From RAW to Knowledge: The KnowSeq user guide

KnowSeq: R/Bioconductor package

Daniel Castillo, Juan Manuel Galvez, Francisco Manuel Ortuno, Luis Javier Herrera and Ignacio Rojas

august, 15 of 2019

Contents

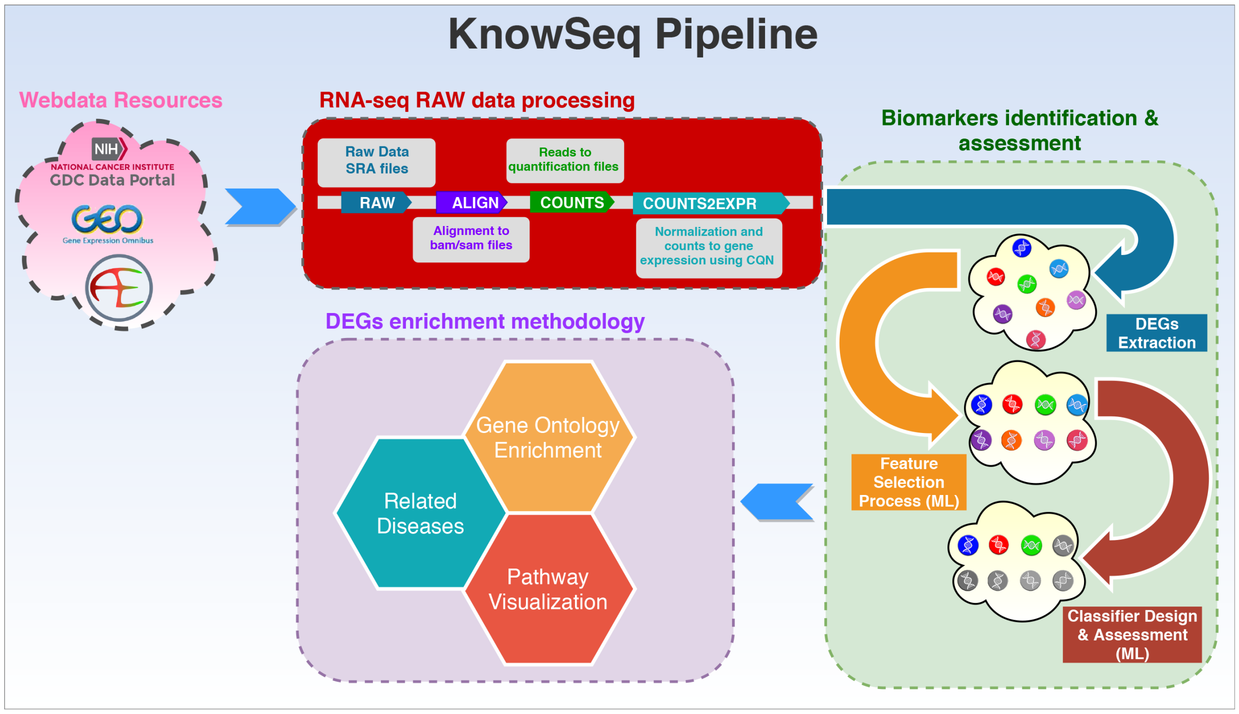
1 Installation

To install and load KnowSeq package in R, it is necessary the previous installation of BiocManager from Bioconductor. The next code shows how this install can be performed:

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

BiocManager::install("KnowSeq")

library(KnowSeq)
```

2 Introduction

KnowSeq proposes a whole pipeline that comprises the most relevant steps in the RNA-seq gene expression analysis, with the main goal of extracting biological knowledge from raw data (Differential Expressed Genes, Gene Ontology enrichment, pathway visualization and related diseases). In this sense, KnowSeq allows aligning raw data from the original fastq or sra files, by using the most renowned aligners such as tophat2, hisat2, salmon and kallisto. Nowadays, there is no package that only from the information of the samples to align -included in a text file-, automatically performs the download and alignment of all of the samples. Furthermore, the package includes functions to: calculate the gene expression values; remove batch effect; calculate the Differentially Expressed Genes (DEGs); plot different graphs; and perform the DEGs enrichment with the GO information, pathways visualization and related diseases information retrieval. Moreover, KnowSeq is the only package that allows applying both a machine learning and DEGs enrichment processes just after the DEGs extraction. To achieve these objectives, there are functions that allows performing a feature selection process as well as a machine learning process using k-NN, RF or SVM algorithms. Similarly, there are functions allowing the retrieval of biological knowledge of the DEGs candidates. This idea emerged with the aim of proposing a complete tool to the research community containing all the necessary steps to carry out complete studies in a simple and fast way. To achieve this goal, the package uses the most relevant and widespread tools in the scientific community for the aforementioned tasks. The current version of the aligner functions works under Unix, but further version will be extended to MAC_OS and to Windows (if the tools were available). This pipeline has been used in our previous publications for processing raw RNA-seq data and to perform the DEGs extraction and the machine learning classifier design steps, also for their integration with microarray data [CGH$^+$17, GCH$^+$18, CGH$^+$19].


KnowSeq Pipeline
Webdata Resources
NIH
NATIONAL CANCER INSTITUTE
GDC Data Portal
GEO
Gene Expression Omnibus
RNA-seq RAW data processing
Raw Data SRA files
Reads to quantification files
RAW
ALIGN
COUNTS
COUNTS2EXPR
Alignment to bam/sam files
Normalization and counts to gene expression using CQN
Biomarkers identification & assessment
DEGs Extraction
Feature Selection Process (ML)
Classifier Design & Assessment (ML)
DEGs enrichment methodology
Gene Ontology Enrichment
Related Diseases
Pathway Visualization

The whole pipeline included in KnowSeq has been designed carefully with the purpose of achieving a great quality and robustness in each of the steps that conform the pipeline. For that, the pipeline has three fundamental processes:

- RNA-seq RAW data processing.
- Biomarkers identification & assessment.
- DEGs enrichment methodology.

The first process is focused on the RNA-seq RAW data treatment. This step has the purpose of extracting a set of count files from raw files stored in the repositories supported by our package (NCBI/GEO [BWL+12], ArrayExpress [KHK+14] and GDC-Portal). The second one comprises the Differential Expressed Genes (DEGs) identification and extraction, and the assessment of those DEGs by applying advanced machine learning techniques (feature selection process and supervised classification). The last process, once the DEGs were assessed, is the DEGs enrichment methodology which allows retrieving biological information from the DEGs. In this process, relevant information (such as related diseases, biological processes associated and pathways) about the DEGs is retrieved by using very well-known tools and databases. The three types of enrichment are the Gene Ontology (GO) study, the pathways visualization taking into account the gene expression, and the related diseases to the DEGs. With the pipeline designed and addressed by KnowSeq, researchers can convert the RAW data of RNA-seq into real knowledge on the identification of possible gene signatures about the studied diseases.

3 RNA-seq RAW data processing

3.1 Aligners preparation

In order to avoid version incompatibilities with the aligners and the installation of the required tools, pre-compiled versions will be used to run the R functions. Consequently, all the tools were compressed and stored in an external server to be downloaded whenever it is required (http://iwbbio.ugr.es/utils/unixUtils.tar.gz). If the tools are directly downloaded from the link, the compressed files must be decompressed in the current project folder in R or RStudio. The name of the resultant folder must be *"utils"*. Nevertheless, this file can be downloaded automatically by just calling the function *rawAlignment*, in case the folder utils is not detected in the project folder. This is all needed to run the different aligners through the function *rawAlignment.* It is not possible to run the alignment without the utils folder. It must be mentioned too that the different files included in the compressed *.tar.gz* are not only the aligners but also functions needed in the raw alignment process. The tools included are the following:

- Bowtie2 [LS12].
- Hisat2 [KLS17].
- Htseq-count [APH15].
- Kallisto [BPMP16].
- Salmon [PDL+17].
- Samtools [Li11].
- Sratoolkit [SX12].
- Tophat2 [KPT+13].
- GDC-client.

3.2 Launching Raw Alignment step

The *rawAlignment* function allows running different aligners, chosen by the user. The function takes as single input a CSV from GEO or ArrayExpress loaded in R. There is the possibility to process data from GDC-portal, but a previous authorization (token file) from this platform is required. Then, the user has to select with the parameter seq which aligner he wants to use, by default this parameter runs tophat2. Furthermore, there is a set of logical parameters to edit the default pipeline followed for the function. With the parameters the user can select if the BAM/SAM/Count files are created. The user can choose if wants to download the reference genome, the GTF, and which version. Even if the user has custom FASTA and GTF files, this can be specified by setting the parameter *referenceGenome* to *"custom"* and using the parameters *customFA* and *customGTF* to indicates the paths to the custom files. Other functionality is the possibility to process BAM files from the GDC Portal database by setting to *TRUE* the parameter *fromGDC.* Then the function will download the specific genome reference of GDC and process the BAM files to Count files. Furthermore, if the user has access to the controlled data, with the token and the manifest acquired from

GDC Portal web platform, the samples can be downloaded automatically. An example to run the function with hisat2 aligner is showed below:

```r
# Downloading one series from NCBI/GEO and one series from ArrayExpress
downloadPublicSeries(c("GSE74251"))

# Using read.csv for NCBI/GEO files (read.csv2 for ArrayExpress files)
GSE74251csv <- read.csv("ReferenceFiles/GSE74251.csv")

# Performing the alignment of the samples by using hisat2 aligner
rawAlignment(GSE74251csv,seq="hisat2",downloadRef=TRUE,downloadSamples=TRUE,BAMfiles = TRUE,
SAMfiles = TRUE,countFiles = TRUE,referenceGenome = 38, fromGDC = FALSE, customFA = "",
customGTF = "", tokenPath = "", manifest = "",tx2Counts = "")
```

To run the function with salmon or kallisto, it is necessary to use the parameter *tx2Counts*. The quantification files of these aligners contain the identification of the transcriptions, but for the count files it is necessary to convert these transcriptions IDs to gene IDs. To perform that, the *tx2Counts* parameter needs a matrix with two columns. One column with the transcription IDs and a second column with the correspondent gene IDs for each transcription. The package *tximportData* [Lov18] has a set of files that contain different transcript conversion that can be used to achieve the tx2Counts matrix. An example to run the function with kallisto aligner is showed below:

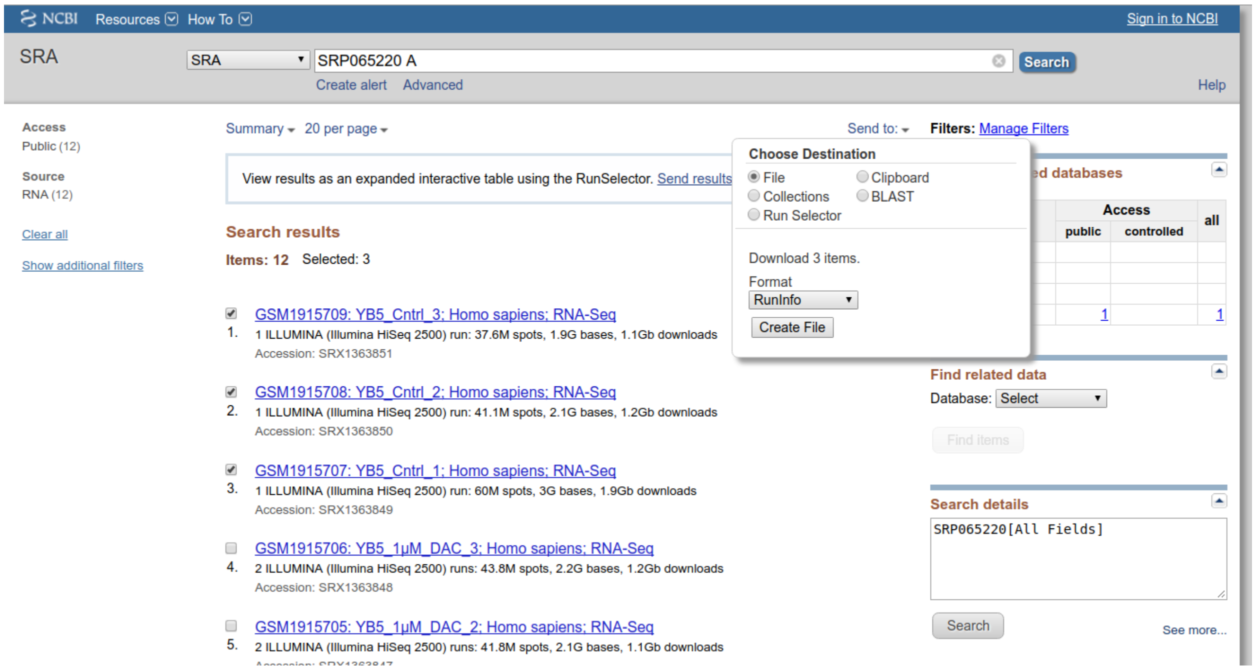
```r
# Downloading one series from NCBI/GEO and one series from ArrayExpress
downloadPublicSeries(c("GSE74251"))

# Using read.csv for NCBI/GEO files (read.csv2 for ArrayExpress files)
GSE74251csv <- read.csv("ReferenceFiles/GSE74251.csv")

# Loading the transcripts to genes converter variable

dir <- system.file("extdata", package="tximportData")

tx2gene <- read.csv(file.path(dir, "tx2gene.ensembl.v87.csv"))

