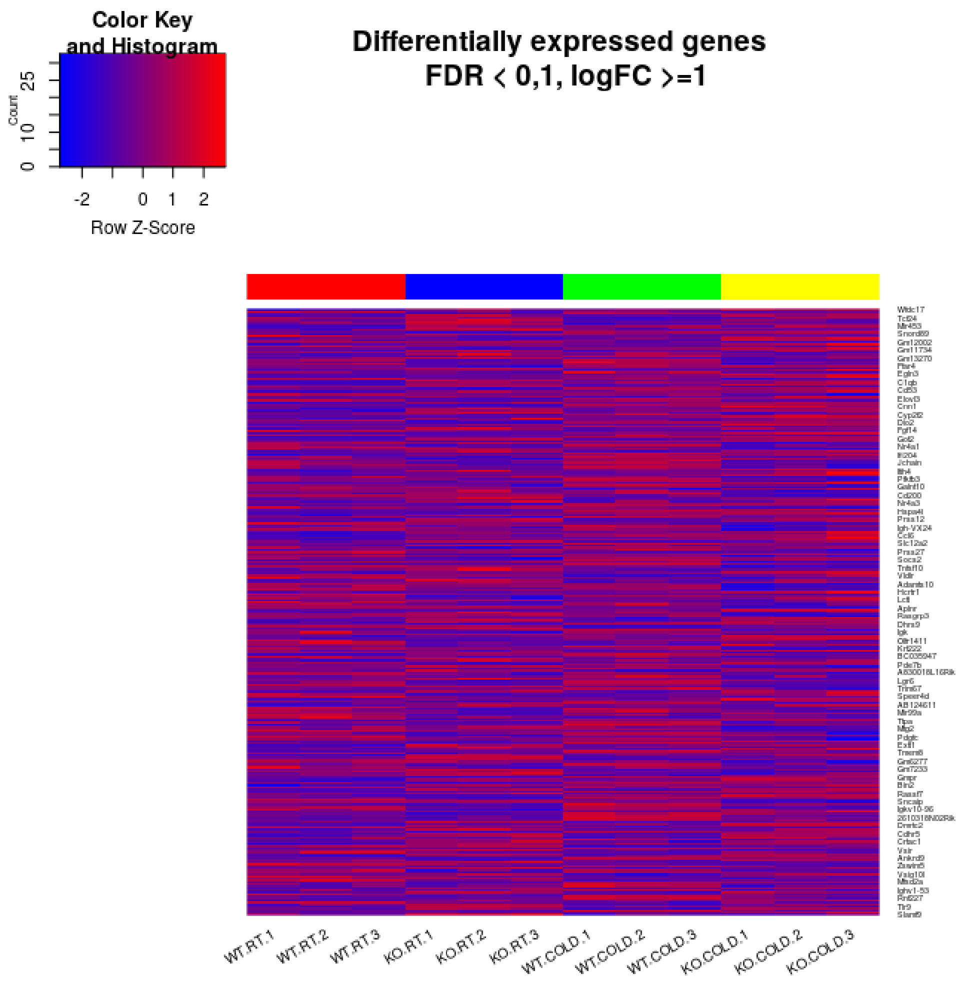
There may be discussion on which genes to select for doing a heatmap. A common option is to select the gens that have been selected in the previous steps, that is the genes that have been called differential expressed in at least one comparison.

```
> probesInHeatmap <- rownames(res.selected)
> HMdata <- exprs(eset_filtered)[rownames(exprs(eset_filtered)) %in% probesInHeatmap,]
>
> geneSymbols <- select(mogene21sttranscriptcluster.db, rownames(HMdata), c("SYMBOL"))
> SYMBOLS<- geneSymbols$SYMBOL
> rownames(HMdata) <- SYMBOLS
> write.csv(HMdata, file = file.path("./results/data4Heatmap.csv"))
```

With the selected data a heatmap can be generated with or without clustering genes and/or samples.

Figure @ref(fig:heatmapNoclustering) shows a heatmap produced for all the genes selected with the same criteria described above (FDR < 0.1 and logFC > 1) where no clustering of genes and samples is performed.

