MultiRNAflow: An R package for analysing RNA-seq raw counts with different time points and several biological conditions

Loubaton Rodolphe [1], ***Champagnat Nicolas*** [1], ***Vallois Pierre*** [1], ***and Vallat Laurent*** [2,3]

[1]Université de Lorraine, CNRS, Inria, IECL, F-54000 Nancy, France
[2]Université de Strasbourg, INSERM, IRFAC UMR-S1113, Strasbourg, France
[3]Department of molecular genetic of cancers, Molecular hematology unit, Strasbourg University Hospital, Strasbourg, France

October 24, 2023

Abstract

Our R package MultiRNAflow provides an easy to use unified framework allowing to make both unsupervised and supervised (DE) analysis for datasets with an arbitrary number of biological conditions and time points. In particular, our code makes a deep downstream analysis of DE information, e.g. identifying temporal patterns across biological conditions and DE genes which are specific to a biological condition for each time.

Contents

1 Introduction

1.1 Context

In eukaryotic cells, genes are expressed in the form of RNA molecules (transcription) which are then translated into proteins with a cellular function. In resting cells, at the steady state, transcription is affected by stochastic phenomena generating a transcriptional noise within cells. After modification of the cellular environment (cellular stress, receptor activation), hundreds of genes are activated, inducing a dynamic temporal transcriptional response allowing an adapted response of the cells to the initial modification of the environment [Yosef et al., 2013]. Alterations in these temporal transcriptional responses are at the origin of pathologies (infection, cancer) and are extensively studied by biologists [Bar-Joseph et al., 2012] through sometimes complex experimental designs.

Recent technological developments now make it possible to quantify the transcription of all genes in the genome by sequencing RNA molecules (RNAseq). These analyses generate raw count data whose properties (discrete data) are different from the fluorescence intensity data (continuous data) generated by previous microarray techniques. These data are presented as a table reporting, for each sample, the number of reads for each gene, obtained from the alignment of collected RNA transcripts to a reference genome (or transcriptome). The raw count of a gene (or transcript) corresponds to the number of reads mapped to the RNA sequence of this gene (or transcript).

Experimental or systematic errors may occur during sequencing (uncertainty on sequencing depth, effective library sizes, ...), which require the transformation of the original raw counts. The first methods developed consist to compute either count per million (CPM) or reads per kilobase of transcripts per million reads mapped (RPKM) [Mortazavi et al., 2008] or transcripts per million (TPM) [Li et al., 2010, Wagner et al., 2012] as they were supposed to be a good measure of RNA molar concentration (RMC) of a transcript in a sample. Although the previous methods are useful for comparing gene expressions within a sample, as they represent the relative abundance of a gene or transcript in a sample, they should not be used for comparison between samples [Zhao et al., 2020, Wagner et al., 2012] above all if samples belong to different biological conditions and/or time points.
New methods of normalization have been developed in order to be able both to compare gene expression within a sample and gene expression between samples which belong to different biological conditions and/or time points. These methods of normalization, ensure that differences between samples are only due to their membership to different biological conditions and/or time points. The most used R package for normalization are DESeq2 [Love et al., 2014] and EdgeR [Robinson et al., 2010].

The main goals of normalization of raw counts data are both to allow for unsupervised analysis of data (within samples or between samples) and to compare gene reads in different biological condition and/or time points, to detect so-called **Differentially Expressed** (DE) genes.

1.2 State of the art

Several R packages propose tools to normalize data, find DE genes and realize unsupervised analysis, such as IDEAL [Marini et al., 2020], RNASeqR [Chao et al., 2021], GEO2RNAseq [Seelbinder et al., 2019] and RNAflow [Lataretu and Hölzer, 2020].

These packages use the package DESeq2 and/or the package EdgedR in order to realize the normalization and the DE analysis. They can all detect DE genes in the case where samples belong to different biological conditions, although RNASeqR is limited to only two biological conditions.

These packages were not designed to deal with temporal data, although they could be adapted to this situation. However, none of them can analyse RNA-seq data with both several time points and several biological conditions. The package IDEAL allows to study more than two biological conditions but all the other packages are restricted only to two biological conditions.

1.3 Our R package MultiRNAflow

Our R package MultiRNAflow, built from the R package DESeq2 [Love et al., 2014], provides an easy to use unified framework allowing to automatically make both unsupervised and supervised (DE) analysis for datasets with an arbitrary number of biological conditions and time points. Specifically, our package realizes:

1. Exploratory (unsupervised) analyses of the data.
2. Statistical (supervised) analysis of transcriptional response (differentially expressed (DE) genes).
3. Analysis of temporal expression profiles of DE genes from different groups.
4. Functional analysis of temporal expression clusters.

In particular, our code makes a deep downstream analysis of DE information, e.g. identifying temporal patterns across biological conditions and DE genes which are specific to a biological condition for each time.

1.4 Supported dataset

Our package takes as input a transcriptional dataset with an experimental design that includes multiple groups of individuals and/or multiple times, t0 (reference time) and ti ($1 \leq i \leq n$). The dataset is a table of raw counts where lines correspond to genes and columns correspond to samples. Each sample show the raw counts of an individual sequencing, corresponding to a biological condition, an individual sampling in this biological condition and a time.

In this document, we will illustrate the use of our package on four examples of datasets (see section Dataset used as examples in the package) belonging to three cases:

- **Case 1**. Samples belong to different biological conditions.
- **Case 2**. Measures were realized at different time points.
- **Case 3**. Samples belong to different biological conditions and measures were realized at different time points.

1.5 Steps of the algorithm

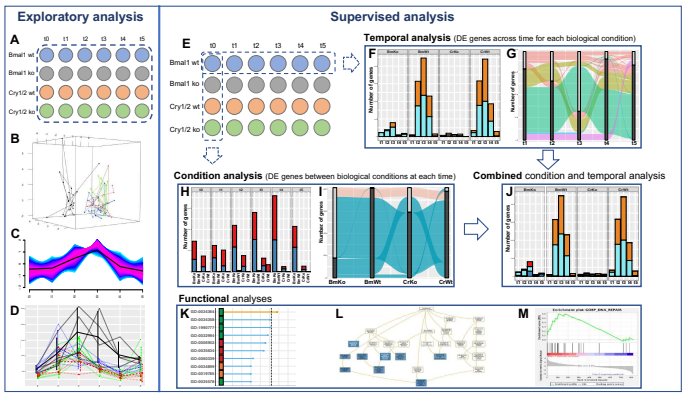


Figure 1: Outputs from the package with a dataset containing several biological conditions and several time points. The experimental design is shown in (A). Exploratory analysis includes 3D PCA (B), temporal clustering of gene expression (C) and detailed temporal gene expression (D). Supervised statistical analysis (analysis map shown in (E)) include DE genes between each time and the reference time for each biological condition (F and G); specific DE genes for each biological condition at each time (H) or at at least one time (I); signature DE genes or each biological condition and each time (J). GO enrichment analysis is realized with the R package gprofiler2 (K) or by generating input files for several GO software programs, such as Webgestalt (L) or GSEA (M).

The package MultiRNAflow realizes the following steps:

- Normalization, realized with the R package DESeq2 [Love et al., 2014].
- Exploratory data analysis (unsupervised) which includes
  - Visualization of individual patterns using factorial analysis with the R package FactoMineR [Lê et al., 2008].
  - Visualization of biological conditions and/or temporal clusters with the R package ComplexHeatmap [Gu et al., 2016]
  - Visualization of groups of genes with similar temporal behavior with the R package Mfuzz [Futschik and Carlisle, 2005, Kumar and Futschik, 2007]
- Statistical analysis (supervised) of the transcriptional response of different groups of individuals over time with the R package DESeq2 [Love et al., 2014], which includes
  - Temporal statistical analysis
  - Statistical analysis by group
  - Combination of temporal and group statistical analysis
  - Gene Ontology (GO) enrichment analysis using the R package GO enrichment with gprofiler2 [Kolberg et al., 2020] (Figure 1.K), and automatically generates outputs that can be implemented in DAVID [Sherman et al., 2022], Webgestalt [Liao et al., 2019] (Figure 1.L) and GSEA [Subramanian et al., 2005] (Figure 1.M) for further analysis using these databases.

Below, we will give a short description of each of these steps before a full description of the package outputs for four examples of datasets. The following figure illustrates the short description below. It gathers a selection of graphs produced by our package for the Mouse data 2 [Weger et al., 2021] in **case 3**. The experimental design is shown in Figure1.A and Figure 1.E. The authors analyzed the role of invalidation (ko) of two genes (Bmal1 and

Cry1/2) on murine transcriptional dynamics over the course of the day (4 groups (Bmal1 ko (invalidated gene), Bmal1 wt (wild type, not invalidated), Cry1/2 ko, Cr1/2 wt) $\times$ 4 individuals per group $\times$ 6 times = 48 transcriptional time points) (Figure 1.A).

1.5.1 Normalization

The function **DATAnormalization()** of our package allows to realize the three methods of normalization proposed in DESeq2, which must always be performed on raw counts:

- Relative log expression (rle) [Anders and Huber, 2010]. Each column of the raw counts is scaled by a specific scalar called size factors, estimated using the "median ratio method".
- Regularized logarithm (rlog) [Love et al., 2014]. This method of normalization transforms the count data to the log2 scale in a way which minimizes differences between samples for rows (so genes) with small counts. This transformation removes the dependence of the variance on the mean, particularly the high variance of the logarithm of count data when the mean is low. This method of normalization is realized by the r function `rlog()` of the R package DESeq2.
- Variance Stabilizing Transformation (vst) [Anders and Huber, 2010]. This method of normalization is similar to the rlog normalization. The vst normalization is faster but the rlog normalization is more robust in the case when the size factors vary widely. This method of normalization is realized by the r function `vst()` of the R package DESeq2.

As mentioned in the DESeq2 manual, the rle transformation is used to realize DE analysis and the rlog and vst transformations are advised for unsupervised analysis (factorial analysis, clustering) or other machine learning methods.