# Performing the alignment of the samples by using kallisto aligner

rawAlignment(GSE74251csv,seq="kallisto",downloadRef=TRUE,downloadSamples=TRUE,BAMfiles = TRUE,
SAMfiles = TRUE,countFiles = TRUE,referenceGenome = 38, fromGDC = FALSE, customFA = "",
customGTF = "", tokenPath = "", manifest = "",tx2Counts = tx2gene)
```

RawAlignment function creates a folder structure in the current project folder which will store all the downloaded and created files. The main folder of this structure is the folder ReferenceFiles but inside of it there are more folders that allows storing the different files used by the process in an organized way.

Another important requirement to take into account is the format of the csv file used to launch the function. It could be from three repositories, two publics (NCBI/GEO and ArrayExpress) and one controlled (GDC Portal). Each of these repositories has its own format in the csv file that contains the information to download and process the desired samples. The necessary format for each repository is explained below.

3.2.1 NCBI/GEO CSV format

Series belonging to RNA-seq have a SRA identifier. If this identifier is clicked, a list with the samples that conform this series is showed. Then, the desired samples of the series can be checked and the CSV is automatically generated by clicking the button shown in the image below:



SRA: SRP065220 A

View results as an expanded interactive table using the RunSelector. Send results

Search results

Items: 12 Selected: 3

Download 3 items.

Format: RunInfo

1. GSM1915709: YB5_Cntrl_3; Homo sapiens; RNA-Seq
1 ILLUMINA (Illumina HiSeq 2500) run: 37.6M spots, 1.9G bases, 1.1Gb downloads
Accession: SRX1363851

2. GSM1915708: YB5_Cntrl_2; Homo sapiens; RNA-Seq
1 ILLUMINA (Illumina HiSeq 2500) run: 41.1M spots, 2.1G bases, 1.2Gb downloads
Accession: SRX1363850

3. GSM1915707: YB5_Cntrl_1; Homo sapiens; RNA-Seq
1 ILLUMINA (Illumina HiSeq 2500) run: 60M spots, 3G bases, 1.9Gb downloads
Accession: SRX1363849

4. GSM1915706: YB5_1μM_DAC_3; Homo sapiens; RNA-Seq
2 ILLUMINA (Illumina HiSeq 2500) runs: 43.8M spots, 2.2G bases, 1.2Gb downloads
Accession: SRX1363848

5. GSM1915705: YB5_1μM_DAC_2; Homo sapiens; RNA-Seq
2 ILLUMINA (Illumina HiSeq 2500) runs: 41.8M spots, 2.1G bases, 1.1Gb downloads

Search details: SRP065220[All Fields]

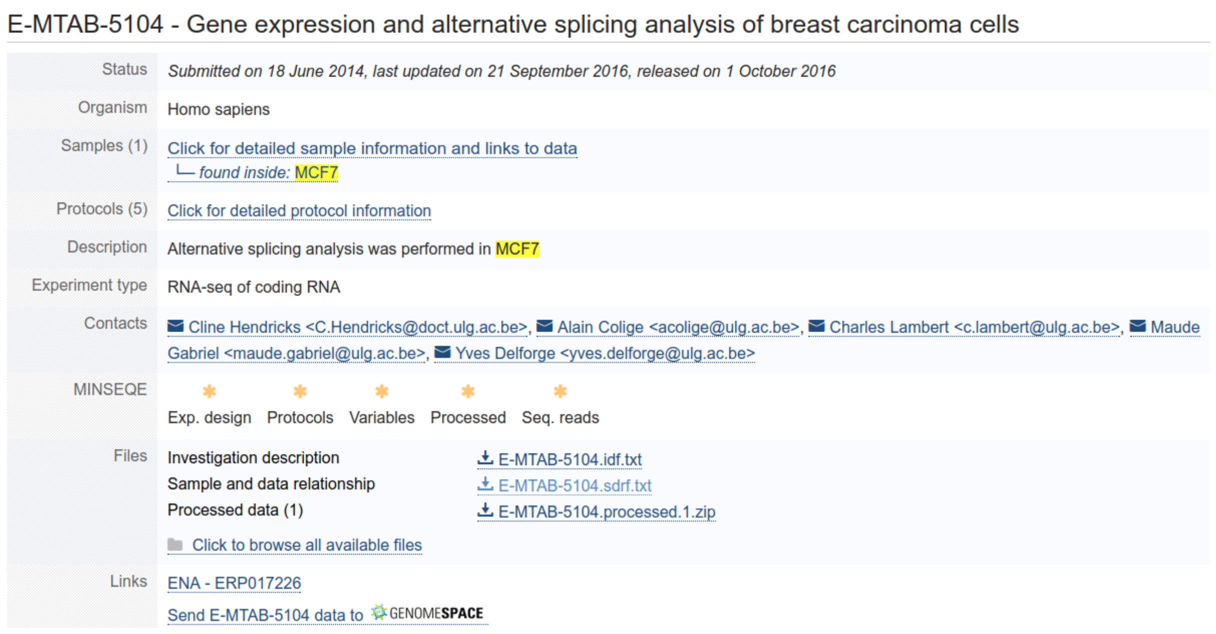
The previous selection generates a csv files that contains a number of columns with information about the samples. However, running the *rawAlignment* function only needs the three columns shown below in the csv (although the rest of the columns can be kept):

Run	download_path	LibraryLayout
SRR2753177	sra-download.ncbi.nlm.nih.gov/traces/sra21/SRR/0026...	SINGLE
SRR2753178	sra-download.ncbi.nlm.nih.gov/traces/sra21/SRR/0026...	SINGLE
SRR2753179	sra-download.ncbi.nlm.nih.gov/traces/sra21/SRR/0026...	SINGLE

There is another way to obtain this csv automatically by calling the function *downloadPublicSeries* with the NCBI/GEO GSE ID of the wanted series, but this option does not let the user to choose the wanted samples and downloads all the samples of each selected series.

3.2.2 ArrayExpress CSV format

The process for ArrayExpress is the very similar to that for NCBI/GEO. It changes the way to download the csv and the name of the columns in the file. To download the csv there is a file finished as .sdrf.txt inside the RNA-seq series in ArrayExpress, as can be seen in the example below:


E-MTAB-5104 - Gene expression and alternative splicing analysis of breast carcinoma cells

Status	Submitted on 18 June 2014, last updated on 21 September 2016, released on 1 October 2016
Organism	Homo sapiens
Samples (1)	Click for detailed sample information and links to data └ found inside: MCF7
Protocols (5)	Click for detailed protocol information
Description	Alternative splicing analysis was performed in MCF7
Experiment type	RNA-seq of coding RNA
Contacts	Cline Hendricks <C.Hendricks@doct.ulg.ac.be>, Alain Colige <acolige@ulg.ac.be>, Charles Lambert <c.lambert@ulg.ac.be>, Maude Gabriel <maude.gabriel@ulg.ac.be>, Yves Delforge <yves.delforge@ulg.ac.be>
MINSEQE	Exp. design, Protocols, Variables, Processed, Seq. reads
Files	Investigation description: E-MTAB-5104.idf.txt Sample and data relationship: E-MTAB-5104.sdrf.txt Processed data (1): E-MTAB-5104.processed.1.zip Click to browse all available files
Links	ENA - ERP017226 Send E-MTAB-5104 data to GENOMESPACE

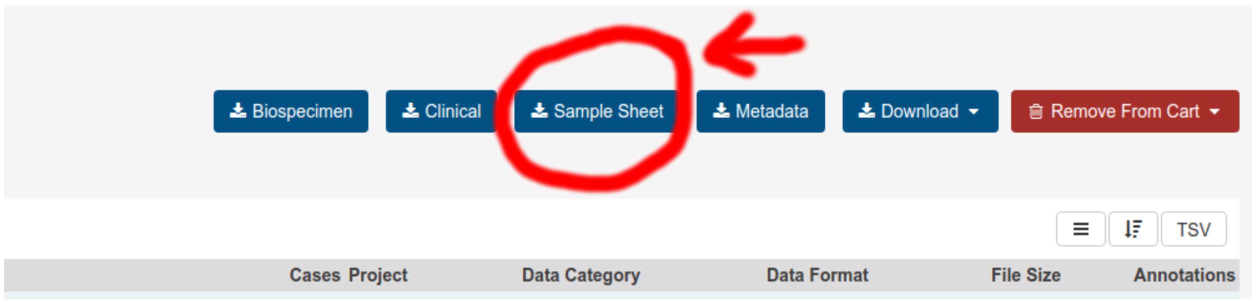
As with the NCBI/GEO csv, the csv of ArrayExpress requires only three columns as is shown below:

Comment[ENA_RUN]	Comment[FASTQ_URI]	Comment[LIBRARY_LAYOUT]
ERR1654640	ftp.sra.ebi.ac.uk/vol1/fastq/ERR165/000/ERR16…	PAIRED
ERR1654640	ftp.sra.ebi.ac.uk/vol1/fastq/ERR165/000/ERR16…	PAIRED

There is another way to achieve this csv automatically by calling the function downloadPublicSeries with the ArrayExpress MTAB ID of the wanted series, but this option does not let the user to choose the wanted samples, and therefore and downloads all the samples of each selected series.

3.2.3 GDC Portal CSV format

GDC portal has the BAM files access restricted or controlled for the user who has access to them. However, the count files are open and can be used directly in this package as input of the function countsToMatrix. If there exist the possibility to download the controlled BAM files, the tsv file that this package uses to convert them into count files is the tsv file generated when the button Sample Sheet is clicked in the cart:



As in the other two repositories, there are a lot of columns inside the tsv files but this package only needs two of them. Furthermore, if the BAM download is carried out by the gdc-client or the web browser, the BAM has to be moved to the path *ReferenceFiles/Samples/RNAseq/BAMFiles/Sample.ID/File.Name/* where Sample.ID and File.Name are the columns with the samples information in the tsv file. This folder is created automatically in the current project folder when the *rawAlignment* function is called, but it can be created manually. However, GDC portal has public access to count files that can be used in a posterior step of the KnowSeq pipeline to merge and analyze them.

3.2.4 Downloading automatically GDC Portal controlled files (GDC permission required)

It exists the possibility to download automatically the raw data from GDC portal by using the *rawAlignment* function. In order to carry this out, the function needs the parameters downloadSamples and fromGDC set to *TRUE*, the path to the token in order to obtain the authentication to download the controlled data and the path to the manifest that contains the information to download the samples. This step needs the permission of GDC portal to the controlled data.

```
# GDC portal controlled data processing with automatic raw data download

rawAlignment(x, downloadRef=TRUE, downloadSamples=TRUE, fromGDC = TRUE,
tokenPath = "~/pathToToken", manifestPath = "~/pathToManifest")
```

3.3 Preparing the count files

From now on, the data that will be used for the documentation are real count files, but with a limited number of genes (around 1000). Furthermore, to reduce the computational cost of this example, only 5 samples from each of the two selected series will be taken into account. With the next code, two RNA-seq series from NCBI/GEO are downloaded automatically and the existing count files prepared to be merged in one matrix with the purpose of preparing the data for further steps:

```
suppressMessages(library(KnowSeq))

# Downloading one series from NCBI/GEO and one series from ArrayExpress

downloadPublicSeries(c("GSE74251","GSE81593"))
```

```
##
## ******************************** NCBI/GEO format detected ********************************
##
## ******************************** RNA-SEQ series detected ********************************
##
## Building the CSV file for RNA-seq samples for series GSE74251
## Exporting CSV to ReferenceFiles folder...
## ******************************** NCBI/GEO format detected ********************************
##
## ******************************** RNA-SEQ series detected ********************************
##
## Building the CSV file for RNA-seq samples for series GSE81593
## Exporting CSV to ReferenceFiles folder...
```

```
# Using read.csv for NCBI/GEO files and read.csv2 for ArrayExpress files

GSE74251 <- read.csv("ReferenceFiles/GSE74251.csv")
GSE81593 <- read.csv("ReferenceFiles/GSE81593.csv")

GSE74251 <- GSE74251[1:5,]
GSE81593 <- GSE81593[8:12,]

dir <- system.file("extdata", package="KnowSeq")
```

```r
# Creating the csv file with the information about the counts files location and the labels
Run <- GSE74251$Run
Path <- paste(dir,"/countFiles/",GSE74251$Run,sep = "")
Class <- rep("Tumor", length(GSE74251$Run))
GSE74251CountsInfo <-  data.frame(Run = Run, Path = Path, Class = Class)

Run <- GSE81593$Run
Path <- paste(dir,"/countFiles/",GSE81593$Run,sep = "")
Class <- rep("Control", length(GSE81593$Run))
GSE81593CountsInfo <-  data.frame(Run = Run, Path = Path, Class = Class)

mergedCountsInfo <- rbind(GSE74251CountsInfo, GSE81593CountsInfo)

write.csv(mergedCountsInfo, file = "ReferenceFiles/mergedCountsInfo.csv")
```

However, the user can run a complete example by coding the following code:

```r
dir <- system.file("script", package="KnowSeq")

# Code to execute the example script
source(paste(dir,"/KnowSeqExample.R",sep=""))

# Code to edit the example script
file.edit(paste(dir,"/KnowSeqExample.R",sep=""))
```

3.4 Processing count files

After the raw alignment step, a list of count files of the samples is available at *ReferenceFiles/Samples/RNAseq/CountFiles*. The next step in the pipeline implemented in this package is the processing of those count files in order to obtain a gene expression matrix by merging all of them.

3.4.1 Merging all count files

After the alignment, there has to be as many count files as samples in the CSV used for the alignment. In order to prepare the data for the DEGs analysis, it is important to merge all these files in one matrix that contains the genes Ensembl ID (or other IDs) in the rows and the name of the samples in the columns. To carry this out, the function countsToMatrix is available. This function reads all count files and joints them in one matrix by using edgeR package [RMS10]. To call the function it is only necessary a CSV with the information about the count files paths. The required CSV has to have the following format:

Run	Path	Class
SRR2753159	~/ReferenceFile/Count/SRR2753159/	Tumor
SRR2753162	~/ReferenceFile/Count/SRR2753162/	Tumor
SRR2827426	~/ReferenceFile/Count/SRR2827426/	Healthy
SRR2827427	~/ReferenceFile/Count/SRR2827427/	Healthy

The column Run is the name of the sample without .count, the column Path is the Path to the count file and the Class column is the labels of the samples. Furthermore, an example of this function is shown below:

```r
# Merging in one matrix all the count files indicated inside the CSV file

countsInformation <- countsToMatrix("ReferenceFiles/mergedCountsInfo.csv")
```

```
##
## /tmp/Rtmp3DZGIQ/Rinst76296a2b027f/KnowSeq/extdata/countFiles/SRR2753159/SRR2753159.count
## /tmp/Rtmp3DZGIQ/Rinst76296a2b027f/KnowSeq/extdata/countFiles/SRR2753160/SRR2753160.count
## /tmp/Rtmp3DZGIQ/Rinst76296a2b027f/KnowSeq/extdata/countFiles/SRR2753161/SRR2753161.count
## /tmp/Rtmp3DZGIQ/Rinst76296a2b027f/KnowSeq/extdata/countFiles/SRR2753162/SRR2753162.count
## /tmp/Rtmp3DZGIQ/Rinst76296a2b027f/KnowSeq/extdata/countFiles/SRR2753163/SRR2753163.count
## /tmp/Rtmp3DZGIQ/Rinst76296a2b027f/KnowSeq/extdata/countFiles/SRR3541296/SRR3541296.count
## /tmp/Rtmp3DZGIQ/Rinst76296a2b027f/KnowSeq/extdata/countFiles/SRR3541297/SRR3541297.count
## /tmp/Rtmp3DZGIQ/Rinst76296a2b027f/KnowSeq/extdata/countFiles/SRR3541298/SRR3541298.count
## /tmp/Rtmp3DZGIQ/Rinst76296a2b027f/KnowSeq/extdata/countFiles/SRR3541299/SRR3541299.count
## /tmp/Rtmp3DZGIQ/Rinst76296a2b027f/KnowSeq/extdata/countFiles/SRR3541300/SRR3541300.count
## Merging 10 counts files...
```

```r
# Exporting to independent variables the counts matrix and the labels

countsMatrix <- countsInformation$countsMatrix

labels <- countsInformation$labels
```

The function returns a list that contains the matrix with the merged counts and the labels of the samples. It is very important to store the labels in a new variable because as it will be required in several functions of KnowSeq.

3.4.2 Getting the annotation of the genes

This step is only required if the user wants to get the gene names and the annotation is retrieved with ensembl biomaRt package [DSBH09]. Normally, the counts matrix has the Ensembl Ids as gene identifier, but with this step, the Ensembl Ids are change by the gene names. However, the user can decide to keep its own annotation or the Ensembl Ids. For example, to achieve the gene names the function needs the current Ensembl Ids and the number of the reference genome to use for the annotation (37 or 38). If the user wants a different annotation than the human annotation, the parameter *notHSapiens* has to be set to *TRUE* and the desired specie dataset from ensembl indicated in the parameter *notHumandataset* (i.e. "mmusculus_gene_ensembl"). An example can be seen below:

```r
# Downloading human annotation
myAnnotation <- getAnnotationFromEnsembl(rownames(countsMatrix),referenceGenome=37)
```

```
## Downloading annotation of the Homo Sapiens...
## Using reference genome 37.
```

```r
# Downloading mus musculus annotation
myAnnotationMusMusculus <- getAnnotationFromEnsembl(rownames(countsMatrix),
notHSapiens = TRUE,notHumandataset = "mmusculus_gene_ensembl")
```

```
## Downloading annotation  mmusculus_gene_ensembl ...
```

3.4.3 Converting to gene expression matrix

Finally, once both the countsMatrix and the annotation are ready, it is time to convert those counts into gene expression values. For that, the function *calculateGeneExpressionValues* uses the cqn package to calculates the equivalent gene expression [HIW12]. This function performs a conversion of counts into gene expression values, and changes the Ensembl Ids by the gene names if the parameter geneNames is equal to *TRUE*. An example of the use of this function is showed next:

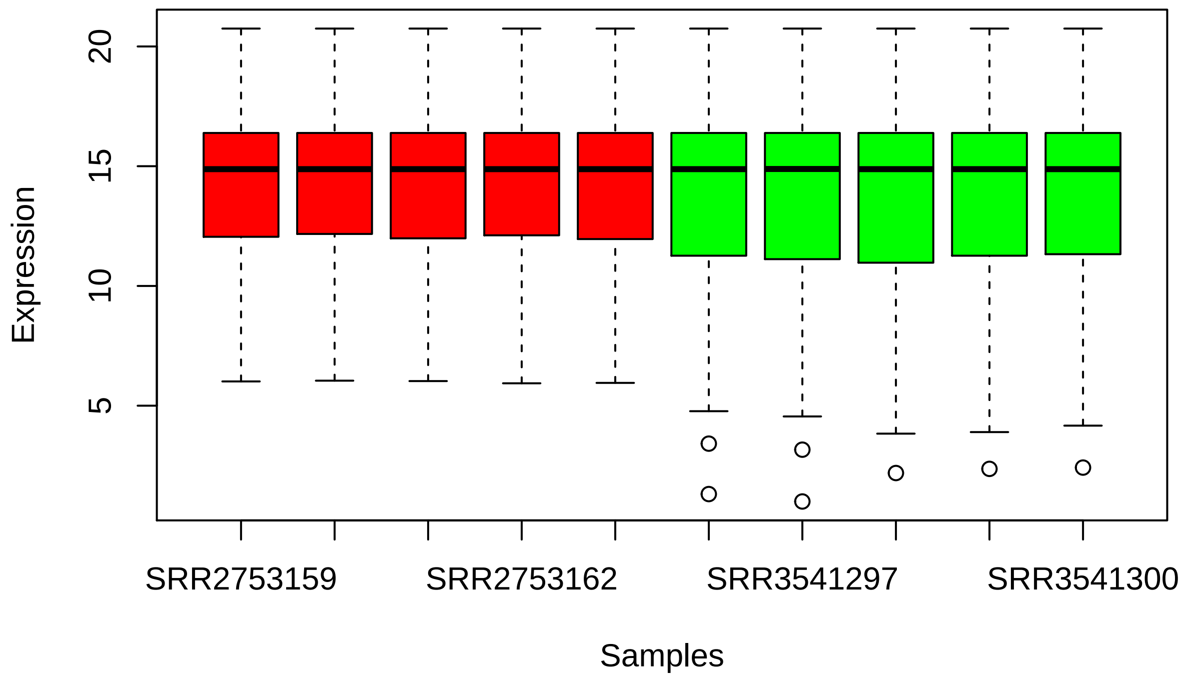
```
# Calculating gene expression values matrix using the counts matrix

expressionMatrix <- calculateGeneExpressionValues(countsMatrix,myAnnotation,
genesNames = TRUE)
```

```
## Calculating gene expression values...
## RQ fit ..........
## SQN .
```

At this time of the pipeline, there is a function that plots the expression data and allows verifying if the data is well normalized. This function has the purpose of join all the important graphical representation of the pipeline in the same function and is called *dataPlot*. It is very easy to use because as only by changing the parameter method many different representations can be achieved. In this case, in order to see the expression boxplot of each sample, the function has to be called with the parameter mode equal to *"boxplot"*. The labels are necessary to colour the different samples depending on the class of the samples. These colours can be selected by the user, by introducing in the parameter colours a vector with the name of the desired colours. The function also allows exporting the plots as PNG and PDF files.

```
# Plotting the boxplot of the expression of each samples for all the genes

dataPlot(expressionMatrix,labels,mode = "boxplot", toPNG = TRUE,
toPDF = TRUE)
```

```
## Creating PNG...

## Creating PDF...
```



4 Biomarkers identification & assessment

4.1 Quality analysis and batch effect removal

Before the DEGs extraction process, it is important to detect and removes any possible outlier that can be present in the samples. The outliers are samples numerically different with respect to the rest of samples, introducing noise in the study. In order to achieve that, the function *RNAseqQA* performs different statistical test by using arrayQualityMetrics bioc package. This package was designed for microarrays but it has been adapted in our function to allow RNA-seq data as input. The output of this function is the same as the output of the *arrayQualityMetrics* package [KGH08], creating a new folder with an index.html file including a report about the results of the different statistical tests and the possible outliers detected by each of them.

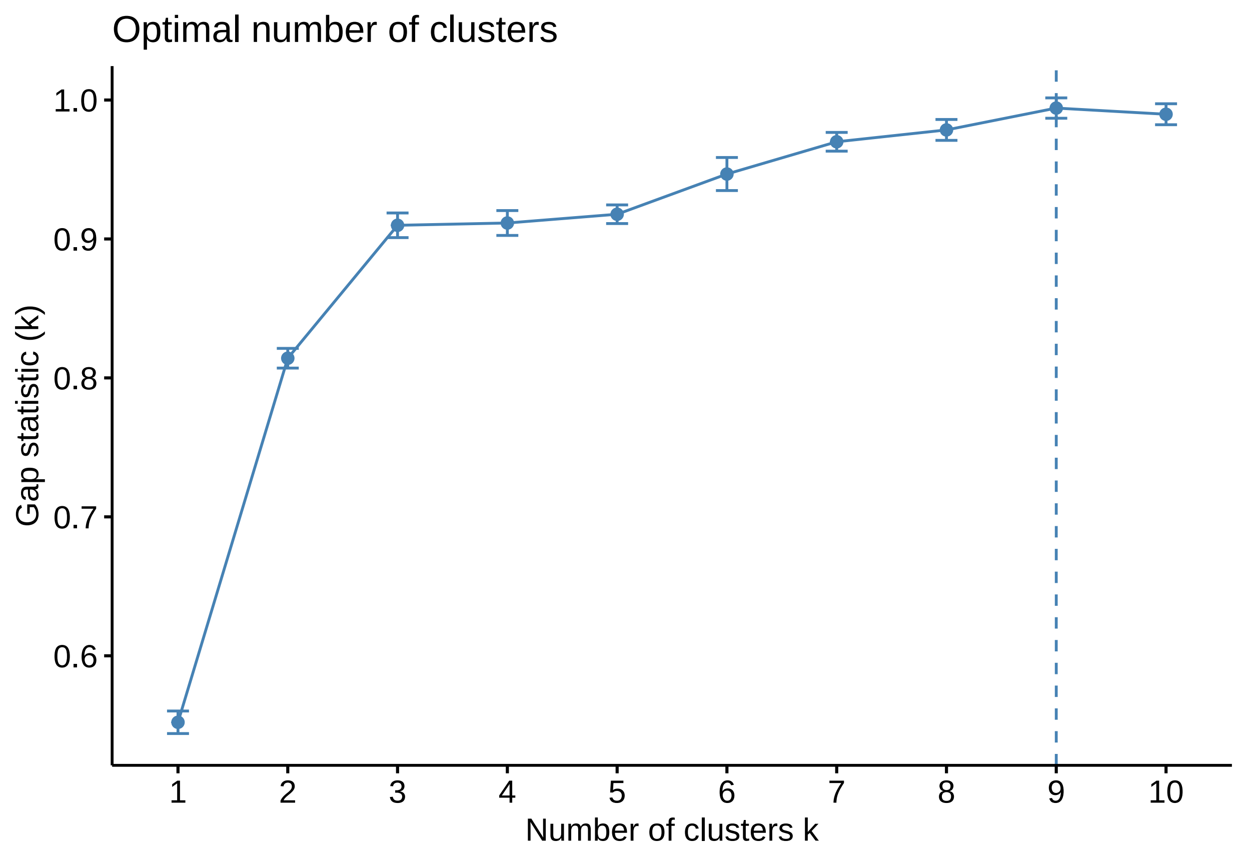
```
# Performing the quality analysis of the samples

RNAseqQA(expressionMatrix)
```

```
## Performing samples quality analysis...
## The directory 'RNAseqQA' has been created.
```

The other important step in this section is the batch effect treatment. It is widely known that this is a crucial step in the omics data processing due to the intrinsic deviations that the data can present due to its origin, sequencing design, etc... Besides, when working with public data it is very difficult to know if exists a real batch effect among the selected datasets. This package provides a way of detecting possible clusters implying possible batch effect groups and correcting them [GWW17]. If there are batch effects in the data, it will present clusters formed because of the batch effect influence. For that, first, the function *dataPlot* with the parameter mode equal to *"optimalClusters"* has to be run with the purpose of detecting the optimal number of clusters existing in the samples. Furthermore, this clusters can be represented graphically by calling the function dataPlot again but this time with the parameter method equal to *"knnClustering"*. Once the optimal number of clusters is calculated, the second and final step to remove the batch effect is by calling the function *batchEffectRemoval*, that makes use of sva package [LJP+19], with the parameter mode equal to *"combat"* and the parameter clusters equal to the optimal number of clusters calculated before. This step allows obtaining an expression matrix with the batch effect treated by combat method. An example to do this is below:

```
# Calculating the optimal number of clusters presented in the samples in order to
# try to identificate the batch effect groups to remove it by combat method

dataPlot(expressionMatrix,labels,mode = "optimalClusters",toPNG = TRUE,toPDF = TRUE)
```