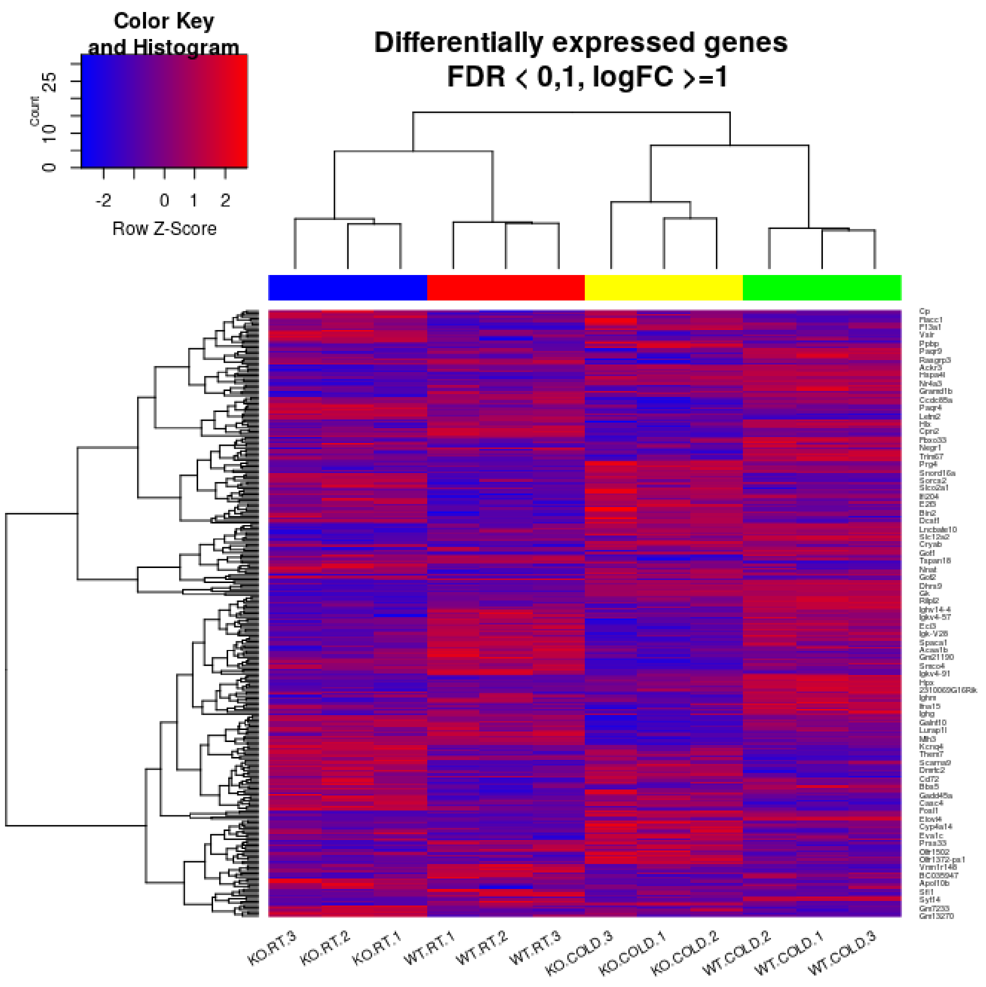
```
> my_palette <- colorRampPalette(c("blue", "red"))(n = 299)
> library(gplots)
>
> heatmap.2(HMdata,
+           Rowv = FALSE,
+           Colv = FALSE,
+           main = "Differentially expressed genes \n FDR < 0,1, logFC >=1",
+           scale = "row",
+           col = my_palette,
+           sepcolor = "white",
+           sepwidth = c(0.05,0.05),
+           cexRow = 0.5,
+           cexCol = 0.9,
+           key = TRUE,
+           keysize = 1.5,
+           density.info = "histogram",
+           ColSideColors = c(rep("red",3),rep("blue",3), rep("green",3), rep("yellow",3)),
+           tracecol = NULL,
+           dendrogram = "none",
+           srtCol = 30)
```