In addition, the function **DATAnormalization()** allows to plot the distribution of normalized read counts for each sample using boxplots.

1.5.2 Exploratory data analysis (unsupervised)

1.5.2.1 Factorial analysis

Factorial analysis is realized by the two functions **PCAanalysis()** and **HCPCanalysis()** which implement two methods: Principal Component Analysis (PCA) and Hierarchical Clustering on Principle Components (HCPC). The two methods allow to visualize the temporal evolution of the transcription within each group of individuals and similarities or differences in transcriptional behaviors between groups. Of note, the PCA visualization is optimized thanks to the dynamic 3D PCA (see Figure 1.B) allowing several viewing angles.

- **Case 1**. In the PCA graphs, samples are colored according to biological condition.
- **Case 2**. In the PCA graphs, consecutive time points within a sample are linked to help visualization of temporal patterns.
- **Case 3**. The PCA graphs combine the two previous displayings.

1.5.2.2 Visualization of groups of genes with similar temporal behavior

The R function **MFUZZanalysis()** allows, in **cases 2** and **cases 3**, to find the most common temporal behavior among all genes and all individuals in a given biological condition. This is done using the R package Mfuzz [Futschik and Carlisle, 2005, Kumar and Futschik, 2007] based on soft clustering. When there are several replicates per time, the Mfuzz package realizes soft clustering from the mean expression per time for each gene (see Figure 1.C). When there are several biological conditions, the algorithm realizes the Mfuzz analysis for each biological condition.

As in any clustering method, we need to find out the optimal number of clusters. Although a method is already implemented in the Mfuzz package, this method seems to fail when the number of genes is too big. Our function **MFUZZclusternumber()** finds the optimal number of cluster using kmeans, from the package R stats [Team, 2021] or HCPC. Among the outputs, **MFUZZclusternumber()** returns

- a graph indicating the selected number of clusters for each biological condition
- the results of the soft clustering for each biological condition

1.5.2.3 Visualization of the data

The R function **DATAplotExpressionGenes()** allows to plot gene expression profiles according to time and/or biological conditions (see Figure 1.D). The user can either use raw counts data or normalized data.

- **Case 1**. The output is a graph where are plotted: a box plot, a violin plot, and error bars (standard deviation) for each biological condition.
- **Case 2**. The output is a graph where are plotted: the evolution of the expression of each replicate across time (red lines) and the evolution of the mean and the standard deviation of the expression across time (black lines).
- **Case 3**. The output is a graph where are plotted: the evolution of the mean and the standard deviation of the expression across time for each biological condition.

1.5.3 Statistical analysis of the transcriptional response of different groups of individuals over time

1.5.3.1 Differentially expressed (DE) genes with DESeq2

The function **DEanalysisGlobal()** detects DE genes.

Case 1. Our algorithm search for differentially expressed (DE) genes between all pairs of biological conditions. This allows in particular to determine which genes are specific to each biological condition. A gene is called specific to a biological condition BC, if the gene is DE between BC and any other biological conditions, but not DE between any pairs of other biological conditions.
Among the outputs, **DEanalysisGlobal()** returns

- a Venn barplot which gives the number of genes for each possible intersection. We consider that a set of pairs of biological conditions forms an intersection if there is at least one gene which is DE for each of these pairs of biological conditions, but not for the others.

- a barplot which gives the number of specific (and over- and under-expressed, also often called up- and down-regulated) genes per biological condition.

Case 2. Our algorithm looks for differentially expressed genes between each time ti ($1 \leq i \leq n$) and the reference time t0.
Among the outputs, **DEanalysisGlobal()** returns

- an alluvial graph of differentially expressed (DE) genes.
- a Venn barplot which gives the number of genes per temporal pattern. By temporal pattern, we mean the set of times ti such that the gene is DE between ti and the reference time t0.

Case 3. Our algorithm realizes a mix of the two previous cases. First, for each biological condition, the algorithm realizes **Case 2**. Then for each time, the algorithm realizes **Case 1**. We then can find specific genes. A gene is called specific to a biological condition BC at a time ti, if the gene is DE between BC and any other biological conditions at time ti, but not DE between any pairs of other biological conditions at time ti. The algorithm also finds signature genes: a gene is called signature of a biological condition BC at a given time ti if the gene is specific for BC at time ti and DE between ti versus t0 for BC.

Among the outputs, **DEanalysisGlobal()** returns

- An alluvial graph of differentially expressed (DE) genes, for each biological condition (see Figure 1.G).
- A barplot which gives the number of DE genes per time, for each biological condition (see Figure 1.F).
- A barplot which gives the number of specific genes for each biological condition, one per time (see Figure 1.H).
- An alluvial graph of genes which are specific at at least one time, for each biological condition (see Figure 1.I).
- A graph which gives for each biological condition, the number of signature genes and non signatures genes per time ti versus the reference time t0 (see Figure 1.J).

1.5.3.2 Heatmaps, ration intensity (MA) plots and volcano plots

Clustering of samples versus genes (heatmap, not shown) allows visualization of correlations between gene expressions according to biological conditions or times. Clustering of samples versus samples allows visualization of correlations between individuals and groups. As there usually are to many genes in a dataset, the heatmaps are realized after the supervised analysis in order to reduce the number of genes for the heatmap.

1.5.3.3 Gene ontology and gene enrichment

Gene Ontology (GO) enrichment analysis is a technique for interpreting DE genes through the Gene Ontology system of classification. The goal is to retrieve a functional profile of DE genes and better understand the underlying biological processes. We recommend the most used online tools and software: GSEA [Subramanian et al., 2005], DAVID [Sherman et al., 2022], WebGestalt [Liao et al., 2019], g:Profiler [Raudvere et al., 2019], Panther [Thomas et al., 2022], ShinyGO [Ge et al., 2020], Enrichr [Kuleshov et al., 2016] and GOrilla [Eden et al., 2009]. Each of these softwares and online tools requires specific input files in order to realize their analysis. The R function **GSEApreprocessing()** automatically creates all required files. Figure 1.L and Figure 1.M were produced by the online tool WebGestalt and the software GSEA.

Alternatively, the function **GSEAquickAnalysis()** provides a GSEA analysis with the R package gprofiler2 [Kolberg et al., 2020]. Among the ouputs, **GSEAquickAnalysis()** returns

- a data.frame (output `resGSEAgprofiler2Fission$GSEAresults`) giving information about all detected gene ontologies the list of associated genes.
- a lollipop graph (Figure 2) showing the most important Gene Ontologies ranked by their $-\log_{10}(\text{pvalue})$. The y-axis indicates the `MaxNumberGO` most significant gene ontologies and pathways associated to the selected DE genes. The gene ontologies and pathways are sorted into descending order. The x-axis indicates the -log10(pvalues). The higher is a lollipop the more significant is a gene ontology or pathway. A lollipop is yellow if the pvalues is smaller than 0.05 (significant) and blue otherwise.
- A Manhattan plot (Figure 3) ranking all genes ontologies according to the functional database (G0::BP, G0::CC, G0::MF and KEGG)

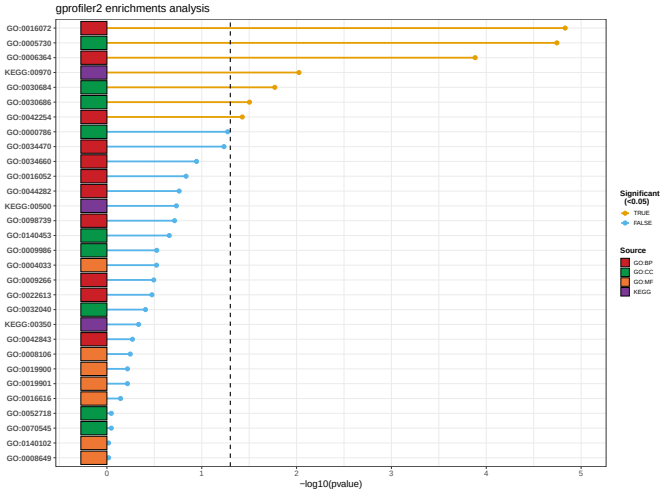


Figure 2: Lollipop chart giving the most significant gene ontologies

1.6 Dataset used as examples in the package

In order to explain the use of each function in our package, we use four datasets.

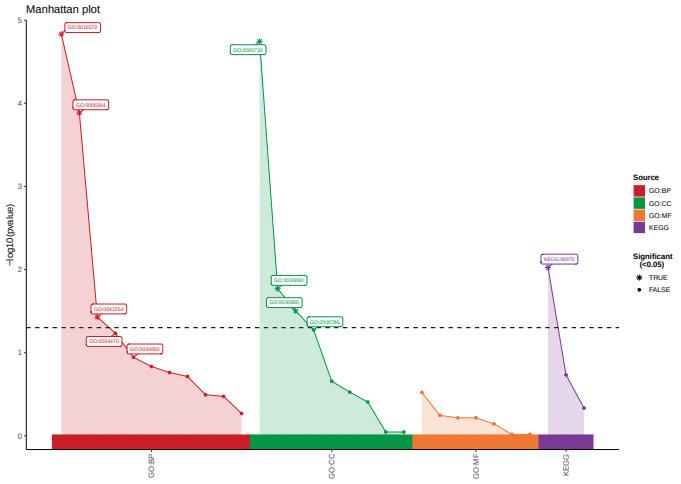


Figure 3: Mahattan plot indicating all genes ontologies

1.6.1 Mouse dataset 1

The **Mouse dataset 1** [Antoszewski et al., 2022] is accessible on the Gene Expression Omnibus (GEO) database with the accession number GSE169116.

This dataset contains the transcription profile of 12 mice belonging to 4 biological conditions:

- 3 mice with wild type Notch1 and wild type Tcf1
- 3 mice with wild type Notch1 and Tcf1 knocked-down
- 3 mice with Notch1 induced and wild type Tcf1
- 3 mice with Notch1 induced and Tcf1 knocked-down.