```
## Creating PNG...
## Creating PDF...
```

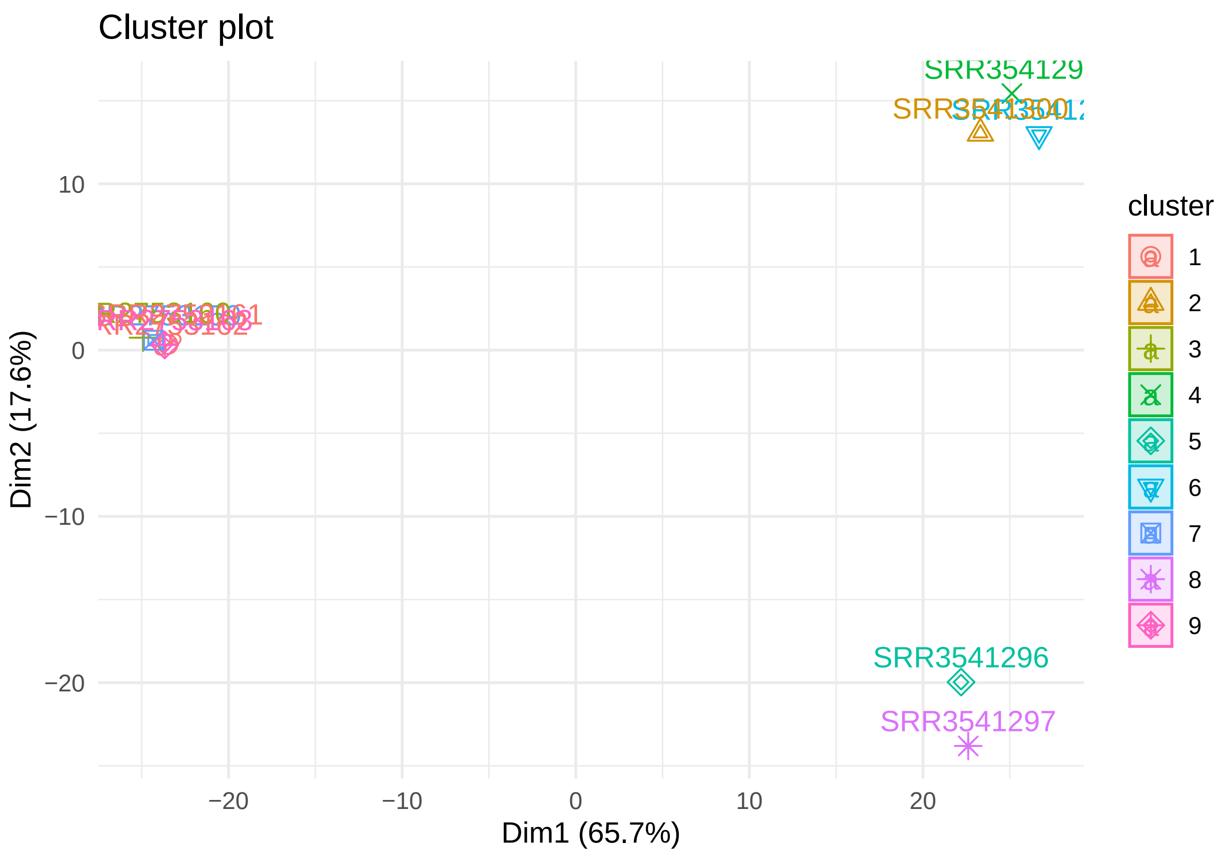


```
## pdf
##   2
```

```
dataPlot(t(expressionMatrix),labels,mode = "knnClustering", clusters = 9,toPNG = TRUE,toPDF = TRUE)
```

```
## Creating PNG...
```

```
## Creating PDF...
```



```
## pdf
##   2
```

```
expressionMatrixCorrected <- batchEffectRemoval(expressionMatrix, labels, clusters = 9,
method = "combat")
```

```
## Correcting batch effect by using combat method...

## Using the 'mean only' version of ComBat

## Found9batches

## Note: one batch has only one sample, setting mean.only=TRUE

## Adjusting for0covariate(s) or covariate level(s)

## Standardizing Data across genes

## Fitting L/S model and finding priors

## Finding parametric adjustments

## Adjusting the Data
```

There is another method in the function that removes the batch effect and it is by using surrogate variable analysis or sva. To use this method, it is not necessary to calculates the optimal number of clusters, the only requirement to use it is to set the parameter method equal to *"sva"*. This method does not return a matrix with the batch effect corrected, instead of this, the function returns a model that has to be used as single input parameter of the function *limmaDEGsExtraction*.

```
# Calculating the surrogate variable analysis to remove batch effect

svaMod <- batchEffectRemoval(expressionMatrix, labels, method = "sva")
```

```
## Calculating sva model to batch effect correction...
## Number of significant surrogate variables is:  1
## Iteration (out of 5 ):1  2  3  4  5
```

4.2 Differential Expressed Genes extraction and visualization

There is a long way between the raw data and the DEGs extraction, for that in this step the samples have to have had a strong pre-processing step applied. At this point of the pipeline the DEGs existing among two or more classes will be extracted using the most extended library for that called limma. The function *limmaDEGsExtraction* receives an expression matrix, the labels of the samples and the restriction imposed for considering a gene as differential expressed gene. The function returns a list containing the table with statistical values of each DEGs and the expression matrix of the DEGs instead all of the genes. The well-known limma package is used internally to perform the DEGs extraction [RPW+15]. The call to the function is listed below:

```r
# Extracting DEGs that pass the imposed restrictions

DEGsInformation <- limmaDEGsExtraction(expressionMatrixCorrected, labels,
lfc = 1.0, pvalue = 0.01, number = 100)

topTable <- DEGsInformation$Table

DEGsMatrix <- DEGsInformation$DEGsMatrix
```

Furthermore, if in the batch effect step the method used was sva, this function has two parameters to indicate that the model of limma would take into account the sva model calculated previously for the expression matrix. To achieve this, *svaCorrection* parameter has to be set to *TRUE* and the sva model has to be passed in the parameter *svaMod*. An example of this is the following:

```r
# Extracting DEGs that pass the imposed restrictions but using sva model
# calculated before to remove batch effect

DEGsInformation <- limmaDEGsExtraction(expressionMatrix, labels, lfc = 2.0,
pvalue = 0.01, number = Inf, svaCorrection = TRUE, svaMod = svaMod)
```

```
## Two classes detected, applying limma biclass
```

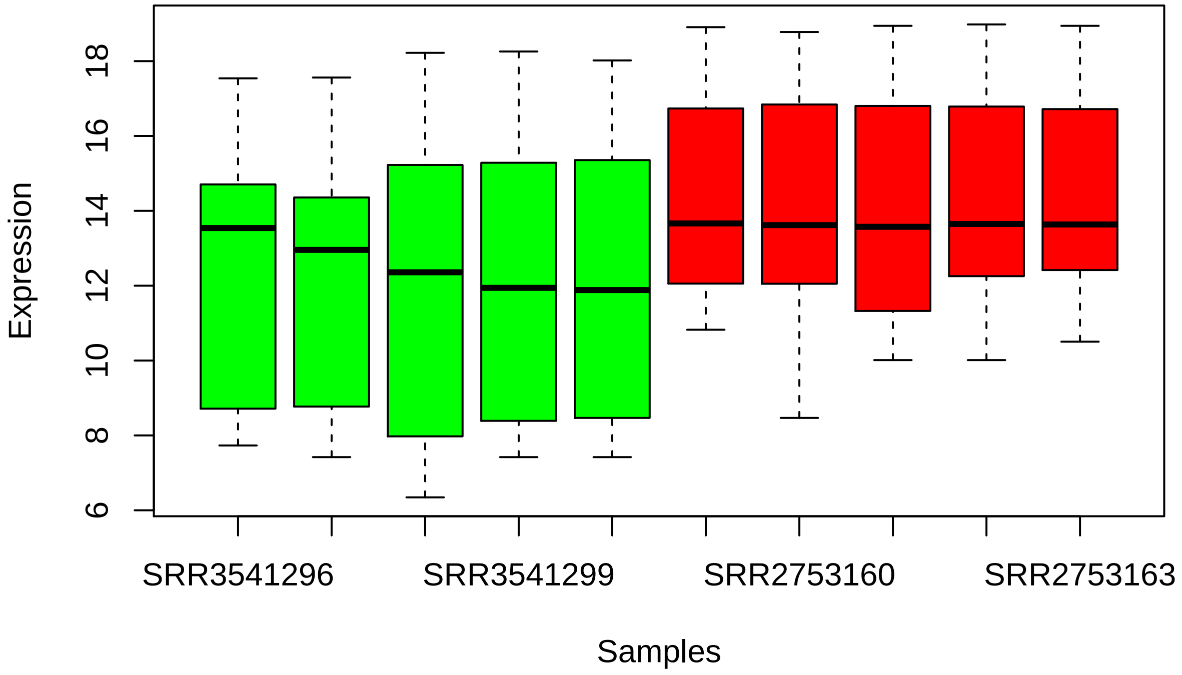
```r
topTable <- DEGsInformation$Table

DEGsMatrix <- DEGsInformation$DEGsMatrix
```

The function also detects automatically if the labels have more than two classes and calculates the limma multiclass DEGs extraction in this case. In order to do that correctly, there is a parameter called cov that represents the number of different pathologies that a certain gen is able to discern. By default, the parameter is set to *1*, so all genes that has the capability to discern among the comparison of two classes would be selected as DEGs. To understand better this parameter, our multiclass study applied to different leukemia sub-types introduces it, and it's publicly available [see CGH+19].

DEGs are genes that have a truly different expression among the studied classes, for that it is important to try to see graphically if those DEGs comply with this requirement. In order to provide a tool to perform this task, the function dataPlot encapsulate a set of graphs that allows plotting in different ways the expression of the DEGs.

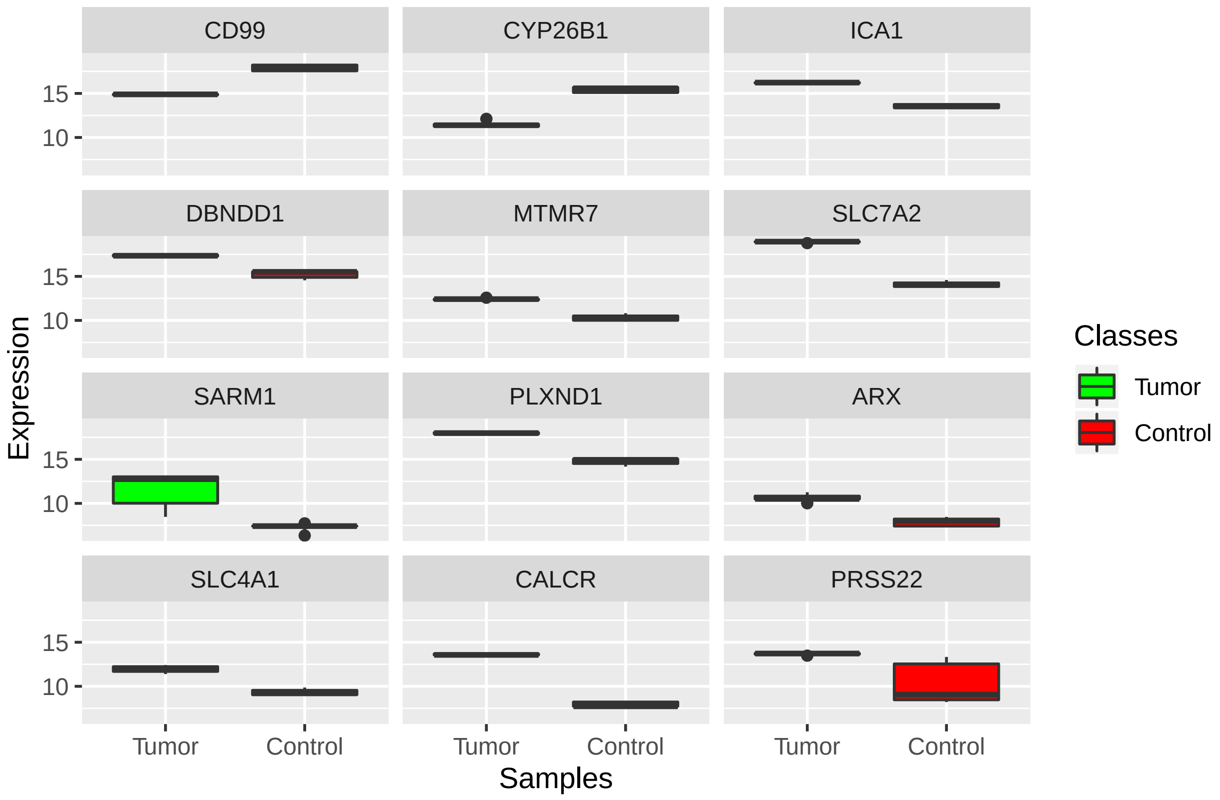
dataPlot function also allows representing an ordered boxplot that internally orders the samples by class and plots a boxplot for each samples and for the first top 12 DEGs in this example. With this plot, the difference at gene expression level between the classes can be seen graphically. The code to reproduce this plot is the following:

```
# Plotting the expression of the first 12 DEGs for each of the samples in an ordered way

dataPlot(DEGsMatrix[1:12,],labels,mode = "orderedBoxplot",toPNG = FALSE,toPDF = FALSE)
```



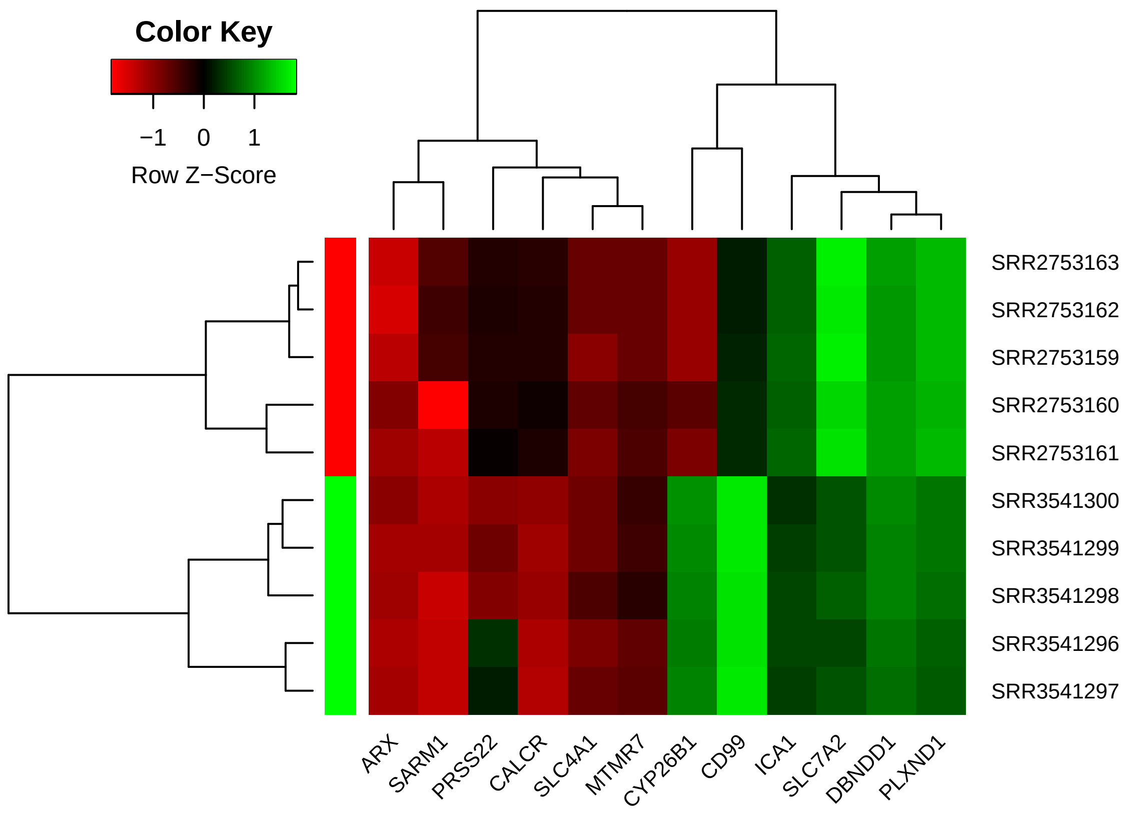
In the previous boxplot the expression of a set of DEGs for each sample its showed, however it is interesting to see the differentiation at gene expression level for each of the top 12 genes used before separately. It is recommendable to use this function with a low number of genes, because with a larger number the plot it is difficult to distinguish the information provided and R would not have enough memory to calculate the plot. For that, the function *dataPlot* with the mode *genesBoxplot* allows to do that by executing the next code:

```
# Plotting the expression of the first 12 DEGs separatelly for all the samples

dataPlot(DEGsMatrix[1:12,],labels,mode = "genesBoxplot",toPNG = FALSE,toPDF = FALSE)
```



Finally, it is possible to plot one of the most widespread visualization methods in the literature, the heatmap. By setting the parameter method to *heatmap*, the function calculates the heatmap for the given samples and classes. The code to do this is the same than for the previous boxplot but changing the method parameter:

```
# Plotting the heatmap of the first 12 DEGs separatelly for all the samples

dataPlot(DEGsMatrix[1:12,],labels,mode = "heatmap",toPNG = FALSE,toPDF = FALSE)
```



4.3 Performing the machine learning processing: classifier design and assessment and gene selection

Normally, in the literature, the last step in the pipeline for differential gene expression analysis is the DEGs extraction step. However, in this package a novel machine learning step is implemented with the purpose of giving to the user an automatic tool to assess the DEGs, and evaluate their robustness in the discernment among the studied pathologies. This library has three possible classification methodologies to take into account. These options are k-NN [PJS$^+$10], SVM [Nob06] and Random Forest [SWA08], three of the most popular classifiers in the literature. Furthermore, it includes two different working procedures for each of them. The first one implements a cross-validation process, in order to assess the expected accuracy with different models and samples the DEGs with a specific number of folds. The second one is to assess a specific test dataset by using a classifier trained using the training dataset separately. Moreover, the function featureSelection allows performing a feature selection process by using either mRMR[DP03] or Random Forest (as feature selector instead of classifier) algorithms with the purpose of finding the best DEGs order to assess the data. The functions return a list with 4 objects that contain the confusion matrices, the accuracy, the sensitivity and the specificity.

To invoke these functions, it is necessary an expression matrix with the samples in the rows and the genes in the columns and the labels of the samples, the genes that will be assessed and the number of fold in the case of the cross-validation function. In the case of the test functions, it is necessary the matrix and the labels for both the training and the test datasets:

```
DEGsMatrixML <- t(DEGsMatrix)

# Feature selection process with mRMR and RF
mrmrRanking <- featureSelection(DEGsMatrixML,labels,colnames(DEGsMatrixML), mode = "mrmr")
```

```
## Calculating the ranking of the most relevant genes by using mRMR algorithm...
## mRMR ranking: CD99 AC004381.6 COPZ2 CYP26B1 ICA1 SLC7A2 PLXND1 ARX SLC4A1 CALCR ABCB4 ITGA3 YBX2 ARH
```

```r
rfRanking <- featureSelection(DEGsMatrixML,labels,colnames(DEGsMatrixML), mode = "rf")
```

```
## Calculating the ranking of the most relevant genes by using Random Forest algorithm...
## Random Forest ranking: SLC4A1 PER3 SPATA20 RPS17P5 IL17RB MAGEC2 EXTL3 SLC7A2 LY75 FUZ AGPAT4 PPP1R3
```

```r
# CV functions with k-NN, SVM and RF
results_cv_knn <- knn_CV(DEGsMatrixML,labels,colnames(DEGsMatrixML)[1:10],5)
```

```
## Tuning the optimal K...

## Loading required package: lattice

## Loading required package: ggplot2

## Training fold 1...
## Training fold 2...
## Training fold 3...
## Training fold 4...
## Training fold 5...
## Classification done successfully!
```

```r
results_cv_svm <- svm_CV(DEGsMatrixML,labels,rfRanking[1:10],5)
```

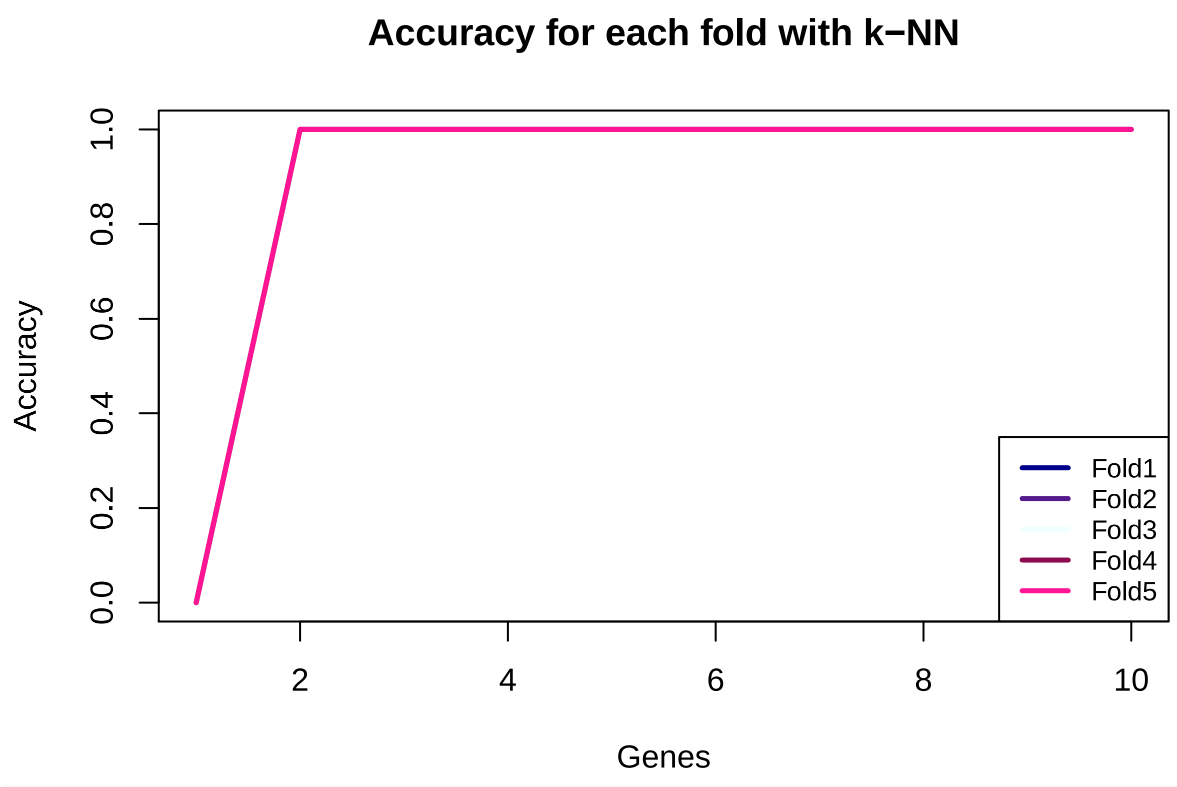
```
## Tuning the optimal C and G...
## Training fold 1...
## Training fold 2...
## Training fold 3...
## Training fold 4...
## Training fold 5...
## Classification done successfully!
```

```r
results_cv_rf <- rf_CV(DEGsMatrixML,labels,names(mrmrRanking)[1:10],5)
```

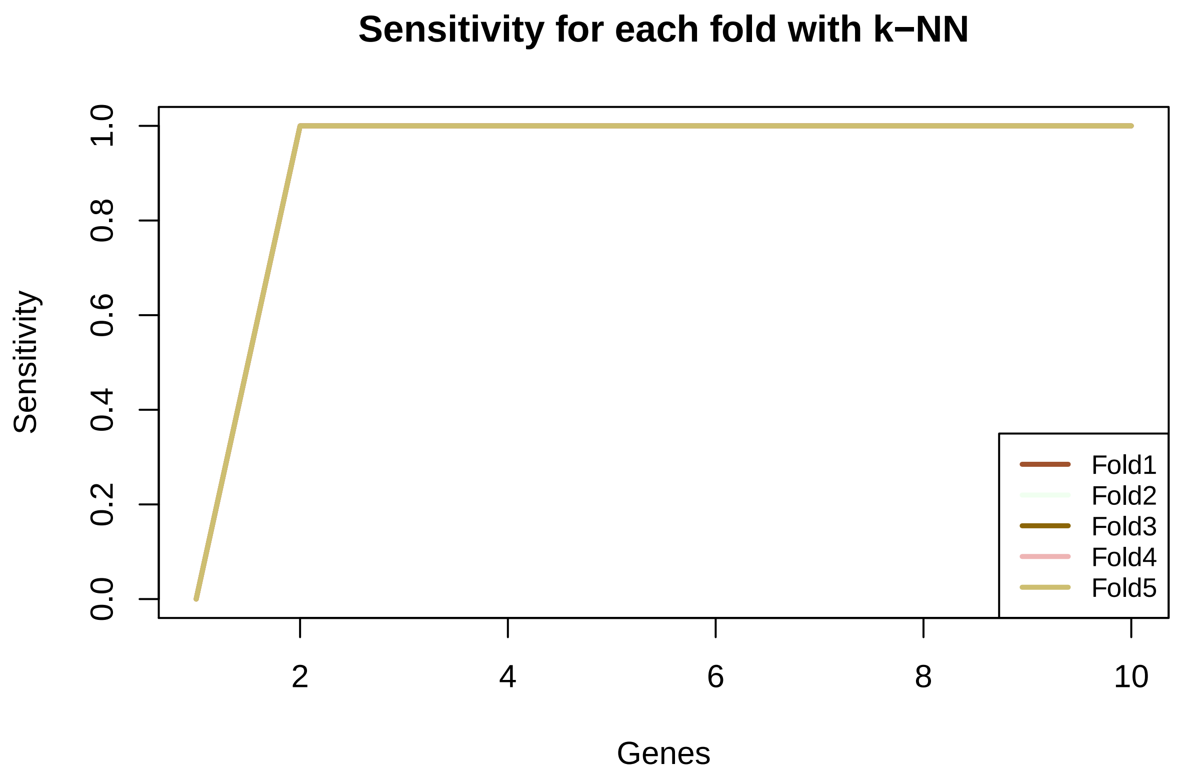
```
## Training fold 1...
## Training fold 2...
## Training fold 3...
## Training fold 4...
## Training fold 5...
## Classification done successfully!
```

It is important to show graphically the results of the classifiers and for that purpose, the function *dataPlot* implements some methods. Concretely, to plot the accuracy, the sensitivity or the specificity reached by the classifiers, the function *dataPlot* has to be run with the parameter method equal to *classResults*. This method generated as many random colours as folds or simulations in the rows of the matrix passed to the function but, through the parameter colours a vector of desired colours can be specified. For the legend, the function uses the rownames of the input matrix but these names can be changed with the parameter legend. An example of this method is showed below:

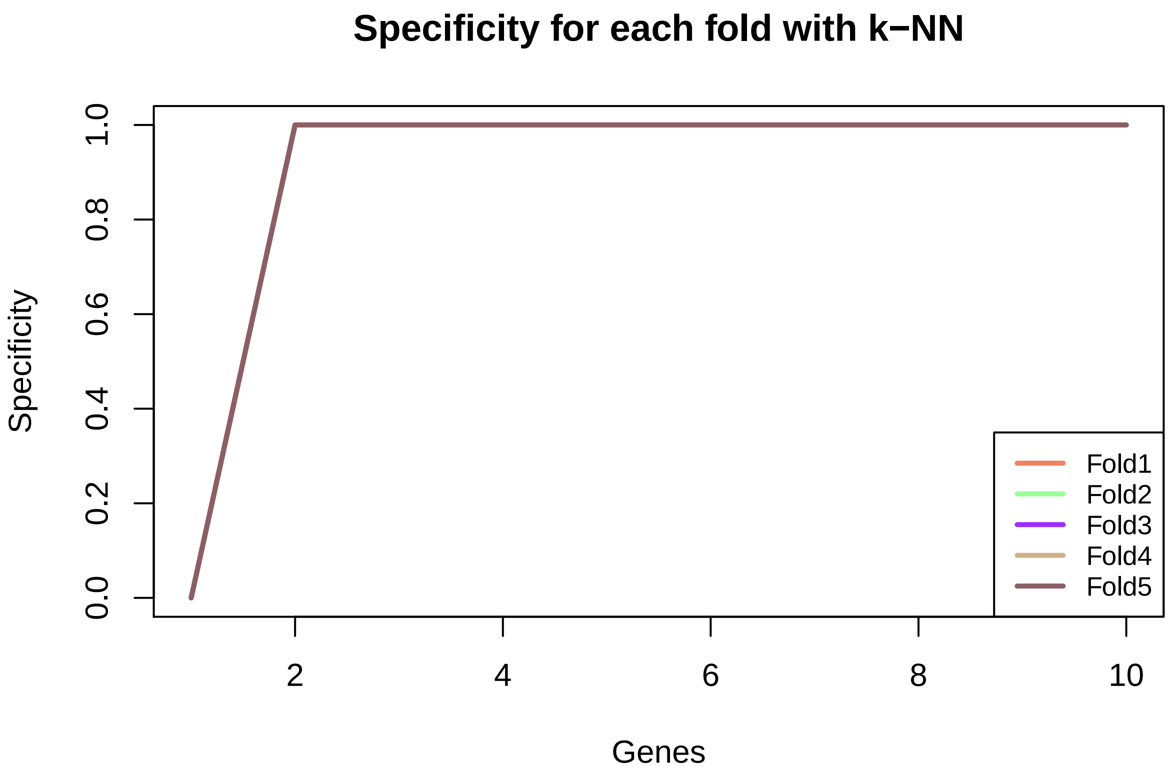
```r
# Plotting the accuracy of all the folds evaluated in the CV process

dataPlot(results_cv_knn$accMatrix,mode = "classResults",
main = "Accuracy for each fold with k-NN", xlab = "Genes", ylab = "Accuracy")
```