Heatmap for expression data without any grouping

Figure @ref(fig:heatmapClustering) shows a heatmap produced for all the genes selected with the same criteria described above (FDR < 0.1 and logFC > 1) where genes and samples are forced to group by row and column similarity respectivelty.

```
> heatmap.2(HMdata,
+           Rowv = TRUE,
+           Colv = TRUE,
+           dendrogram = "both",
+           main = "Differentially expressed genes \n FDR < 0,1, logFC >=1",
+           scale = "row",
+           col = my_palette,
+           sepcolor = "white",
+           sepwidth = c(0.05,0.05),
+           cexRow = 0.5,
+           cexCol = 0.9,
+           key = TRUE,
+           keysize = 1.5,
+           density.info = "histogram",
+           ColSideColors = c(rep("red",3),rep("blue",3), rep("green",3), rep("yellow",3)),
+           tracecol = NULL,
+           srtCol = 30)
```



Heatmap for expression data grouping genes (rows) and samples (columns) by their similarity

4.20 Biological Significance of results

Once a list of gene has being obtained that characterizes the difference between two conditions it has to be interpreted. Although this requires, of course, a good understanding of the underlying biological problem, a statistical approach known as "Gene Set Analysis" can be useful for suggesting ideas for the interpretation.

With this aim these types of analyses seek to establish whether, given a list of genes selected for being differential expressed between two conditions, the functions, biological processes or molecular pathways that characterize them appear on this list more frequently than among the rest of the genes analyzed.

There are many variants of these types of analysis, see Khatri, Sirota, and Butte (2012), but here we will use the basic enrichment analysis as described in implemented in the `ReactomePA` Bioconductor package. The analysis is done on the ReactomePA annotation database https://reactome.org/.

Analyses of this type need a minimum number of genes to be reliable, preferably a few hundreds than a few dozens, so it is common to perform a selection less restrictive than with the previous steps. For instance an option is to include all genes with a non-stringent FDR cutoff, such as FDR < 0.15 without filtering by minimum "fold-change").

As a first step we prepare the list of gene lists that will be analyzed:

```
> listOfTables <- list(KOvsWT.COLD = topTab_KOvsWT.COLD,
+                      KOvsWT.RT  = topTab_KOvsWT.RT,
+                      INT = topTab_INT)
> listOfSelected <- list()
> for (i in 1:length(listOfTables)){
+   # select the toptable
+   topTab <- listOfTables[[i]]
+   # select the genes to be included in the analysis
+   whichGenes<-topTab["adj.P.Val"]<0.15
+   selectedIDs <- rownames(topTab)[whichGenes]
+   # convert the ID to Entrez
+   EntrezIDs<- select(mogene21sttranscriptcluster.db, selectedIDs, c("ENTREZID"))
+   EntrezIDs <- EntrezIDs$ENTREZID
+   listOfSelected[[i]] <- EntrezIDs
+   names(listOfSelected)[i] <- names(listOfTables)[i]
+ }
> sapply(listOfSelected, length)
```

```
KOvsWT.COLD   KOvsWT.RT         INT
        769         453          85
```

The analysis also requires to have the Entrez Identifiers for all genes analyzed. It is an open discussion if what one should use is "all genes analyzed" -that is genes that have been retained in the analysis and are part of the "topTable"- or all genes available. In this case we use the second option and define our universe to be all genes that have at least one annotation in the Gene Ontology

```
> mapped_genes2GO <- mappedkeys(org.Mm.egGO)
> mapped_genes2KEGG <- mappedkeys(org.Mm.egPATH)
> mapped_genes <- union(mapped_genes2GO , mapped_genes2KEGG)
```