The dataset contains temporal expression data of 39017 genes. Notch1 is a well-established lineage specifier for T cells and among the most frequently mutated genes throughout all subclasses of T cell acute lymphoblastic leukemia [Antoszewski et al., 2022].

To illustrate the use of our package in **case 1**, we selected 500 genes giving a representative sample of each DE profile across biological conditions, in particular genes that are specific to each biological condition.
This sub dataset is saved in the file **RawCounts_Antoszewski2022_MOUSEsub500**.

1.6.2 Fission dataset

The **Fission dataset** [Leong et al., 2014] is accessible on the Gene Expression Omnibus (GEO) database with the accession number GSE56761. The dataset can also be obtained with the R package "fission" [Leong et al., 2014].

This dataset contains the temporal transcription profile of 18 wild type fission yeasts (wt) and 18 fission yeasts where atf1 is knocked-out (mut), hence 36 samples. The dataset contains temporal expression data of 7039 genes. Data were collected 0, 15, 30, 60, 120 and 180 minutes after an osmotic stress. The gene atf1 codes for a transcription factor which alters sensitivity to oxidative stress.

To illustrate the use of our package in **case 2**, we focus on the biological condition wt and select 500 genes giving a representative sample of each temporal DE profile in this biological condition. This sub dataset is saved in the file **RawCounts_Leong2014_FISSIONsub500wt**.

1.6.3 Leukemia dataset

The **Leukemia data** [Schleiss et al., 2021] is accessible on the Gene Expression Omnibus (GEO) database with the accession number GSE130385.

This dataset contains the temporal transcription profile of 3 Proliferating (P) and 3 Non Proliferating (NP) primary chronic lymphocytic leukemia (CLL) B-cells samples. Data were collected at 0, 1h, 1h30, 3h30, 6h30, 12h, 24h, 48h and 96h after cell stimulation (so $(3+3)\times 9 = 54$ samples in total). The latest time point corresponds to the emergence of the proliferation clusters. The dataset contains temporal expression data of 25369 genes.

To illustrate the use of our package in **case 3** with two biological conditions, we selected 500 genes giving a representative sample of each DE profile across time and biological conditions, in particular genes that are signature genes of each biological condition. This sub dataset is saved in the file **RawCounts_Schleiss2021_CLLsub500**.

1.6.4 Mouse data 2

The **Mouse dataset 2** [Weger et al., 2021] is accessible on the Gene Expression Omnibus (GEO) database with the accession number GSE135898.

This dataset contains the temporal transcription profile of 16 mice with Bmal1 and Cry1/2 knocked-down under an ad libitum (AL) or night restricted feeding (RF) regimen. Data were collected at 0, 4h, 8h, 16, 20h and 24h. Therefore, there are six time points and eight biological conditions. As there are only two mice per biological condition, we decided not to take into account the effect of the regimen. The dataset contains temporal expression data of 40327 genes.

To illustrate the use of our package in **case 3** with more than two biological conditions, we take 500 genes, over the global 40327 genes in the original dataset. This sub dataset is saved in the file **RawCounts_Weger2021_MOUSEsub500**.

2 Preamble

2.1 R version and R packages to install

Before installing the necessary packages, you must install (or update) the R software in a version superior to 4.2.1 "Funny-Looking Kid" (released on 2022/06/23) from CRAN (Comprehensive R Archive Network).

Then, in order to use the MultiRNAflow package, the following R packages must be installed:

- From CRAN: scales (>= 1.2.1), ggsci (>= 2.9), RColorBrewer (>= 1.1.3), reshape2 (>= 1.4.4), plyr (>= 1.8.8), ggplot2 (>= 3.4.0), ggalluvial (>= 0.12.3), ggrepel (>= 0.9.2), FactoMineR (>= 2.6), factoextra (>= 1.0.7), plot3D (>= 1.4), plot3Drgl (>= 1.0.3), UpSetR (>= 1.4.0), gprofiler2 (>= 0.2.1).
- From CRAN and usually already included by default in R: graphics (>= 4.2.2), grDevices (>= 4.2.2), grid (>= 4.2.2), methods (>= 4.2.2), stats (>= 4.2.2), utils (>= 4.2.2).
- From Bioconductor: SummarizedExperiment (>= 1.28.0), DESeq2 (>= 1.38.1), ComplexHeatmap (>= 2.14.0), Mfuzz (>= 2.58.0).

Before installing a package, for instance the package FactoMineR, the user must check if the package is already installed with the command `library(FactoMineR)`. If the package has not been previously installed, the user must use the command `install.packages("FactoMineR")` (for CRAN). For beginners in programming, we recommend to follow the steps below for importing CRAN and Bioconductor packages.

For the packages which must be download from CRAN,

```
Cran.pck<-c("scales", "reshape2", "plyr",
            "ggplot2", "ggrepel", "ggalluvial",
            "FactoMineR", "factoextra",
            "plot3D", "plot3Drgl",
            "UpSetR", "gprofiler2")
```

the user can copy and paste the following lines of code for each package in order to download the missing packages.

```
Select.package.CRAN<-"FactoMineR"
#
if(!require(package=Select.package.CRAN,
            quietly=TRUE, character.only=TRUE, warn.conflicts=FALSE)){
  install.packages(pkgs=Select.package.CRAN, dependencies=TRUE)
}# if(!require(package=Cran.pck[i], quietly=TRUE, character.only=TRUE))
```

If the package is already installed (for instance here "FactoMineR"), the previous lines of code will return nothing.

For the packages which must be download from Bioconductor,

```
Bioconductor.pck<-c("Mfuzz","SummarizedExperiment","DESeq2","ComplexHeatmap")
```

the user must first copy and paste the following lines of code in order to install "BiocManager"

```
if(!require(package="BiocManager",
            quietly=TRUE, character.only=TRUE, warn.conflicts=FALSE)){
  install.packages("BiocManager")
}# if(!require(package="BiocManager", quietly=TRUE, character.only=TRUE))
```

then copy and paste the following lines of code in order to install the version 3.16 of bioconductor (it works with R version 4.2.0)

```
BiocManager::install(version = "3.16")
```

and then copy and paste the following lines of code for each package in order to download the missing packages.

```
Select.package.CRAN<-"DESeq2"
if(!require(package=Select.package.CRAN,
            quietly=TRUE, character.only=TRUE, warn.conflicts=FALSE)){
  BiocManager::install(pkgs=Select.package.CRAN)
}# if(!require(package=Select.package.CRAN, quietly=TRUE, character.only=TRUE))
```

If the package is already installed (for instance here "DESeq2"), the previous lines of code will return nothing.

Once all packages have been installed, you may load our package.

```
library(MultiRNAflow)
```

2.2 Main functions

Our package contains 38 functions among which 11 are main functions. The user should only use

- **DATAprepSE()** to store all information about the dataset in a standardized way (SummarizedExperiment class object)
- these main functions for exploratory data analysis (unsupervised analysis)
  - **DATAnormalization()**. This function allows to normalize raw counts data and the results will be used by the functions **PCAanalysis()**, **HCPCanalysis()**, **MFUZZanalysis()** and **DATAplotExpressionGenes()**.
  - **PCAanalysis()**. The function realizes the PCA analysis.
  - **HCPCanalysis()**. The function realizes the clustering analysis with the R package HCPC.
  - **MFUZZanalysis()**. The function realizes the temporal clustering analysis with the R package Mfuzz
  - **DATAplotExpressionGenes()**. This function allows to plot the profile expression of all chosen genes.
- these main functions for supervised analysis
  - **DEanalysisGlobal()**. This function realizes the differential expression analysis.

- **DEplotVolcanoMA()**. The function plots and save all Volcano and MA plots from the output of **DEanalysisGlobal()**.
- **DEplotHeatmaps()**. The function plots a correlation heatmap and a heatmap of the normalized data for a selection of DE genes.
- **GSEAQuickAnalysis()**. The function realizes the GSEA analysis with the R package gprofiler2.
- **GSEApreprocessing()**. The function saves files to be used by 8 GSEA software and online tools.

2.3 Load of the dataset

If the user wants to use our package with in one of the dataset included in MultiRNAflow, the user must first write in the R console either

```
data("RawCounts_Antoszewski2022_MOUSEsub500")
```

in order to load the Mouse dataset 1, either

```
data("RawCounts_Leong2014_FISSIONsub500wt")
```

in order to load the Fission dataset, either

```
data("RawCounts_Schleiss2021_CLLsub500")
```

in order to load the Leukemia dataset, either

```
data("RawCounts_Weger2021_MOUSEsub500")
```

in order to load the Mouse data 2.

2.4 Structure of the dataset

The dataset must be a data.frame containing raw counts data. If it is not the case, the function **DATAprepSE()** will stop and print an error. Each line should correspond to a gene, each column to a sample, except a particular column that may contain strings of characters describing the names of the genes. The first line of the data.frame should contain the names of the columns (strings of characters) that must have the following structure.

```
data("RawCounts_Leong2014_FISSIONsub500wt")
colnames(RawCounts_Leong2014_FISSIONsub500wt)
```

```
##  [1] "Gene"     "wt_t0_r1" "wt_t0_r2" "wt_t0_r3" "wt_t1_r1" "wt_t1_r2"
##  [7] "wt_t1_r3" "wt_t2_r1" "wt_t2_r2" "wt_t2_r3" "wt_t3_r1" "wt_t3_r2"
## [13] "wt_t3_r3" "wt_t4_r1" "wt_t4_r2" "wt_t4_r3" "wt_t5_r1" "wt_t5_r2"
## [19] "wt_t5_r3"
```

In this example, "Gene" indicates the column which contains the names of the different genes. The other column names contain all kind of information about the sample, including the biological condition, the time of measurement and the name of the individual (e.g patient, replicate, mouse, yeasts culture...). Other kinds of information can be stored in the column names (such as patient information), but they will not be used by the package. The various information in the column names must be separated by underscores. The order of

these information is arbitrary but must be the same for all columns. For instance, the sample "wt_t0_r1" corresponds to the first replicate (r1) of the wild type yeast (wt) at time t0 (reference time).