```r
# Plotting the sensitivity of all the folds evaluated in the CV process

dataPlot(results_cv_knn$sensMatrix,mode = "classResults",
main = "Sensitivity for each fold with k-NN", xlab = "Genes", ylab = "Sensitivity")
```

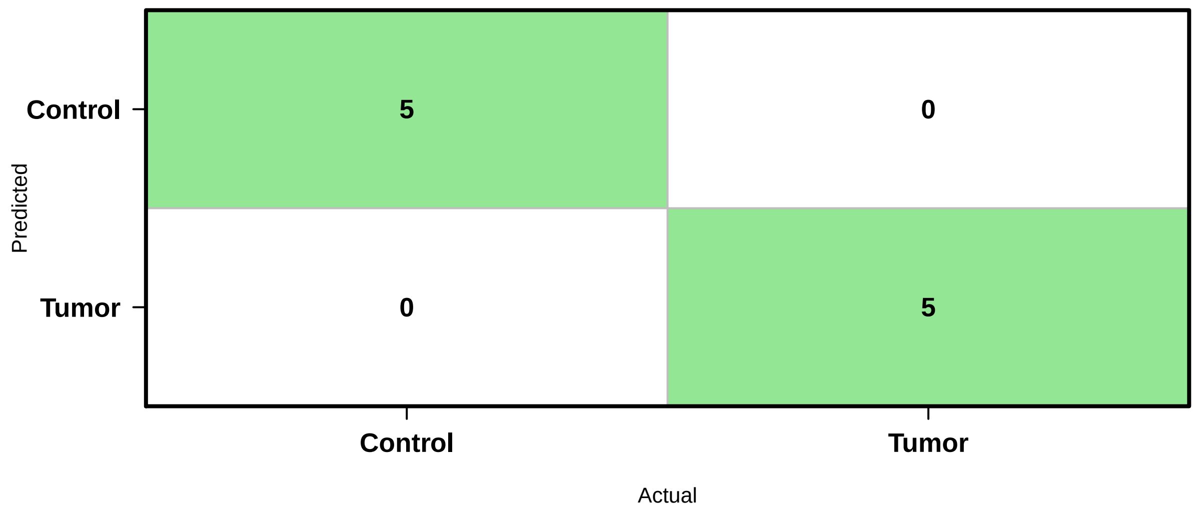


```
# Plotting the specificity of all the folds evaluated in the CV process

dataPlot(results_cv_knn$specMatrix,mode = "classResults",
main = "Specificity for each fold with k-NN", xlab = "Genes", ylab = "Specificity")
```



Furthermore, the function *dataPlot* counts with another similar mode to the previous but this time to represents confusion matrices. This mode is called *confusionMatrix* and allows creating graphically a confusion matrix with the most important statistical measures. The following code allows doing this:

```
# Plotting the confusion matrix with the sum of the confusion matrices
# of each folds evaluated in the CV process

allCfMats <- results_cv_knn$cfMats[[1]]$table + results_cv_knn$cfMats[[2]]$table +
results_cv_knn$cfMats[[3]]$table + results_cv_knn$cfMats[[4]]$table +
results_cv_knn$cfMats[[5]]$table

dataPlot(allCfMats,labels,mode = "confusionMatrix")
```



DETAILS

Accuracy	Sensitivity	Specificity
100	100	100

```r
# Test functions with k-NN, SVM and RF
trainingMatrix <- DEGsMatrixML[c(1:4,6:9),]
trainingLabels <- labels[c(1:4,6:9)]
testMatrix <- DEGsMatrixML[c(5,10),]
testLabels <- labels[c(5,10)]

results_test_knn <- knn_test(trainingMatrix, trainingLabels, testMatrix,
testLabels, names(mrmrRanking)[1:10])
```

```
## Tuning the optimal K...
## Testing with 2 variables...
## Testing with 3 variables...
## Testing with 4 variables...
## Testing with 5 variables...
## Testing with 6 variables...
## Testing with 7 variables...
## Testing with 8 variables...
## Testing with 9 variables...
## Testing with 10 variables...
## Classification done successfully!
```

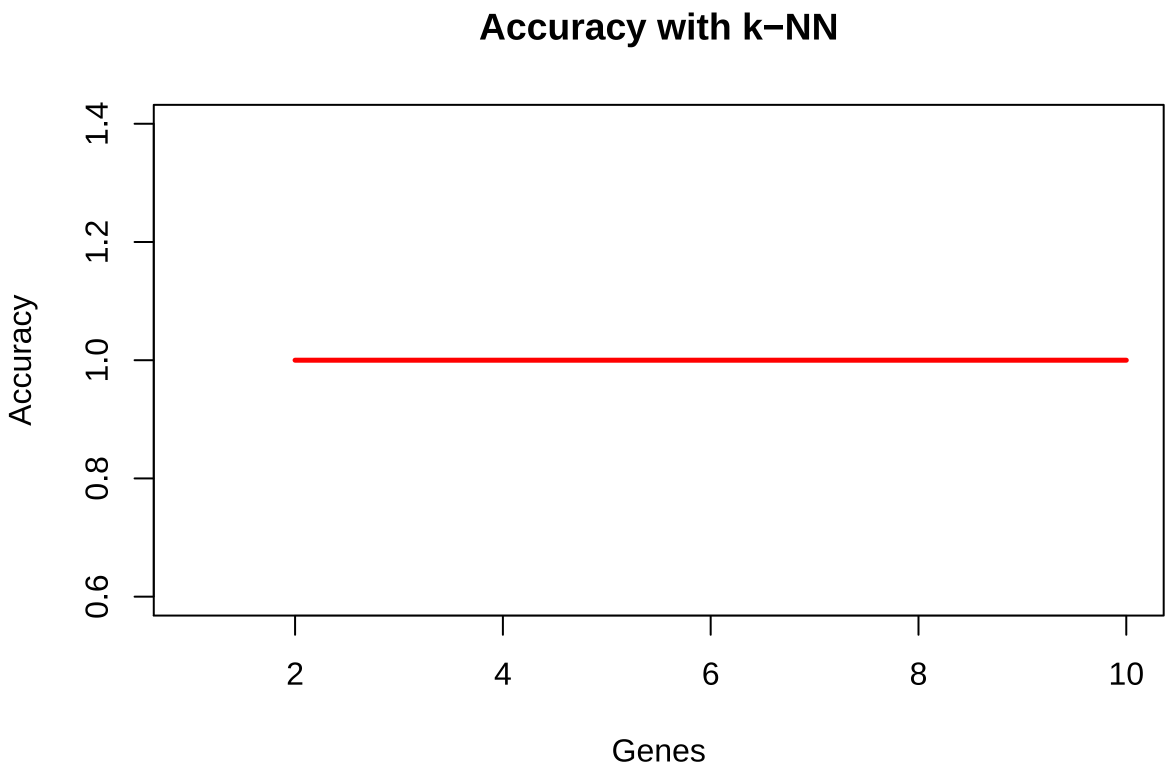
```r
results_test_svm <- svm_test(trainingMatrix, trainingLabels, testMatrix,
testLabels, rfRanking[1:10])
```

```
## Tuning the optimal C and G...
## Testing with 1 variables...
## Testing with 2 variables...
## Testing with 3 variables...
## Testing with 4 variables...
## Testing with 5 variables...
## Testing with 6 variables...
## Testing with 7 variables...
## Testing with 8 variables...
## Testing with 9 variables...
## Testing with 10 variables...
## Classification done successfully!
```

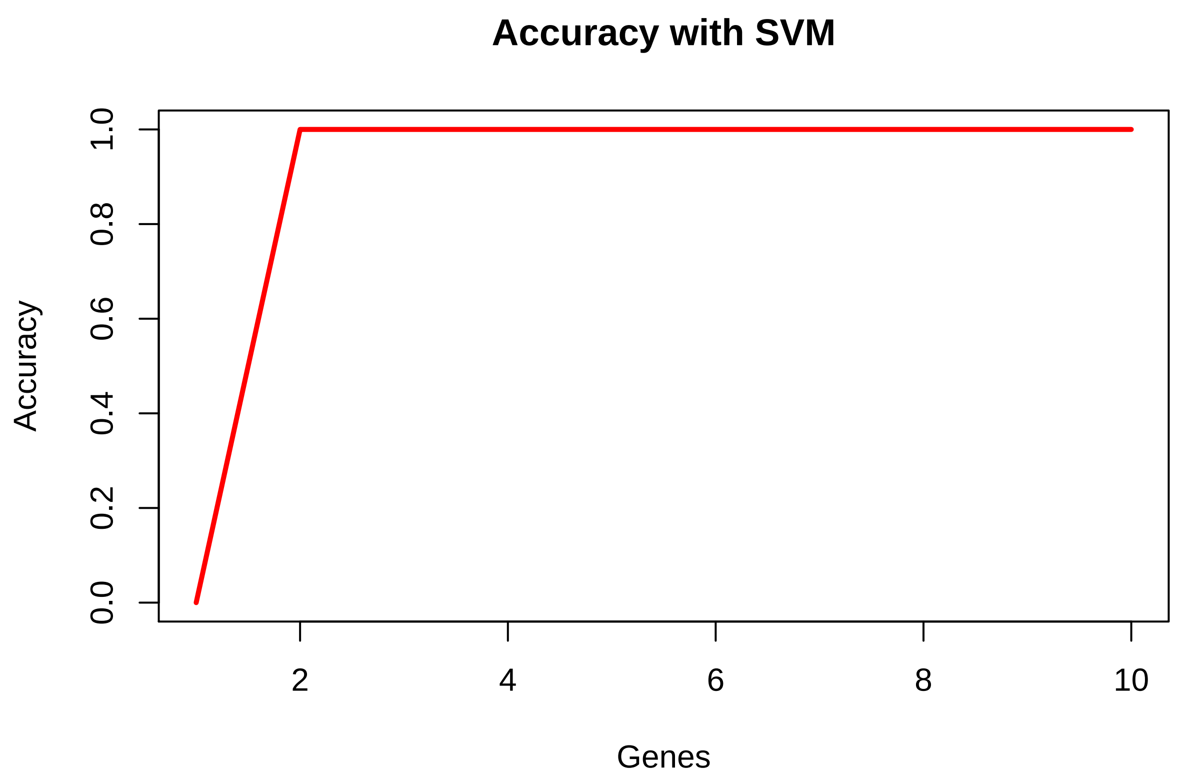
```
results_test_rf <- rf_test(trainingMatrix, trainingLabels, testMatrix,
testLabels, colnames(DEGsMatrixML)[1:10])
```

```
## Testing with 2 variables...
## Testing with 3 variables...
## Testing with 4 variables...
## Testing with 5 variables...
## Testing with 6 variables...
## Testing with 7 variables...
## Testing with 8 variables...
## Testing with 9 variables...
## Testing with 10 variables...
## Classification done successfully!
```

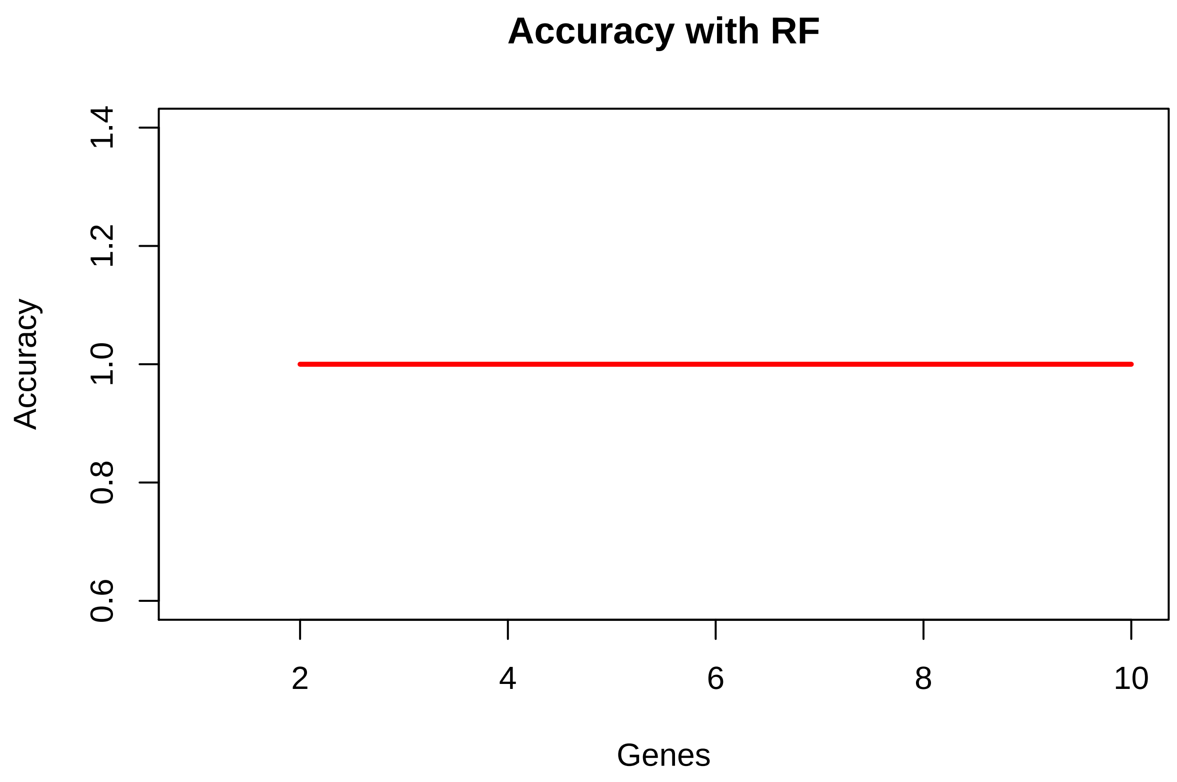
```
# Plotting the accuracy achieved in the test process

dataPlot(results_test_knn$accVector,mode = "classResults",
main = "Accuracy with k-NN", xlab = "Genes", ylab = "Accuracy")
```