The Biological significance analysis will be applied only to the first two lists. Sometimes yet another decomposition is applied so that up and downregulated genes are separately analyzed. This will not be done here because there is no clear biological argument to proceed so in all cases.

```
> library(ReactomePA)
>
> listOfData <- listOfSelected[1:2]
> comparisonsNames <- names(listOfData)
> universe <- mapped_genes
>
```

```
> for (i in 1:length(listOfData)){
+   genesIn <- listOfData[[i]]
+   comparison <- comparisonsNames[i]
+   enrich.result <- enrichPathway(gene = genesIn,
+                                  pvalueCutoff = 0.05,
+                                  readable = T,
+                                  pAdjustMethod = "BH",
+                                  organism = "mouse",
+                                  universe = universe)
+ 
+   cat("##################################")
+   cat("\nComparison: ", comparison,"\n")
+   print(head(enrich.result))
+ 
+   if (length(rownames(enrich.result@result)) != 0) {
+   write.csv(as.data.frame(enrich.result),
+              file =paste0("./results/","ReactomePA.Results.",comparison,".csv"),
+              row.names = FALSE)
+ 
+   pdf(file=paste0("./results/","ReactomePABarplot.",comparison,".pdf"))
+     print(barplot(enrich.result, showCategory = 15, font.size = 4,
+             title = paste0("Reactome Pathway Analysis for ", comparison,". Barplot")))
+   dev.off()
+ 
+   pdf(file = paste0("./results/","ReactomePAcnetplot.",comparison,".pdf"))
+     print(cnetplot(enrich.result, categorySize = "geneNum", schowCategory = 15,
+          vertex.label.cex = 0.75))
+   dev.off()
+   }
+ }
```

```
##################################
Comparison:  KOvsWT.COLD
                         ID                                  Description
R-MMU-75105     R-MMU-75105                  Fatty acyl-CoA biosynthesis
R-MMU-8978868 R-MMU-8978868                        Fatty acid metabolism
R-MMU-75876     R-MMU-75876 Synthesis of very long-chain fatty acyl-CoAs
              GeneRatio  BgRatio       pvalue   p.adjust     qvalue
R-MMU-75105       8/329  33/8698 2.332981e-05 0.01558431 0.01558431
R-MMU-8978868    18/329 168/8698 6.210131e-05 0.02074184 0.02074184
R-MMU-75876       6/329  22/8698 1.248588e-04 0.02780190 0.02780190

R-MMU-75105                                                                 Elovl2/Acsl5/Elovl
R-MMU-8978868 Aloxe3/Cyp4a14/Ptgds/Abcd1/Acot1/Elovl2/Acsl5/Elovl7/Crot/Elovl3/Acads/Cpt1a/Acs
R-MMU-75876                                                                           Elovl2/A
              Count
R-MMU-75105       8
R-MMU-8978868    18
R-MMU-75876       6
```

```
##################################
Comparison:  KOvsWT.RT
                         ID                                 Description
R-MMU-1483249 R-MMU-1483249              Inositol phosphate metabolism
R-MMU-400451   R-MMU-400451 Free fatty acids regulate insulin secretion
R-MMU-977606   R-MMU-977606            Regulation of Complement cascade
R-MMU-112043   R-MMU-112043                    PLC beta mediated events
R-MMU-112040   R-MMU-112040                   G-protein mediated events
R-MMU-1855204 R-MMU-1855204     Synthesis of IP3 and IP4 in the cytosol
```