The information located to the left of the first underscore will be considered to be in position 1, the information located between the first underscore and the second one will be considered to be in position 2, and so on. In the previous example, the biological condition is in position 1, the time is in position 2 and the replicate is in position 3.

In most of the functions of our package, the order of the previous information in the column names will be indicated with the inputs `Group.position`, `Time.position` and `Individual.position`. Similarly the input `Column.gene` will indicate the number of the column containing gene names. For example, in the previous dataset, the function **DATAprepSE()** must be called with the following arguments:

```
resSEexample <- DATAprepSE(RawCounts=RawCounts_Leong2014_FISSIONsub500wt,
                           Column.gene=1,
                           Group.position=NULL,
                           Time.position=2,
                           Individual.position=3)
```

Here, the argument `Column.gene=1` means that the first column of the dataset contain genes name, `Time.position=2` means that the time of measurements is between the first and the second underscores in the columns names, `Individual.position=3` means that the name of the individual is between the second and the third underscores in the columns names and `Group.position=NULL` means that there is only one biological condition in the dataset. Similarly, `Time.position=NULL` would mean that there is only one time of measurement for each individual and `Column.gene=NULL` would mean that there is no column containing gene names.

3 Preprocessing step for the four dataset with DATAprepSE()

The preprocessing step is realized by our R function **DATAprepSE()** to store all information about the dataset in a standardized way (SummarizedExperiment class object). The user must realize this step in order to realize exploratory data analysis (unsupervised analysis, section 4) or statistical analysis of the transcriptional response (supervised analysis, section 5)

3.1 Mouse data (case 1)

In this section we use the mouse subdataset **RawCounts_Antoszewski2022_MOUSEsub500** (see the subsection 1.6.1) in order to explain **DATAprepSE()** in **Case 1**.

The following lines of code realize the normalization step

```
resSEmus1500 <- DATAprepSE(RawCounts=RawCounts_Antoszewski2022_MOUSEsub500,
                           Column.gene=1,
                           Group.position=1,
                           Time.position=NULL,
                           Individual.position=2)
```

The function returns

- *SummarizedExperiment* class object containing all information of the dataset to be used for exploratory data analysis
- *DESeqDataSet* class object to be used for statistical analysis of the transcriptional response.

Write `?DATAprepSE` in your console for more information about the function.

3.2 Fission data (case 2)

In this section we use the fission yeast subdataset **RawCounts_Leong2014_FISSIONsub500wt** (see the subsection Fission dataset) in order to explain **DATAprepSE()** in **case 2**.

The following lines of code realize the normalization step

```
resSEfission500 <- DATAprepSE(RawCounts=RawCounts_Leong2014_FISSIONsub500wt,
                              Column.gene=1,
                              Group.position=NULL,
                              Time.position=2,
                              Individual.position=3)
```

The function returns

- *SummarizedExperiment* class object containing all information of the dataset to be used for exploratory data analysis
- *DESeqDataSet* class object to be used for statistical analysis of the transcriptional response.

Write `?DATAprepSE` in your console for more information about the function.

3.3 Leukemia data (Case 3 with two biological conditions)

In this section we use the Chronic lymphocytic leukemia (CLL) subdataset **RawCounts_Schleiss2021_CLLsub500** (see subsection Leukemia dataset) in order to explain **DATAprepSE()** in **case 3** when there only two biological conditions.

The following lines of code realize the normalization step.

```
resSEleuk500 <- DATAprepSE(RawCounts=RawCounts_Schleiss2021_CLLsub500,
                           Column.gene=1,
                           Group.position=2,
                           Time.position=4,
                           Individual.position=3)
```

The function returns

- *SummarizedExperiment* class object containing all information of the dataset to be used for exploratory data analysis
- *DESeqDataSet* class object to be used for statistical analysis of the transcriptional response.

Write `?DATAprepSE` in your console for more information about the function.

3.4 Mouse data 2 (case 3 with more than two biological conditions)

In this section we use the mouse subdataset **RawCounts_Weger2021_MOUSEsub500** (see Subsection Mouse dataset 1) in order to explain **DATAprepSE()** in **case 3** when there are more than two biological conditions.

The following lines of code realize the normalization step.

```
resSEmus2500 <- DATAprepSE(RawCounts=RawCounts_Weger2021_MOUSEsub500,
                           Column.gene=1,
                           Group.position=1,
                           Time.position=2,
                           Individual.position=3)
```

The function returns

- *SummarizedExperiment* class object containing all information of the dataset to be used for exploratory data analysis
- *DESeqDataSet* class object to be used for statistical analysis of the transcriptional response.

Write `?DATAprepSE` in your console for more information about the function.

4 Exploratory data analysis for the four dataset (unsupervised analysis)

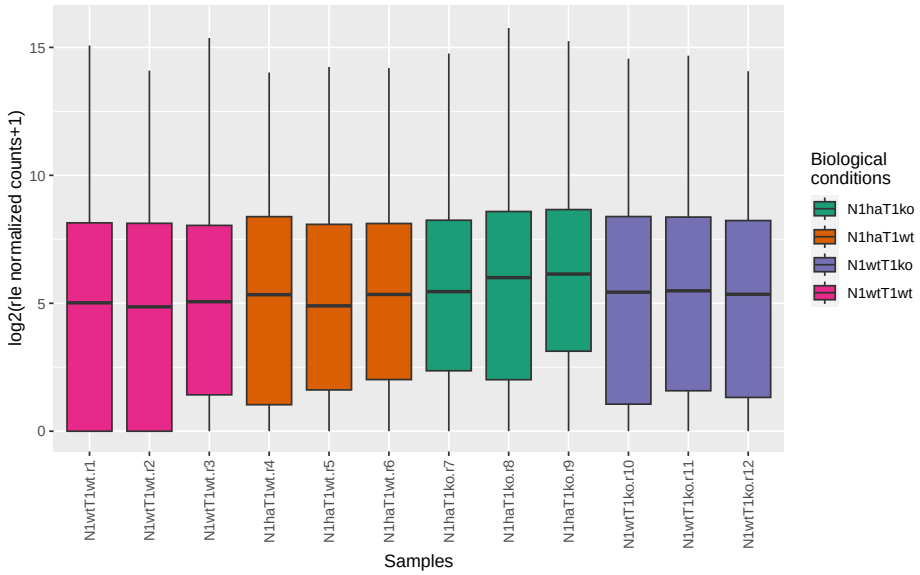
4.1 Normalization with DATAnormalization()

4.1.1 Mouse data (case 1)

In this section we use the mouse subdataset **RawCounts_Antoszewski2022_MOUSEsub500** (see the subsection 1.6.1) in order to explain **DATAnormalization()** in **Case 1**.

The following lines of code realize the normalization step from the results of the function **DATAprepSE()** (subsection **??**)

```
resNORMmus1500 <- DATAnormalization(SEres=resSEmus1500,
                                    Normalization="rle",
                                    Blind.rlog.vst=FALSE,
                                    Plot.Boxplot=TRUE,
                                    Colored.By.Factors=TRUE,
                                    Color.Group=NULL,
                                    path.result=NULL)
```



If `Plot.Boxplot=TRUE` a boxplot showing the distribution of the normalized expression (`Normalization="rle"` means that the rle method is used) of genes for each sample is returned.

If `Colored.By.Factors=TRUE`, the color of the boxplots would be different for different biological conditions. By default (if `Color.Group=NULL`), a color will be automatically applied for each biological condition. Colors can be changed the colors by creating the following data.frame

```
data.col.mus50<-data.frame(Name=c("N1wtT1wt","N1haT1wt","N1haT1ko","N1wtT1ko"),
                           Col=c("black","red","green","blue"))
```

and setting `Color.Group=data.col.mus50`.

The x-labels correspond to the new name of the samples which are just biological information, time information and individual information separated by a dot. If the user wants to see the 6th first rows of the normalized data, he can write in his console `head(res.Norm.Mus500$NormalizedData, n=6)`.

The user can save the graph in a folder thanks to the input path.result. If `path.result=NULL` the results will still be plotted but not saved in a folder.

Write `?DATAnormalization` in your console for more information about the function.

4.1.2 Fission data (case 2)

In this section we use the fission yeast subdataset **RawCounts_Leong2014_FISSIONsub500wt** (see the subsection Fission dataset) in order to explain **DATAnormalization** in **case 2**.

The following lines of code realize the normalization step from the results of the function **DATAprepSE()** (subsection 3.2)

```
resNORMyeast500 <- DATAnormalization(SEres=resSEfission500,
                                     Normalization="rlog",
                                     Blind.rlog.vst=FALSE,
                                     Plot.Boxplot=FALSE,
                                     Colored.By.Factors=TRUE,
                                     Color.Group=NULL,
                                     Plot.genes=FALSE,
                                     path.result=NULL)
```

If `Plot.Boxplot=TRUE` a boxplot showing the distribution of the normalized expression (`Normalization="rlog"` means that the rlog method is used) of genes for each sample is returned (similar to the figure of the subsection Mouse data (case 1)).

If `Colored.By.Factors=TRUE`, the color of the boxplots would be different for different time points. In case 2, the option `Color.Group` is not used. The x-labels indicate time information and individual information separated by a dot.

For more details on the use of the inputs `Column.gene`, `Time.position`, `Group.position` and `Individual.position`, see Section Structure of the dataset.

If the user wants to see the 10 first rows of the normalized data, he can write in his console `head(resNORMyeast500$NormalizedData, n=10)`.

The user chooses the path of the folder where the graph can be saved. If `path.result=NULL`, results are plotted but not saved.

Write `?DATAnormalization` in your console for more information about the function.

4.1.3 Leukemia data (Case 3 with two biological conditions)

In this section we use the Chronic lymphocytic leukemia (CLL) subdataset **RawCounts_Schleiss2021_CLLsub500** (see Subsection Leukemia dataset) in order to explain **DATAnormalization()** in **case 3** when there only two biological conditions.