```
dataPlot(results_test_svm$accVector,mode = "classResults",
main = "Accuracy with SVM", xlab = "Genes", ylab = "Accuracy")
```



```r
dataPlot(results_test_rf$accVector,mode = "classResults",
main = "Accuracy with RF", xlab = "Genes", ylab = "Accuracy")
```



5 DEGs enrichment methodology

The main goal of the previous pipeline is the extraction of biological relevant information from the DEGs. For that, this package provides a set of tools that allows doing it. The last step of the pipeline conformed by all the available tools in KnowSeq is the DEGs enrichment and this enrichment has three different points of view. The gene ontology information, the pathway visualization and the relationship between the DEGs and diseases related to the studied pathologies.

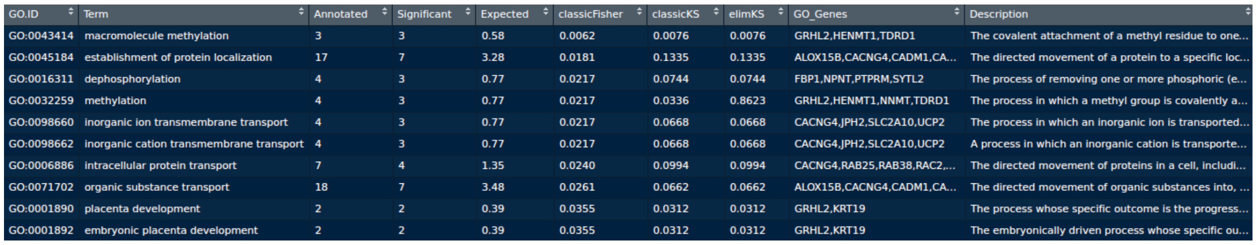
5.1 Gene Ontology

Gene ontology (GO) provides information about the biological functions of the genes. In order to complete this pipeline, it is important to know if the DEGs have functions related with the studied pathologies. In this sense, this package brings the possibility to know the GOs from the three different ontologies (BP, MF and CC) by using the function *geneOntologyEnrichment* that internally used the packaged *topGO* [AR18]. The only requirement is to put the label of first class to *1* and the label of the second class to *0*. Furthermore, with the parameter *nGOs*, the number of resultant GOs that are returned can be modified. The function returns a list that contains a matrix for each ontology and a matrix with the GOs of the three ontologies together. Moreover, the matrices have different statistical measures and the description of the functionality of each GO.

```r
# Retrieving the GO information from the three different ontologies

labelsGo <- gsub("Control",0,labels)

labelsGo <- gsub("Tumor",1,labelsGo)

GOsMatrix <- geneOntologyEnrichment(DEGsMatrix,labelsGo,nGOs = 20)
```

```
## Searching GOs from BP Ontology...Searching GOs from MF Ontology...Searching GOs from CC Ontology...G
```

For example, in this example, the top 10 GOs from the BP ontology for the extracted DEGs are shown in the following image.



GO.ID	Term	Annotated	Significant	Expected	classicFisher	classicKS	elimKS	GO_Genes	Description
GO:0043414	macromolecule methylation	3	3	0.58	0.0062	0.0076	0.0076	GRHL2,HENMT1,TDRD1	The covalent attachment of a methyl residue to one...
GO:0045184	establishment of protein localization	17	7	3.28	0.0181	0.1335	0.1335	ALOX15B,CACNG4,CADM1,CA...	The directed movement of a protein to a specific loc...
GO:0016311	dephosphorylation	4	3	0.77	0.0217	0.0744	0.0744	FBP1,NPNT,PTPRM,SYTL2	The process of removing one or more phosphoric (e...
GO:0032259	methylation	4	3	0.77	0.0217	0.0336	0.8623	GRHL2,HENMT1,NNMT,TDRD1	The process in which a methyl group is covalently a...
GO:0098660	inorganic ion transmembrane transport	4	3	0.77	0.0217	0.0668	0.0668	CACNG4,JPH2,SLC2A10,UCP2	The process in which an inorganic ion is transported...
GO:0098662	inorganic cation transmembrane transport	4	3	0.77	0.0217	0.0668	0.0668	CACNG4,JPH2,SLC2A10,UCP2	A process in which an inorganic cation is transporte...
GO:0006886	intracellular protein transport	7	4	1.35	0.0240	0.0994	0.0994	CACNG4,RAB25,RAB38,RAC2,...	The directed movement of proteins in a cell, includi...
GO:0071702	organic substance transport	18	7	3.48	0.0261	0.0662	0.0662	ALOX15B,CACNG4,CADM1,CA...	The directed movement of organic substances into, ...
GO:0001890	placenta development	2	2	0.39	0.0355	0.0312	0.0312	GRHL2,KRT19	The process whose specific outcome is the progress...
GO:0001892	embryonic placenta development	2	2	0.39	0.0355	0.0312	0.0312	GRHL2,KRT19	The embryonically driven process whose specific ou...

5.2 Pathways visualization

Another important step in the enrichment methodology in this pipeline is the pathway visualization. The function uses the DEGs to show graphically the expression of the samples in the pathways in which those genes appear. For that, the function makes use of a DEGsMatrix with the expression of the DEGs and the annotation of those DEGs in which appear the pathway or pathways of each DEGs. Internally, the function *DEGsPathwayVisualization* uses *pathview* package [LB13] to retrieve and colour the pathways, but a maximum number of 24 samples can be used, for that, if the input matrix has more than 24 samples, only the first 24 will be used by the operation. Furthermore, the function needs the expression matrix with all the genes in order to use them to colour the rest of the elements in the pathways. It is important to retrieve the annotation from Ensembl for both the DEGsMatrix and the expressionMatrix because the entrezgene IDs and the KEGG enzyme of each gene are necessary.

```r
# Downloading and filling with expression the pathways of the DEGs

myDEGsAnnotation <- getAnnotationFromEnsembl(rownames(DEGsMatrix)[1:3],
referenceGenome=38,attributes = c("external_gene_name","entrezgene_id"),
filters = "external_gene_name")
```

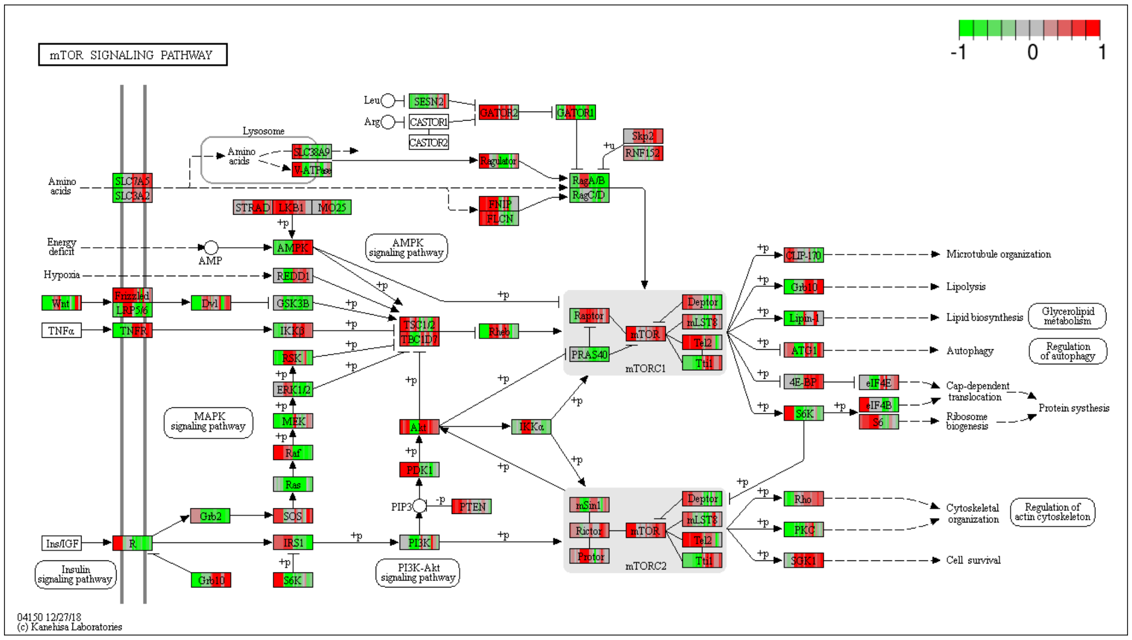
```
## Downloading annotation of the Homo Sapiens...
## Using reference genome 38.
```

```r
allMyAnnotation <-  getAnnotationFromEnsembl(rownames(expressionMatrix),
referenceGenome=38,attributes = c("external_gene_name","entrezgene_id"),
filters = "external_gene_name")
```

```
## Downloading annotation of the Homo Sapiens...
## Using reference genome 38.
```

```r
DEGsPathwayVisualization(DEGsMatrix[1:3,], myDEGsAnnotation, expressionMatrix,
allMyAnnotation, labels)
```

```
## Retrieving DEGs associated pathways...
## A total of 5 pathways have been retrieved!
## hsa04514
##  hsa04670
##  hsa00830
##  hsa01100
##  hsa04940
## This pathway cannot be processed successfully...This pathway cannot be processed successfully...This
```



5.3 Related diseases

Finally, the last enrichment method implemented is the related diseases enrichment. In this step, the function *DEGsToDisease* searchs the diseases related to a list of genes or DEGs indicated as parameter. The function returns a list of diseases only for genes and also for group of genes with several statistical values to know the relation between the diseases and the gene or group of genes. This information can be retrieved from two different web platforms: the first one is the Gene Set to Diseases [FAN16] and the second one targetValidation [KACS+16]. The web platform to use can be chosen by changing the method parameter.

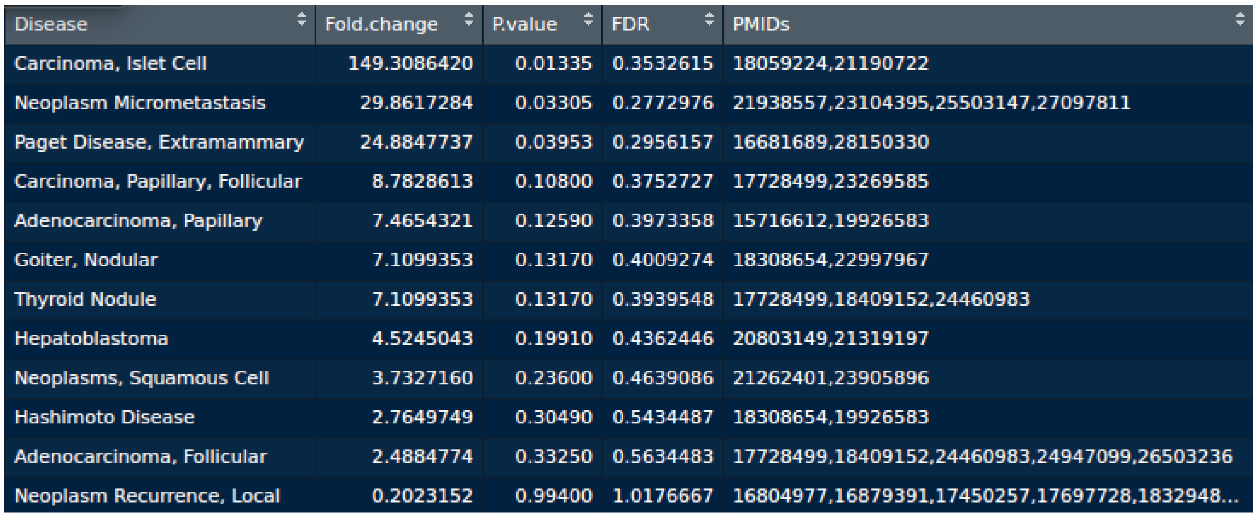
```
# Downloading the information about the DEGs related diseases

diseasesGenes2Diseases <- DEGsToDiseases(rownames(DEGsMatrix), method = "genes2Diseases", minCitation =
```

```
## Obtaining related diseases with the DEGs from genes2Diseases platform...
## Obtaining diseases related with sets of DEGs...
## Diseases acquired successfully!
```

```
diseasesTargetValidation <- DEGsToDiseases(rownames(DEGsMatrix), method = "targetValidation", size = 5)
```

```
## Obtaining related diseases with the DEGs from targetValidation platform...
## Removing AC004381.6 from the list, there is no exists associated diseases for this gene.
## Removing RPS17P5 from the list, there is no exists associated diseases for this gene.
## Diseases acquired successfully!
```

For example, the image below shows the diseases related only with KRT19 gene that the function returns with the method parameter setted to genes2Diseases.



Disease	Fold.change	P.value	FDR	PMIDs
Carcinoma, Islet Cell	149.3086420	0.01335	0.3532615	18059224,21190722
Neoplasm Micrometastasis	29.8617284	0.03305	0.2772976	21938557,23104395,25503147,27097811
Paget Disease, Extramammary	24.8847737	0.03953	0.2956157	16681689,28150330
Carcinoma, Papillary, Follicular	8.7828613	0.10800	0.3752727	17728499,23269585
Adenocarcinoma, Papillary	7.4654321	0.12590	0.3973358	15716612,19926583
Goiter, Nodular	7.1099353	0.13170	0.4009274	18308654,22997967
Thyroid Nodule	7.1099353	0.13170	0.3939548	17728499,18409152,24460983
Hepatoblastoma	4.5245043	0.19910	0.4362446	20803149,21319197
Neoplasms, Squamous Cell	3.7327160	0.23600	0.4639086	21262401,23905896
Hashimoto Disease	2.7649749	0.30490	0.5434487	18308654,19926583
Adenocarcinoma, Follicular	2.4884774	0.33250	0.5634483	17728499,18409152,24460983,24947099,26503236
Neoplasm Recurrence, Local	0.2023152	0.99400	1.0176667	16804977,16879391,17450257,17697728,1832948...

6 Session Info