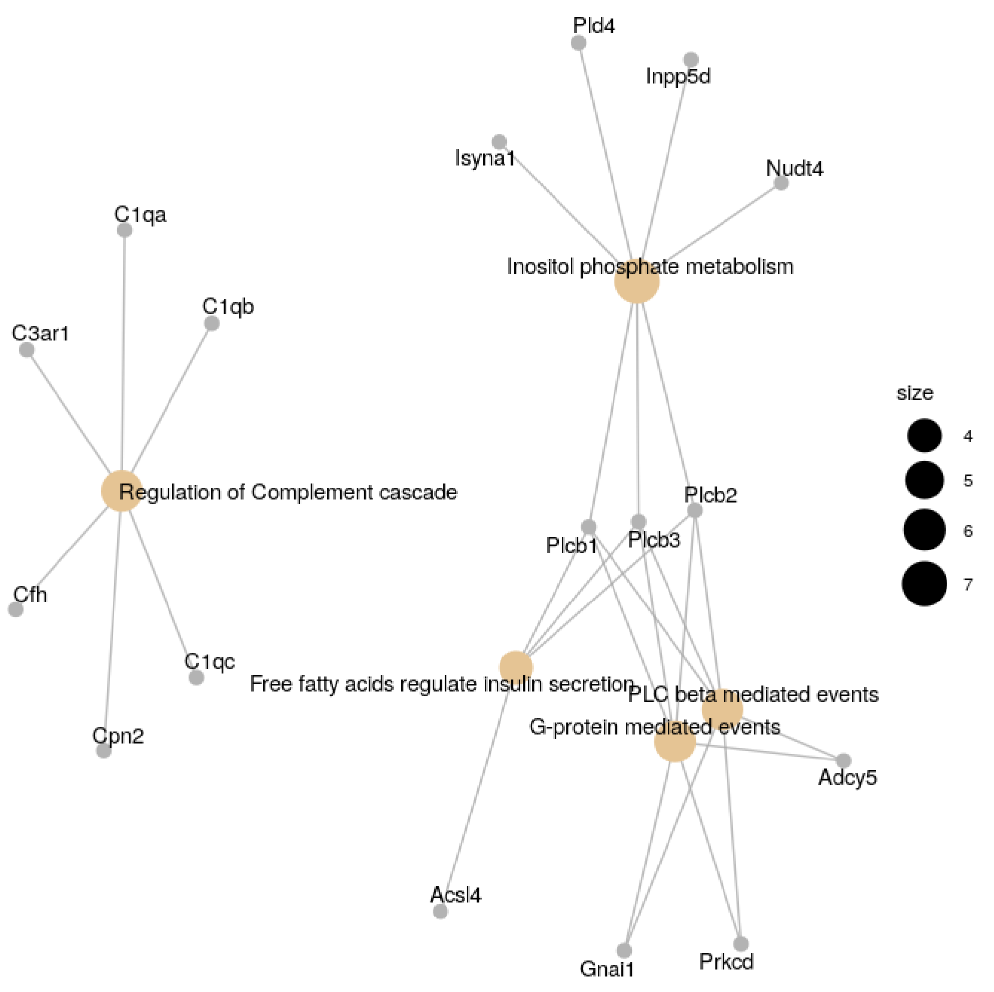
```
R-MMU-1855204 R-MMU-1855204     Synthesis of IP3 and IP4 in the cytosol
              GeneRatio BgRatio       pvalue   p.adjust     qvalue
R-MMU-1483249     7/201 46/8698 7.898619e-05 0.01721053 0.01583064
R-MMU-400451      4/201 11/8698 8.042305e-05 0.01721053 0.01583064
R-MMU-977606      6/201 37/8698 1.806995e-04 0.02088957 0.01921470
R-MMU-112043      6/201 38/8698 2.104972e-04 0.02088957 0.01921470
R-MMU-112040      6/201 39/8698 2.440370e-04 0.02088957 0.01921470
R-MMU-1855204     5/201 27/8698 3.344361e-04 0.02385644 0.02194370
                                                  geneID Count
R-MMU-1483249 Isyna1/Nudt4/Inpp5d/Pld4/Plcb3/Plcb2/Plcb1     7
R-MMU-400451                     Plcb3/Plcb2/Acsl4/Plcb1     4
R-MMU-977606                Cpn2/C1qb/C3ar1/Cfh/C1qc/C1qa     6
R-MMU-112043          Prkcd/Adcy5/Plcb3/Gnai1/Plcb2/Plcb1     6
R-MMU-112040          Prkcd/Adcy5/Plcb3/Gnai1/Plcb2/Plcb1     6
R-MMU-1855204               Inpp5d/Pld4/Plcb3/Plcb2/Plcb1     5
```

The results obtained in the analysis of biological significance are:

- a *.csv* file with a summary of all the enriched pathways and the associated statistics.
- a bar plot with the best enriched pathways. Height of the bar plot is the number of genes of our analysis related with that pathway. Moreover, pathways are ordered by statistical significance.
- a plot with a network of the enriched pathways and the relation among the genes included.

Figure @ref(fig:network) shows the network produced from the genes selected in the comparison "KO and WT in RT"

```
> cnetplot(enrich.result, categorySize = "geneNum", schowCategory = 15,
+         vertex.label.cex = 0.75)
```


Pld4
Inpp5d
Isyna1
Nudt4
Inositol phosphate metabolism
C1qa
C1qb
C3ar1
Regulation of Complement cascade
Plcb2
Plcb1
Plcb3
Cfh
C1qc
Free fatty acids regulate insulin secretion
PLC beta mediated events
G-protein mediated events
Adcy5
Cpn2
Acsl4
Gnai1
Prkcd
size
4
5
6
7

Network obtained from the Reactome enrichment analysis on the list obtained from the comparison between KO and WT in RT

In our study for comparison *KOvsWT.COLD* three enriched pathway have been found, for example *Synthesis of very long-chain fatty acyl-CoAs*, and in comparison *KOvsCTL.RT*, five enriched pathways have been found (Table @ref(tab:tableReacto)). An interesting one is *Inositol phosphate metabolism*.