The following lines of code realize the normalization step from the results of the function **DATAprepSE()** (subsection 3.3)

```
resNORMleuk500<-DATAnormalization(SEres=resSEleuk500,
                                  Normalization="vst",
```

```
                                        Blind.rlog.vst=FALSE,
                                        Plot.Boxplot=FALSE,
                                        Colored.By.Factors=TRUE,
                                        Color.Group = NULL,
                                        path.result=NULL)
```

If `Plot.Boxplot=TRUE` a boxplot showing the distribution of the normalized expression (`Normalization="vst"` means that the vst method is used) of genes for each sample is returned (similar to the figure of the subsection Mouse data (case 1)).

If `Colored.By.Factors=TRUE`, the color of the boxplots would be different for different biological conditions. By default (if `Color.Group=NULL`), a color will be automatically applied for each biological condition. You can change the colors by creating the following data.frame

```
data.col.Leuk<-data.frame(Name=c("NP","P"),
                          Col=c("black","red"))
```

and setting `Color.Group=data.col.Leuk`.

The x-labels correspond to the new name of the samples which are just biological information, time information and individual information separated by a dot. If the user wants to see the 6th first rows of the normalized data, he can write in his console `head(resNORMleuk500$NormalizedData, n=6)`.

The user can save the graph in a folder thanks to the input path.result. If `path.result=NULL` the results will still be plotted but not saved in a folder.

Write `?DATAnormalization` in your console for more information about the function.

4.1.4 Mouse data 2 (case 3 with more than two biological conditions)

In this section we use the mouse subdataset **RawCounts_Weger2021_MOUSEsub500** (see Subsection Mouse dataset 1) in order to explain **DATAnormalization()** in **case 3** when there are more than two biological conditions.

The following lines of code realize the normalization step.

```
resNORMmus2500 <- DATAnormalization(SEres=resSEmus2500,
                                    Normalization="vst",
                                    Blind.rlog.vst=FALSE,
                                    Plot.Boxplot=FALSE,
                                    Colored.By.Factors=TRUE,
                                    Color.Group=NULL,
                                    path.result=NULL)
```

If `Plot.Boxplot=TRUE` a boxplot showing the distribution of the normalized expression (`Normalization="vst"` means that the vst method is used) of genes for each sample is returned (similar to the figure of the subsection Mouse data (case 1)).

If `Colored.By.Factors=TRUE`, the color of the boxplots would be different for different biological conditions. By default (if `Color.Group=NULL`), a color will be automatically applied for each biological condition. You can change the colors by creating the following data.frame

```
data.col.Mus2 <- data.frame(Name=c("BmKo","BmWt","CrKo","CrWt"),
                            Col=c("red","blue","orange","darkgreen"))
```

and setting `Color.Group=data.col.Mus2`.

The x-labels correspond to the new name of the samples which are just biological information, time information and individual information separated by a dot. If the user wants to see the 6th first rows of the normalized data, he can write in his console `head(resNORMmus2500$NormalizedData, n=6)`

The user can save the graph in a folder thanks to the input path.result. If `path.result=NULL` the results will still be plotted but not saved in a folder.

Write `?DATAnormalization` in your console for more information about the function.

4.2 Factorial analysis: PCA with PCAanalysis() and clustering with HCPCanalysis()

4.2.1 Mouse data (case 1)

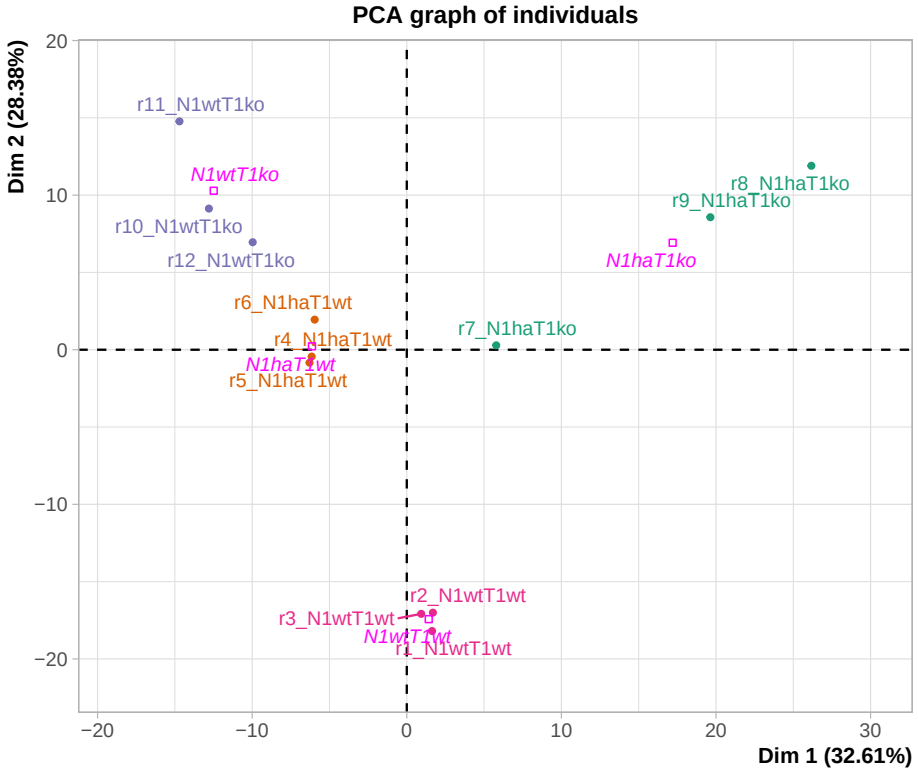
In this section we use the mouse subdataset **RawCounts_Antoszewski2022_MOUSEsub500** in order to explain **PCAanalysis()** and **HCPCanalysis()** in **case 1**.

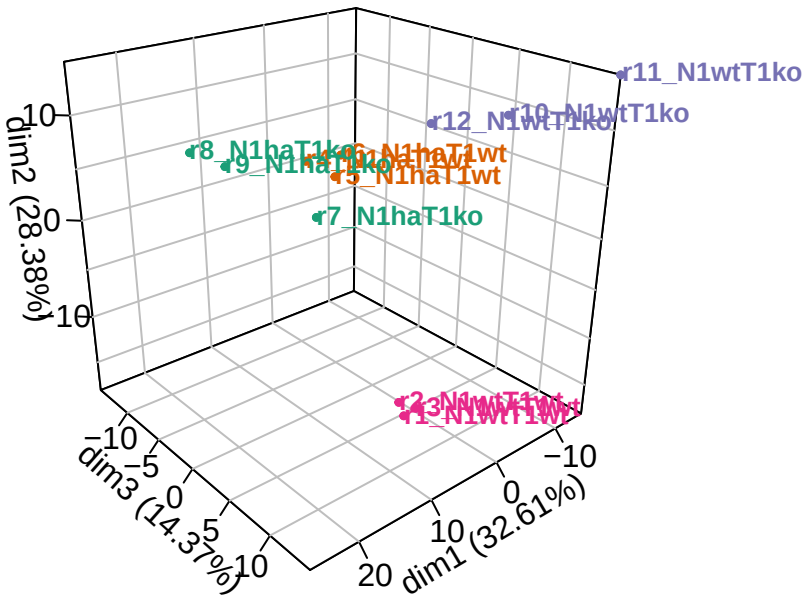
4.2.1.1 PCA (case 1)

When samples belong only to different biological conditions, the lines of code below return from the results of the function **DATAnormalization()**

- the results of the function `PCA()`
- one 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window (only if `D3.mouvement=FALSE`) where samples are colored with different colors for different biological conditions.

```
ResPCAMus500 <- PCAanalysis(SEresNorm=resNORMmus1500,
                            sample.deletion=NULL,
                            Supp.del.sample=FALSE,
                            gene.deletion=NULL,
                            Plot.PCA=TRUE,
                            Mean.Accross.Time=FALSE,
                            Color.Group = NULL,
                            Cex.label=0.8,
                            Cex.point=0.7, epsilon=0.2,
                            Phi=25,Theta=140,
                            D3.mouvement=FALSE,
                            path.result=NULL)
```





The graphs are

- displayed if `Plot.PCA=TRUE`
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

By default (if `Color.Group=NULL`), a color will be automatically assigned to each biological condition. The user can change the colors by creating the following data.frame

```
data.col.mus50<-data.frame(Name=c("N1wtT1wt","N1haT1wt","N1haT1ko","N1wtT1ko"),
                           Col=c("black","red","green","blue"))
data.col.mus50
```

```
##       Name   Col
## 1 N1wtT1wt black
## 2 N1haT1wt   red
## 3 N1haT1ko green
## 4 N1wtT1ko  blue
```

and setting `Color.Group=data.col.mus50`.

If the user wants to delete, for instance, the genes 'ENSMUSG00000064842' and 'ENSMUSG00000051951' (respectively the second and forth gene) and/or delete the samples 'N1wtT1wt_r2' and 'N1haT1wt_r5', he can set

- `gene.deletion=c("ENSMUSG00000064842","ENSMUSG00000051951")` and/or `sample.deletion=c("N1wtT1wt_r2","N1haT1wt_r5")`
- `gene.deletion=c(2,4)` and/or `sample.deletion=c(3,6)`.
  The integers in `gene.deletion` and `sample.deletion` represent respectively the row numbers and the column numbers of `RawCounts` where the selected genes and samples are located.

Write `?PCAanalysis` in your console for more information about the function.

4.2.1.2 HCPC (case 1)

The user can realize the clustering with HCPC using the function **HCPCanalysis()** as below. The lines of code below return from the results of the function **DATAnormalization()**

- The results of the function `HCPC()`
- A dendrogram
- A graph indicating by color for each sample, its cluster and the biological condition associated to the sample
- One 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window (only if `D3.mouvement=FALSE`). These PCA graphs are identical to the outputs of **PCAanalysis()** but samples are colored with different colors for different clusters.

```
ResHCPCMus500 <- HCPCanalysis(SEresNorm=resNORMmus1500,
                              gene.deletion=NULL,
                              sample.deletion=NULL,
                              Supp.del.sample=FALSE,
                              Plot.HCPC=FALSE,
                              Cex.label=0.8, Cex.point=0.7, epsilon=0.2,
                              Phi=25, Theta=140,
                              D3.mouvement=FALSE,
                              path.result=NULL)
```

The optimal number of clusters is automatically computed by the R function `HCPC()`.