```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.3 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.10-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.10-bioc/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] org.Hs.eg.db_3.10.0  multtest_2.42.0      caret_6.0-84
##  [4] ggplot2_3.2.1        lattice_0.20-38      KnowSeq_1.0.0
##  [7] topGO_2.38.0         GO.db_3.10.0         AnnotationDbi_1.48.0
## [10] IRanges_2.20.0       S4Vectors_0.24.0     Biobase_2.46.0
## [13] graph_1.64.0         BiocGenerics_0.32.0  mclust_5.4.5
## [16] quantreg_5.51        SparseM_1.77
##
## loaded via a namespace (and not attached):
##   [1] R.utils_2.9.0              tidyselect_0.2.5
##   [3] RSQLite_2.1.2              htmlwidgets_1.5.1
##   [5] beadarray_2.36.0           grid_3.6.1
##   [7] BiocParallel_1.20.0        munsell_0.5.0
##   [9] codetools_0.2-16           preprocessCore_1.48.0
##  [11] withr_2.1.2                colorspace_1.4-1
##  [13] tximportData_1.13.5        knitr_1.25
##  [15] rstudioapi_0.10            setRNG_2013.9-1
##  [17] ggsignif_0.6.0             pathview_1.26.0
##  [19] labeling_0.3               KEGGgraph_1.46.0
##  [21] tximport_1.14.0            GenomeInfoDbData_1.2.2
##  [23] hwriter_1.3.2              bit64_0.9-7
##  [25] rhdf5_2.30.0               vctrs_0.2.0
##  [27] generics_0.0.2             ipred_0.9-9
##  [29] xfun_0.10                  BiocFileCache_1.10.0
##  [31] randomForest_4.6-14        R6_2.4.0
##  [33] cqn_1.32.0                 GenomeInfoDb_1.22.0
##  [35] illuminaio_0.28.0          gridSVG_1.7-1
##  [37] locfit_1.5-9.1             bitops_1.0-6
##  [39] assertthat_0.2.1           scales_1.0.0
##  [41] nnet_7.3-12                gtable_0.3.0
##  [43] affy_1.64.0                sva_3.34.0
##  [45] timeDate_3043.102          rlang_0.4.1
##  [47] MatrixModels_0.4-1         zeallot_0.1.0
##  [49] genefilter_1.68.0          systemfonts_0.1.1
##  [51] splines_3.6.1              lazyeval_0.2.2
##  [53] ModelMetrics_1.2.2         acepack_1.4.1
##  [55] hexbin_1.27.3              checkmate_1.9.4
##  [57] BiocManager_1.30.9         yaml_2.2.0
##  [59] reshape2_1.4.3             backports_1.1.5
##  [61] Hmisc_4.2-0                tools_3.6.1
##  [63] lava_1.6.6                 nor1mix_1.3-0
##  [65] affyio_1.56.0              gplots_3.0.1.1
##  [67] RColorBrewer_1.1-2         Rcpp_1.0.2
##  [69] plyr_1.8.4                 base64enc_0.1-3
##  [71] progress_1.2.2             zlibbioc_1.32.0
##  [73] purrr_0.3.3                RCurl_1.95-4.12
##  [75] prettyunits_1.0.2          ggpubr_0.2.3
##  [77] rpart_4.1-15              openssl_1.4.1
##  [79] ggrepel_0.8.1             cluster_2.1.0
##  [81] factoextra_1.0.5          magrittr_1.5
##  [83] data.table_1.12.6         matrixStats_0.55.0
##  [85] hms_0.5.1                 praznik_7.0.0
##  [87] evaluate_0.14             xtable_1.8-4
##  [89] XML_3.98-1.20             gcrma_2.58.0
##  [91] gridExtra_2.3             compiler_3.6.1
##  [93] biomaRt_2.42.0            tibble_2.1.3
##  [95] KernSmooth_2.23-16        crayon_1.3.4
##  [97] R.oo_1.22.0               htmltools_0.4.0
##  [99] mgcv_1.8-30               Formula_1.2-3
## [101] lubridate_1.7.4           DBI_1.0.0
## [103] dbplyr_1.4.2              MASS_7.3-51.4
## [105] rappdirs_0.3.1            Matrix_1.2-17
## [107] vsn_3.54.0                R.methodsS3_1.7.1
## [109] gdata_2.18.0              gower_0.2.1
## [111] GenomicRanges_1.38.0      pkgconfig_2.0.3
## [113] foreign_0.8-72            recipes_0.1.7
## [115] foreach_1.4.7             svglite_1.2.2
## [117] annotate_1.64.0           BeadDataPackR_1.38.0
## [119] affyPLM_1.62.0            XVector_0.26.0
## [121] prodlim_2018.04.18        stringr_1.4.0
## [123] digest_0.6.22             Biostrings_2.54.0
## [125] rmarkdown_1.16            base64_2.0
## [127] htmlTable_1.13.2          edgeR_3.28.0
## [129] gdtools_0.2.1             curl_4.2
## [131] kernlab_0.9-27            gtools_3.8.1
## [133] nlme_3.1-141              jsonlite_1.6
## [135] Rhdf5lib_1.8.0            askpass_1.1
## [137] limma_3.42.0              pillar_1.4.2
## [139] KEGGREST_1.26.0           httr_1.4.1
## [141] survival_2.44-1.1         glue_1.3.1
## [143] png_0.1-7                 iterators_1.0.12
## [145] Rgraphviz_2.30.0          bit_1.1-14
## [147] class_7.3-15              stringi_1.4.3
## [149] arrayQualityMetrics_3.42.0 blob_1.2.0
## [151] latticeExtra_0.6-28       caTools_1.17.1.2
## [153] memoise_1.1.0             dplyr_0.8.3
## [155] e1071_1.7-2
```

References

[APH15] Anders, Simon ; Pyl, Paul T. ; Huber, Wolfgang: HTSeq—a Python framework to work with high-throughput sequencing data. In: *Bioinformatics* 31 (2015), Nr. 2, S. 166–169

[AR18] Alexa, Adrian ; Rahnenfuhrer, Jorg: *topGO: Enrichment Analysis for Gene Ontology. R package version 2.34.0.* 2018

[BPMP16] Bray, Nicolas L. ; Pimentel, Harold ; Melsted, Páll ; Pachter, Lior: Near-optimal probabilistic RNA-seq quantification. In: *Nature biotechnology* 34 (2016), Nr. 5, S. 525

[BWL+12] Barrett, Tanya ; Wilhite, Stephen E. ; Ledoux, Pierre ; Evangelista, Carlos ; Kim, Irene F. ; Tomashevsky, Maxim ; Marshall, Kimberly A. ; Phillippy, Katherine H. ; Sherman, Patti M. ; Holko, Michelle u. a.: NCBI GEO: archive for functional genomics data sets—update. In: *Nucleic acids research* 41 (2012), Nr. D1, S. D991–D995

[CGH+17] Castillo, Daniel ; Gálvez, Juan M. ; Herrera, Luis J. ; San Román, Belén ; Rojas, Fernando ; Rojas, Ignacio: Integration of RNA-Seq data with heterogeneous microarray data for breast cancer profiling. In: *BMC bioinformatics* 18 (2017), Nr. 1, S. 506

[CGH+19] Castillo, Daniel ; Galvez, Juan M. ; Herrera, Luis J. ; Rojas, Fernando ; Valenzuela, Olga ; Caba, Octavio ; Prados, Jose ; Rojas, Ignacio: Leukemia multiclass assessment and classification from Microarray and RNA-seq technologies integration at gene expression level. In: *PloS one* 14 (2019), Nr. 2, S. e0212127

[DP03] Ding, C. ; Peng, H.: Minimum redundancy feature selection from microarray gene expression data, 2003, S. 523–528

[DSBH09] Durinck, Steffen ; Spellman, Paul T. ; Birney, Ewan ; Huber, Wolfgang: Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. In: *Nature protocols* 4 (2009), Nr. 8, S. 1184

[FAN16] Fontaine, Jean F. ; Andrade-Navarro, Miguel A.: Gene set to diseases (gs2d): Disease enrichment analysis on human gene sets with literature data. In: *Genomics and Computational Biology* 2 (2016), Nr. 1, S. e33–e33

[GCH+18] Gálvez, Juan M. ; Castillo, Daniel ; Herrera, Luis J. ; San Román, Belén ; Valenzuela, Olga ; Ortuño, Francisco M. ; Rojas, Ignacio: Multiclass classification for skin cancer profiling based on the integration of heterogeneous gene expression series. In: *PloS one* 13 (2018), Nr. 5, S. e0196836

[GWW17] Goh, Wilson Wen B. ; Wang, Wei ; Wong, Limsoon: Why batch effects matter in omics data, and how to avoid them. In: *Trends in biotechnology* 35 (2017), Nr. 6, S. 498–507

[HIW12] Hansen, Kasper D. ; Irizarry, Rafael A. ; Wu, Zhijin: Removing technical variability in RNA-seq data using conditional quantile normalization. In: *Biostatistics* 13 (2012), Nr. 2, S. 204–216

[KACS+16] Koscielny, Gautier ; An, Peter ; Carvalho-Silva, Denise ; Cham, Jennifer A. ; Fumis, Luca ; Gasparyan, Rippa ; Hasan, Samiul ; Karamanis, Nikiforos ; Maguire, Michael ; Papa, Eliseo u. a.: Open Targets: a platform for therapeutic target identification and validation. In: *Nucleic acids research* 45 (2016), Nr. D1, S. D985–D994

[KGH08] Kauffmann, Audrey ; Gentleman, Robert ; Huber, Wolfgang: arrayQualityMetrics—a bioconductor package for quality assessment of microarray data. In: *Bioinformatics* 25 (2008), Nr. 3, S. 415–416

[KHK+14] Kolesnikov, Nikolay ; Hastings, Emma ; Keays, Maria ; Melnichuk, Olga ; Tang, Y A. ; Williams, Eleanor ; Dylag, Miroslaw ; Kurbatova, Natalja ; Brandizi, Marco ; Burdett, Tony u. a.: ArrayExpress update—simplifying data submissions. In: *Nucleic acids research* 43 (2014), Nr. D1, S. D1113–D1116

[KLS17] Kim, Daehwan ; Langmead, B ; Salzberg, S: *HISAT2: graph-based alignment of next-generation sequencing reads to a population of genomes.* 2017

[KPT+13] Kim, Daehwan ; Pertea, Geo ; Trapnell, Cole ; Pimentel, Harold ; Kelley, Ryan ; Salzberg, Steven L.: TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. In: *Genome biology* 14 (2013), Nr. 4, S. R36

[LB13] Luo, Weijun ; Brouwer, Cory: Pathview: an R/Bioconductor package for pathway-based data integration and visualization. In: *Bioinformatics* 29 (2013), Nr. 14, S. 1830–1831

[Li11] Li, Heng: A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. In: *Bioinformatics* 27 (2011), Nr. 21, S. 2987–2993

[LJP+19] Leek, JT ; Johnson, WE ; Parker, HS ; Fertig, EJ ; Jaffe, AE ; Storey, JD ; Zhang, Y ; Torres, LC: *sva: Surrogate Variable Analysis. R package version 3.30.1.* 2019

[Lov18] Love, Michael: *tximportData: tximportData. R package version 1.10.0.* 2018

[LS12] Langmead, Ben ; Salzberg, Steven L.: Fast gapped-read alignment with Bowtie 2. In: *Nature methods* 9 (2012), Nr. 4, S. 357

[Nob06] Noble, William S.: What is a support vector machine? In: *Nature Biotechnology* 24 (2006), S. 1565 – 1567

[PDL+17] Patro, Rob ; Duggal, Geet ; Love, Michael I. ; Irizarry, Rafael A. ; Kingsford, Carl: Salmon provides fast and bias-aware quantification of transcript expression. In: *Nature methods* 14 (2017), Nr. 4, S. 417

[PJS+10] Parry, RM ; Jones, W ; Stokes, TH ; Phan, JH ; Moffitt, RA ; Fang, H ; Shi, L ; Oberthuer, A ; Fischer, M ; Tong, W u. a.: k-Nearest neighbor models for microarray gene expression analysis and clinical outcome prediction. In: *The pharmacogenomics journal* 10 (2010), Nr. 4, S. 292

[RMS10] Robinson, Mark D. ; McCarthy, Davis J. ; Smyth, Gordon K.: edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. In: *Bioinformatics* 26 (2010), Nr. 1, S. 139–140

[RPW+15] Ritchie, Matthew E. ; Phipson, Belinda ; Wu, Di ; Hu, Yifang ; Law, Charity W. ; Shi, Wei ; Smyth, Gordon K.: limma powers differential expression analyses for RNA-sequencing and microarray studies. In: *Nucleic acids research* 43 (2015), Nr. 7, S. e47–e47

[SWA08] Statnikov, Alexander ; Wang, Lily ; Aliferis, Constantin F.: A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. In: *BMC bioinformatics* 9 (2008), Nr. 1, S. 319

[SX12] Sherry, Stephen ; Xiao, Chunlin: Ncbi sra toolkit technology for next generation sequence data. In: *Plant and Animal Genome XX Conference (January 14-18, 2012). Plant and Animal Genome*, 2012