First rows and columns for Reactome results on KOvsWT.RT.csv comparison

	Description	GeneRatio	BgRatio	pvalue	p.adjust
R-MMU-1483249	Inositol phosphate metabolism	7/201	46/8698	7.89861917854937e-05	0.017210532182374
R-MMU-400451	Free fatty acids regulate insulin secretion	4/201	11/8698	8.0423047581187e-05	0.017210532182374
R-MMU-977606	Regulation of Complement cascade	6/201	37/8698	0.000180699532427207	0.0208895657554835
R-MMU-112043	PLC beta mediated events	6/201	38/8698	0.000210497172826187	0.0208895657554835

4.21 Summary of results

Once the process has been completed one has obtained a, sometimes long, list of files with the data and the analysis results. These files are of the basis for discussing the results and looking for a biological interpretation. Both aspects exceed the goals of this chapter so they are ommited here.

It is useful to create a file with the type, name and description of all the files generated along the analysis. Table @ref(tab:listOfFiles) shows the list of files generated in the current case study.

List of files generated in the analysis

List_of_Files
data4Heatmap.csv
normalized.Data.csv
normalized.Data.Rda
normalized.Filtered.Data.csv
QCDir.Norm
ReactomePA.Results.KOvsWT.COLD.csv
ReactomePA.Results.KOvsWT.RT.csv
ReactomePABarplot.KOvsWT.COLD.pdf
ReactomePABarplot.KOvsWT.RT.pdf

List_of_Files
ReactomePAcnetplot.KOvsWT.COLD.pdf
ReactomePAcnetplot.KOvsWT.RT.pdf
topAnnotated_INT.csv
topAnnotated_KOvsWT_COLD.csv
topAnnotated_KOvsWT_RT.csv

5 Notes

1 R and Bioconductor are open source software. This mean they are free (as in "free bier") but it also means that compatibility between versions is not always 100% granted. It is important to know which version of R and Bioconductor one is using for the analysis.

2 Although it may seem irrelevant it is important to be aware of regional settings before reading or writing text files. For instance in some European countries the decimal point is the "comma" while in anglosaxon ones it is the "dot". R can read any of these data formats but, of course it need to be informed of which format is used in any situation.]

3 Sometimes package installation may present some difficulties, often related to the operating system being used or even the workplace (for example if the user is behindd a proxy). In these cases the recommendation is to try to install packages one by one and, if needed contact the institution IT team.

References

Allison, David B., Xiangqin Cui, Grier P. Page, and Mahyar Sabripour. 2006. "Microarray data analysis: From disarray to consolidation and consensus." *Nature Reviews Genetics* 7 (1): 55–65. https://doi.org/10.1038/nrg1749.

Benjamini, Yoav, and Yosef Hochberg. 1995. "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing." *Source Journal of the Royal Statistical Society. Series B (Methodological)* 57 (1): 289–300. http://www.jstor.org/stable/2346101 http://www.jstor.org/ http://www.jstor.org/action/showPublisher?publisherCode=black.

Chrominski, Kornel, and Magdalena Tkacz. 2015. "Comparison of High-Level Microarray Analysis Methods in the Context of Result Consistency." *PLOS ONE* 10 (6). Public Library of Science: e0128845. https://doi.org/10.1371/JOURNAL.PONE.0128845.

Clough, Emily, and Tanya Barrett. 2016. "The Gene Expression Omnibus Database." In *Methods in Molecular Biology (Clifton, N.j.)*, 1418:93–110. https://doi.org/10.1007/978-1-4939-3578-9_5.

Draghici, Sorin. 2012. *Statistics and data analysis for microarrays using R and Bioconductor*. CRC Press. https://www.crcpress.com/Statistics-and-Data-Analysis-for-Microarrays-Using-R-and-Bioconductor/Draghici/p/book/9781439809754.