Write `?HCPCanalysis` in your console for more information about the function.

4.2.2 Fission data (case 2)

In this section we use the fission yeast subdataset **RawCounts_Leong2014_FISSIONsub500wt** (see the subsection Fission dataset) in order to explain **PCAanalysis()** and **HCPCanalysis()** in **case 2**.

4.2.2.1 PCA (case 2)

In **case 2**, the lines of code below return from the results of the function **DATAnormalization()**

- the results of the function `PCA()`
- one 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window (only if `D3.mouvement=FALSE`) where samples are colored with different colors for different time points. Furthermore, lines are drawn in gray between each pair of consecutive points for each individual.
- the same graphs as above but without lines (not shown).

```
ResPCAYeast500 <- PCAanalysis(SEresNorm=resNORMyeast500,
                              gene.deletion=NULL,
                              sample.deletion=NULL,
                              Supp.del.sample=FALSE,
                              Plot.PCA=FALSE,
                              Mean.Accross.Time=FALSE,
                              Cex.label=0.8, Cex.point=0.7, epsilon=0.3,
                              Phi=25,Theta=140,
                              D3.mouvement=FALSE,
                              path.result=NULL)
```

These figures show that the temporal behavior is similar between individuals.

If the user wants for instance, to perform the PCA analysis without the genes 'SPAC212.11' and 'SPNCRNA.70' (first and third gene) and/or without the samples 'wt_t0_r2' and 'wt_t1_r1', he can set

- either `gene.deletion=c("SPAC212.11","SPNCRNA.70")` and/or `sample.deletion=c("wt_t0_r2","wt_t1_r1")`,
- or `gene.deletion=c(1,3)` and/or `sample.deletion=c(3,5)`.
  The integers in `gene.deletion` and `sample.deletion` represent respectively the row numbers (resp. the column numbers) of `RawCounts` corresponding to genes (resp. samples) that need to be removed from `RawCounts`.

In **case 2**, the user cannot select a color for each time. A specific palette is automatically used.

Write `?PCAanalysis` in your console for more information about the function.

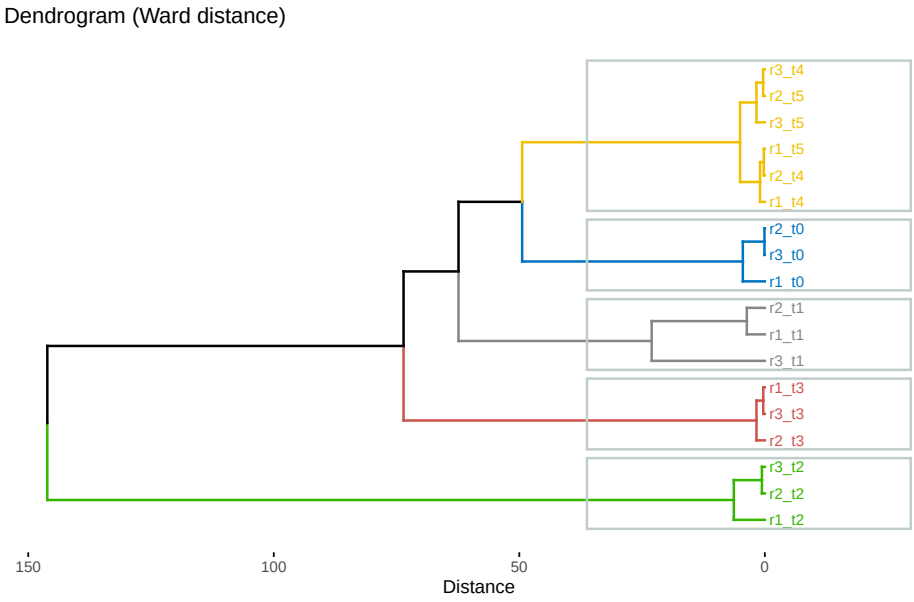
4.2.2.2 HCPC (case 2)

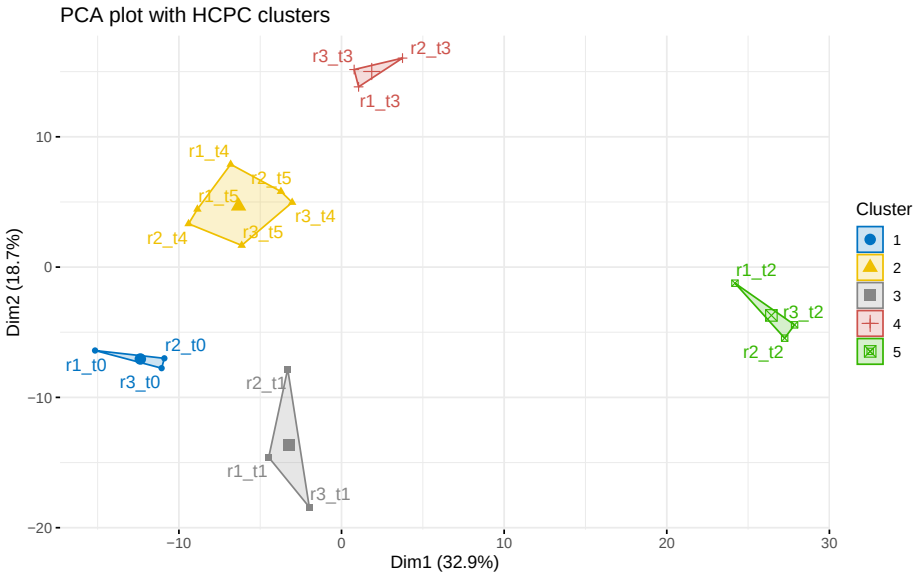
The lines of code below return from the results of the function **DATAnormalization()**

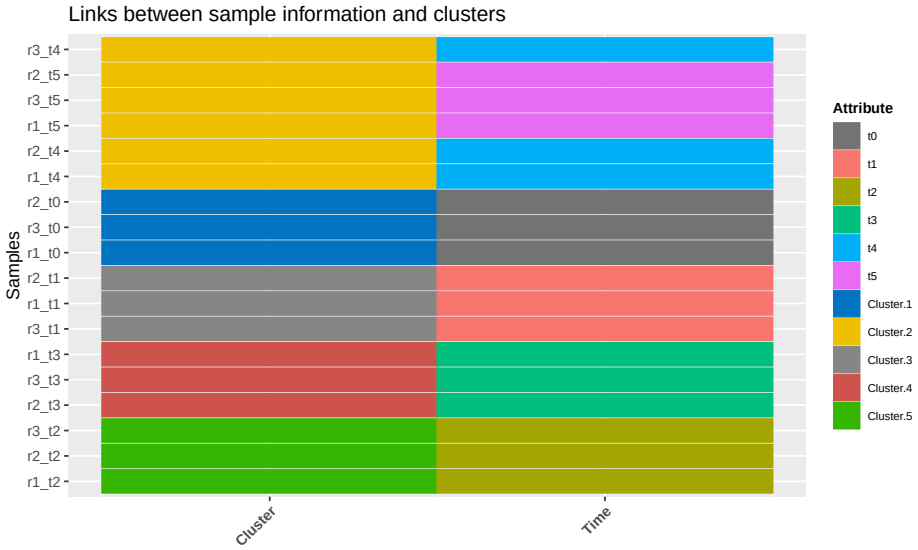
- The results of the function `HCPC()`

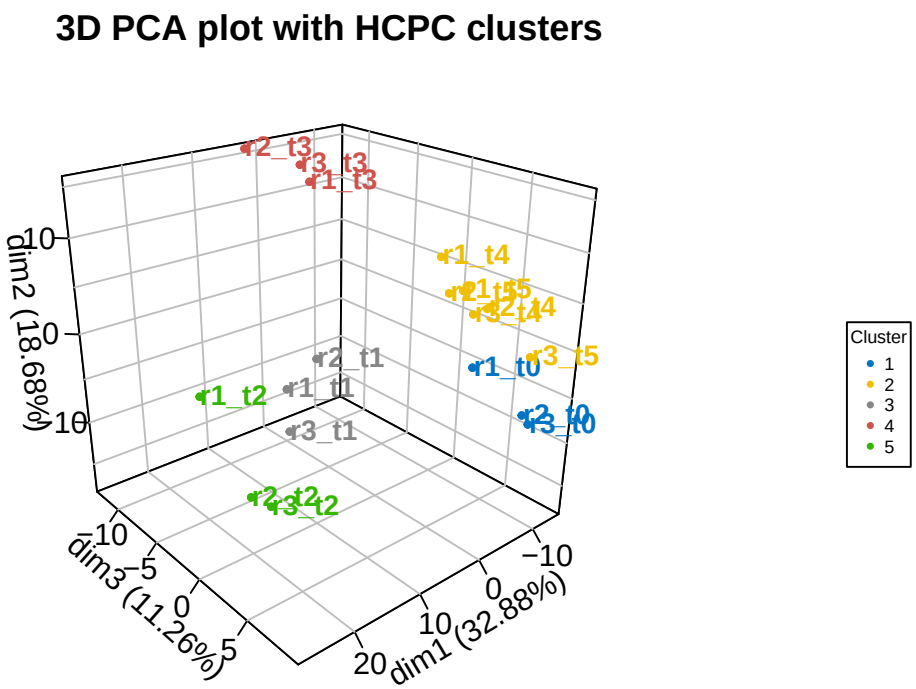
- A dendrogram
- A graph indicating by color for each sample, its cluster and the time point associated to the sample.
- one 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window (only if `D3.mouvement=FALSE`). These PCA graphs are identical to the outputs of **PCAanalysis()** but samples are colored with different colors for different HCPC clusters.

```
ResHCPCYeast500 <- HCPCanalysis(SEresNorm=resNORMyeast500,
                                gene.deletion=NULL,
                                sample.deletion=NULL,
                                Supp.del.sample=FALSE,
                                Plot.HCPC=TRUE,
                                Cex.label=0.9,
                                Cex.point=0.7,
                                epsilon=0.2,
                                Phi=25,Theta=140,
                                D3.mouvement=FALSE,
                                path.result=NULL)
```




PCA plot with HCPC clusters
r3_t3
r2_t3
r1_t3
r1_t4
r2_t5
r1_t5
r3_t4
r2_t4
r3_t5
r1_t2
r3_t2
r2_t2
r2_t0
r1_t0
r3_t0
r2_t1
r1_t1
r3_t1
Dim2 (18.7%)
Dim1 (32.9%)
Cluster
1
2
3
4
5


Links between sample information and clusters
Samples
Cluster
Time
Attribute
t0
t1
t2
t3
t4
t5
Cluster.1
Cluster.2
Cluster.3
Cluster.4
Cluster.5



The graphs are

- displayed if `Plot.HCPC=TRUE`
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

This function has similar inputs as **PCAanalysis()** and the optimal number of clusters is automatically computed by the R function `HCPC()`.

Write `?HCPCanalysis` in your console for more information about the function.

4.2.3 Leukemia data (case 3 with only two biological conditions)

In this section we use the Chronic lymphocytic leukemia (CLL) subdataset **RawCounts_Schleiss2021_CLLsub500** (see Subsection Leukemia dataset) in order to explain **PCAanalysis()** and **HCPCanalysis()** in **case 3** when there only two biological conditions.

4.2.3.1 PCA (case 3 two conditions)

When samples belong to different biological conditions and different time points, the previous lines of code return from the results of the function **DATAnormalization()**:

- The results of the function `PCA()`
- one 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window (only if `D3.mouvement=FALSE`) where samples are colored with different colors for different biological conditions. Furthermore, lines are drawn between each pair of consecutive points for each sample (if `Mean.Accross.Time=FALSE`, otherwise it will be only between means).

- one 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window (only if `D3.mouvement=FALSE`) for each biological condition, where samples are colored with different colors for different time points. Furthermore, lines are drawn between each pair of consecutive points for each sample (if `Mean.Accross.Time=FALSE`, otherwise it will be only between means).
- the same graphs describe above but without lines.

```
ResPCALeuk500 <- PCAanalysis(SEresNorm=resNORMleuk500,
                             gene.deletion=NULL,
                             sample.deletion=NULL,
                             Supp.del.sample=FALSE,
                             Plot.PCA=FALSE,
                             Mean.Accross.Time=FALSE,
                             Color.Group = NULL,
                             Cex.label=0.9,
                             Cex.point=0.8,epsilon=0.2,
                             Phi=25,Theta=140,
                             D3.mouvement=FALSE,
                             path.result=NULL,
                             Name.folder.pca=NULL)
```

The graphs are

- displayed if `Plot.PCA=TRUE`
- similar to those described in subsection PCA (case 2).
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

By default (if `Color.Group=NULL`), a color will be automatically assigned to each biological condition. The user can change the colors by creating the following data.frame

```
data.col.Leuk<-data.frame(Name=c("NP","P"),
                          Col=c("black","red"))
```

and setting `Color.Group=data.col.Leuk`. The user cannot change the color associated to each time point.

If the user wants to delete, for instance, the genes 'ABCA7' and 'ADAM28' (respectively the second and sixth gene) and/or delete the samples 'CLL_P_r1_t1' and 'CLL_P_r2_t2', he can set

- `gene.deletion=c("ABCA7","ADAM28")` and/or `sample.deletion=c("CLL_P_r1_t1", "CLL_P_r2_t2")`
- `gene.deletion=c(2,6)` and/or `sample.deletion=c(3,13)`.
  The integers in `gene.deletion` and `sample.deletion` represent respectively the row numbers and the column numbers of `RawCounts` where the selected genes and samples are located.

Write `?PCAanalysis` in your console for more information about the function.

4.2.3.2 HCPC (case 3 two conditions)

The lines of code below return from the results of the function **DATAnormalization()**:

- The results of the function `HCPC()`
- A dendrogram
- A graph indicating by color for each sample, its cluster, the biological condition and the time point associated to the sample.
- One 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window (only if `D3.mouvement=FALSE`). These PCA graphs are identical to the outputs of `PCA analysis()` with all samples but samples are colored with different colors for different clusters.

```
ResHCPCLeuk500 <- HCPCanalysis(SEresNorm=resNORMleuk500,
                               gene.deletion=NULL,
                               sample.deletion=NULL,
                               Supp.del.sample=FALSE,
                               Plot.HCPC=FALSE,
                               Phi=25,Theta=140,
                               Cex.point=0.7,
                               epsilon=0.2,
                               Cex.label=0.9,
                               D3.mouvement=FALSE,
                               path.result=NULL,
                               Name.folder.hcpc=NULL)
```

The graphs are

- displayed if `Plot.HCPC=TRUE`
- similar to those described in subsection HCPC (case 2).
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

The optimal number of clusters is automatically computed by the R function `HCPC()`.

Write `?HCPCanalysis` in your console for more information about the function.

4.2.4 Mouse data 2 (case 3 with more than two biological conditions)

In this section we use the mouse subdataset **RawCounts_Weger2021_MOUSEsub500** (see Subsection Mouse data 2) in order to explain **PCAanalysis()** and **HCPCanalysis()** in **case 3** when there are more than two biological conditions.

4.2.4.1 PCA (case 3 with more than two conditions)

When samples belong to different biological conditions and different time points, the previous lines of code return from the results of the function **DATAnormalization()**:

- The results of the function `PCA()`

- one 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window (only if `D3.mouvement=FALSE`) where samples are colored with different colors for different biological conditions. Furthermore, lines are drawn between each pair of consecutive points for each sample (if `Mean.Accross.Time=FALSE`, otherwise it will be only between means).
- one 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window (only if `D3.mouvement=FALSE`) for each biological condition, where samples are colored with different colors for different time points. Furthermore, lines are drawn between each pair of consecutive points for each sample (if `Mean.Accross.Time=FALSE`, otherwise it will be only between means).
- the same graphs describe above but without lines.

```
ResPCAMus2500 <- PCAanalysis(SEresNorm=resNORMmus2500,
                             gene.deletion=NULL,
                             sample.deletion=NULL,
                             Supp.del.sample=FALSE,
                             Plot.PCA=FALSE,
                             Mean.Accross.Time=FALSE,
                             Color.Group=NULL,
                             Cex.label=0.6,
                             Cex.point=0.7, epsilon=0.2,
                             Phi=25,Theta=140,
                             D3.mouvement=FALSE,
                             path.result=NULL,
                             Name.folder.pca=NULL)
```

The graphs are

- displayed if `Plot.PCA=TRUE`
- similar to those described in subsection PCA (case 2).
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

By default (if `Color.Group=NULL`), a color will be automatically applied for each biological condition. You can change the colors by creating the following data.frame

```
data.col.Mus2<-data.frame(Name=c("BmKo","BmWt","CrKo","CrWt"),
                          Col=c("red","blue","orange","darkgreen"))
```

and setting `Color.Group=data.col.Mus2`. The user can not change the color associated to each time point.

If you want to delete, for instance, the genes 'ENSMUSG00000025921' and 'ENSMUSG00000026113' (respectively the second and sixth gene) and/or delete the samples 'BmKo_t2_r1' and 'BmKo_t5_r2', set

- `gene.deletion=c("ENSMUSG00000025921", "ENSMUSG00000026113")` and/or `sample.deletion=c("BmKo_t2_r1", "BmKo_t5_r2")`
- `gene.deletion=c(2,6)` and/or `sample.deletion=c(3,13)`.
  The integers in `gene.deletion` and `sample.deletion` represent respectively the row numbers and the column numbers of `RawCounts` where the selected genes and samples are located.

Write `?PCAanalysis` in your console for more information about the function.

4.2.4.2 HCPC (case 3 with more than two conditions)

The lines of code below return from the results of the function **DATAnormalization()**:

- The results of the function `HCPC()`
- A dendrogram
- A graph indicating by color for each sample, its cluster, the biological condition and the time point associated to the sample.
- One 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window (only if `D3.mouvement=FALSE`). These PCA graphs are identical to the outputs of `PCA analysis()` with all samples but samples are colored with different colors for different clusters.

The user can realize the clustering with HCPC using the function **HCPCanalysis()** as below.

```
ResHCPCMus2500 <- HCPCanalysis(SEresNorm=resNORMmus2500,
                               gene.deletion=NULL,
                               sample.deletion=NULL,
                               Supp.del.sample=FALSE,
                               Plot.HCPC=FALSE,
                               Phi=25,Theta=140,
                               Cex.point=0.6,
                               epsilon=0.2,
                               Cex.label=0.6,
                               D3.mouvement=FALSE,
                               path.result=NULL,
                               Name.folder.hcpc=NULL)
```

The graphs are

- displayed if `Plot.HCPC=TRUE`
- similar to those described in subsection HCPC (case 2).
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

The optimal number of clusters is automatically computed by the R function `HCPC()`.

Write `?HCPCanalysis` in your console for more information about the function.