Efron, Bradley. 2013. *Large-scale inference : empirical Bayes methods for estimation, testing, and prediction*. Cambridge University Press. http://admin.cambridge.org/academic/subjects/statistics-probability/statistical-theory-and-methods/large-scale-inference-empirical-bayes-methods-estimation-testing-and-prediction.

Gonzalo Sanz, Ricardo, and Alex Sánchez-Pla. 2019. "Statistical Analysis of Microarray Data." In *Microarray Bioinformatics*, edited by Verónica Bolón-Canedo and Amparo Alonso-Betanzos, 87–121. New York, NY: Springer New York. https://doi.org/10.1007/978-1-4939-9442-7_5.

Hackstadt, Amber J, and Ann M Hess. 2009. "Filtering for increased power for microarray data analysis." *BMC Bioinformatics* 10 (January). BioMed Central: 11. https://doi.org/10.1186/1471-2105-10-11.

Irizarry, Rafael A., Bridget Hobbs, Francois Collin, Yasmin D. Beazer-Barclay, Kristen J. Antonellis, Uwe Scherf, and Terence P. Speed. 2003. "Exploration, Normalization, and Summaries of High Density Oligonucleotide Array Probe Level Data." *Biostatistics* 4 (2): 249–64. https://doi.org/10.1093/biostatistics/4.2.249.

Jeanmougin, Marine, Aurelien de Reynies, Laetitia Marisa, Caroline Paccard, Gregory Nuel, and Mickael Guedj. 2010. "Should We Abandon the t-Test in the Analysis of Gene Expression Microarray Data: A Comparison of Variance Modeling Strategies." Edited by Kerby Shedden. *PLoS ONE* 5 (9). Public Library of Science: e12336. https://doi.org/10.1371/journal.pone.0012336.

Khatri, Purvesh, Marina Sirota, and Atul J. Butte. 2012. "Ten Years of Pathway Analysis: Current Approaches and Outstanding Challenges." *PLOS Computational Biology* 8 (2): e1002375. https://doi.org/10.1371/journal.pcbi.1002375.

Li, Siming, Lin Mi, Lei Yu, Qi Yu, Tongyu Liu, Guo-Xiao Wang, Xu-Yun Zhao, Jun Wu, and Jiandie D. Lin. 2017. "Zbtb7b Engages the Long Noncoding Rna Blnc1 to Drive Brown and Beige Fat Development and Thermogenesis." *Proceedings of the National Academy of Sciences* 114 (34). National Academy of Sciences: E7111–E7120. https://doi.org/10.1073/pnas.1703494114.

Mehta, Jai Prakash, and Sweta Rani. 2011. "Software and Tools for Microarray Data Analysis." In *Methods in Molecular Biology (Clifton, N.j.)*, 784:41–53. https://doi.org/10.1007/978-1-61779-289-2_4.

Sánchez-Pla, Alex. 2014. "DNA Microarrays Technology: Overview and Current Status." In, edited by Alejandro Cifuentes Carolina Simó and Virginia García-Cañas, Volume 63:1–23. Fundamentals of Advanced Omics Technologies: From Genes to Metabolites. Elsevier. http://www.sciencedirect.com/science/article/pii/B9780444626516000015.

Sánchez-Pla, Alex, Ferran Reverter, M. Carme Ruíz de Villa, and Manuel Comabella. 2012. "Transcriptomics: mRNA and alternative splicing." *Journal of Neuroimmunology* 248 (1-2): 23–31. https://doi.org/10.1016/j.jneuroim.2012.04.008.

Smyth, G. K. 2005. "limma: Linear Models for Microarray Data." In *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, 397–420. New York: Springer-Verlag. https://doi.org/10.1007/0-387-29362-0_23.

Smyth, Gordon K. 2004. "Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments." *Statistical Applications in Genetics and Molecular Biology* 3 (1): 1–25. https://doi.org/10.2202/1544-6115.1027.

Tusher, V G, R Tibshirani, and G Chu. 2001. "Significance analysis of microarrays applied to the ionizing radiation response." *Proceedings of the National Academy of Sciences of the United States of America* 98 (9): 5116–21. https://doi.org/10.1073/pnas.091062498.

1. https://github.com/ASPteaching/Omics_Data_Analysis-Case_Study_1-Microarrays↩