4.3 Temporal clustering analysis with MFUZZanalysis()

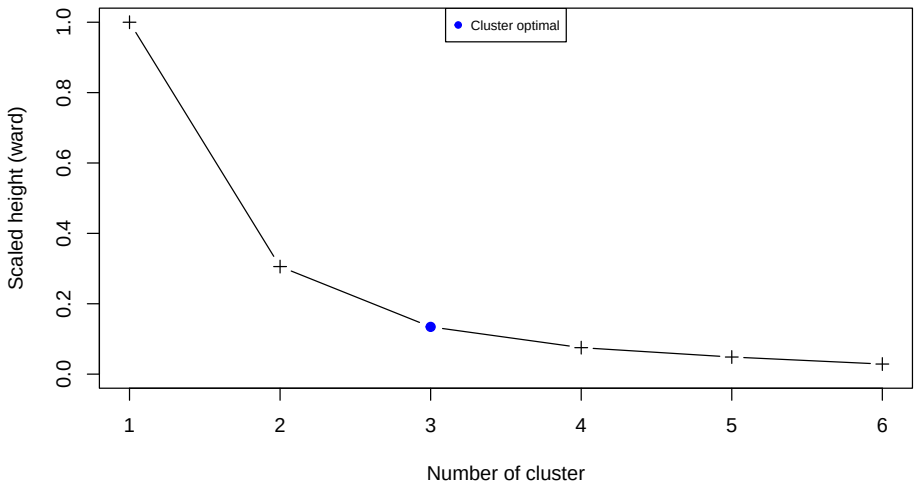
4.3.1 Fission data (case 2)

In this section we use the fission yeast subdataset **RawCounts_Leong2014_FISSIONsub500wt** (see the subsection Fission dataset) in order to explain **MFUZZanalysis()** in **case 2**.

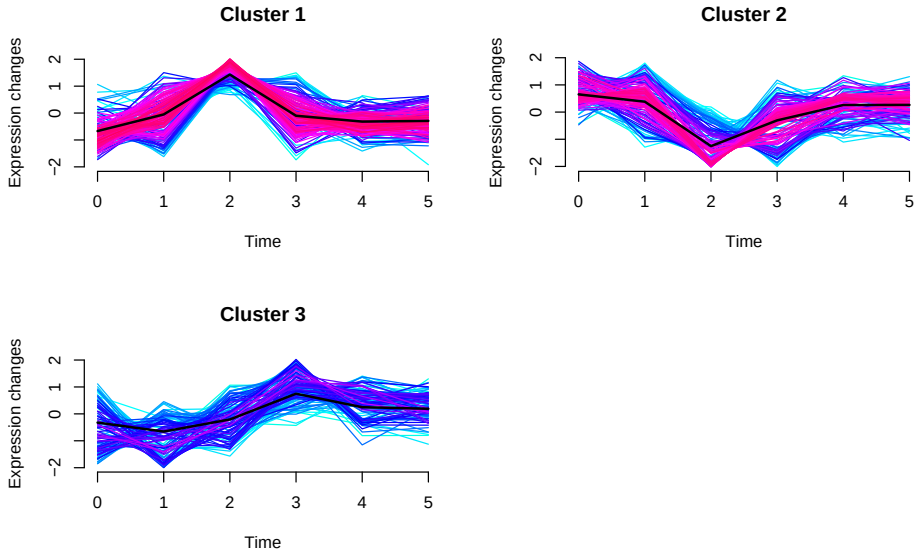
The following function realizes the temporal clustering analysis. It takes as input, a number of clusters (`DataNumberCluster`) that can be chosen automatically if `DataNumberCluster=NULL` and the results of the function **DATAnormalization()**. The lines of code below return

- the summary of the results of the function `mfuzz()`.
- the scaled height plot, computed with the `HCPC()` function, and shows the cluster chosen automatically (if `DataNumberCluster=NULL`). If `Method="hcpc"`, the function plots the scaled within-cluster inertia, but if `Method="kmeans"`, the function plots the scaled within-cluster inertia. As the number of genes can be very high, we recommend to select `Method="hcpc"` which is by default.
- the output graphs from the R package Mfuzz, showing the most common temporal behavior among all genes.

```
ResMfuzzYeast500 <- MFUZZanalysis(SEresNorm=resNORMyeast500,
                                  DataNumberCluster=NULL,
                                  Method="hcpc",
                                  Membership=0.5,
                                  Min.std=0.1,
                                  Plot.Mfuzz=TRUE,
                                  path.result=NULL)
```




```
## 0 genes excluded.
## 31 genes excluded.
```



The graphs are

- displayed if `Plot.Mfuzz=TRUE`
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

Other temporal information are shown in the alluvial graph of the subsection DE analysis with DEanalysisGlobal() that can be compared with the previous graphs.

Write `?MFUZZanalysis` in your console for more information about the function.

4.3.2 Leukemia data (case 3 with only two biological condition)

In this section we use the Chronic lymphocytic leukemia (CLL) subdataset **RawCounts_Schleiss2021_CLLsub500** (see Subsection Leukemia dataset) in order to explain **MFUZZanalysis()** in **case 3** when there only two biological conditions.

The following function realizes the temporal clustering analysis. It takes as input, a number of clusters (`DataNumberCluster`) that can be chosen automatically if `DataNumberCluster=NULL` and the results of the function **DATAnormalization()**. The lines of code below return for each biological condition

- the summary of the results of the function `mfuzz()`
- the scaled height plot, computed with the `HCPC()` function, and shows the cluster chosen automatically (if `DataNumberCluster=NULL`). If `Method="hcpc"`, the function plots the scaled within-cluster inertia, but if `Method="kmeans"`, the function plots the scaled within-cluster inertia. As the number of genes can be very high, we recommend to select `Method="hcpc"` which is by default.
- the output graphs from the R package Mfuzz showing the most common temporal behavior among all genes for each biological condition. The plots below correspond to the biological condition 'P'.

```
ResMfuzzLeuk500 <- MFUZZanalysis(SEresNorm = resNORMleuk500,
                                 DataNumberCluster=NULL,
                                 Method="hcpc",
```

```
                                    Membership=0.7,
                                    Min.std=0.1,
                                    Plot.Mfuzz=FALSE,
                                    path.result=NULL,
                                    Name.folder.mfuzz=NULL)
## 0 genes excluded.
## 6 genes excluded.
## 0 genes excluded.
## 9 genes excluded.
```

The graphs are

- displayed if `Plot.Mfuzz=TRUE`
- similar to those described in subsection Fission data (case 2)
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

Other temporal information are shown in the alluvial graph of the subsection DE analysis with DEanalysisGlobal() that can be compared with the previous graphs.

Write `?MFUZZanalysis` in your console for more information about the function.

4.3.3 Mouse data 2 (case 3 with more than two biological conditions)

In this section we use the mouse subdataset **RawCounts_Weger2021_MOUSEsub500** (see Subsection Mouse dataset 1) in order to explain **MFUZZanalysis()** in **case 3** when there are more than two biological conditions.

The following function realizes the temporal clustering analysis. It takes as input, a number of clusters (`DataNumberCluster`) that can be chosen automatically if `DataNumberCluster=NULL` and the results of the function **DATAnormalization()**. The lines of code below return for each biological condition

- the summary of the results of the function `mfuzz()`
- the scaled height, computed with the `HCPC()` function, and shows the cluster chosen automatically (if `DataNumberCluster=NULL`). If `Method="hcpc"`, the function plots the scaled within-cluster inertia, but if `Method="kmeans"`, the function plots the scaled within-cluster inertia. As the number of genes can be very high, we recommend to select `Method="hcpc"` which is by default.
- the output graph from the R package Mfuzz which represents the most common temporal behavior among all genes for the biological condition 'BmKo.'

```
ResMfuzzLeuk500 <- MFUZZanalysis(SEresNorm=resNORMmus2500,
                                    DataNumberCluster=NULL,
                                    Method="hcpc",
                                    Membership=0.5,
                                    Min.std=0.1,
                                    Plot.Mfuzz=FALSE,
                                    path.result=NULL, Name.folder.mfuzz=NULL)
## 0 genes excluded.
## 31 genes excluded.
```

```
## 0 genes excluded.
## 25 genes excluded.
## 0 genes excluded.
## 38 genes excluded.
## 0 genes excluded.
## 22 genes excluded.
```

The graphs are

- displayed if `Plot.Mfuzz=TRUE`
- similar to those described in subsection Fission data (case 2).
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

Other temporal information are shown in the alluvial graph of the subsection DE analysis with DEanalysisGlobal() that can be compared with the previous graphs.

Write `?MFUZZanalysis` in your console for more information about the function.

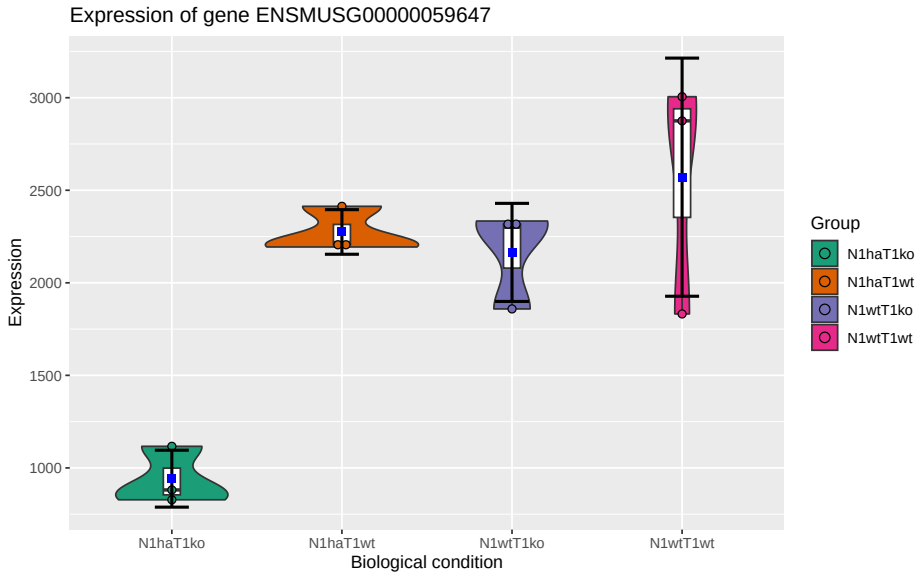
4.4 Plot expression of data with DATAplotExpressionGenes()

4.4.1 Mouse data (case 1)

In this section we use the mouse subdataset **RawCounts_Antoszewski2022_MOUSEsub500** (see the subsection Mouse dataset 1) in order to explain **DATAplotExpressionGenes()** in **case 1**.

The lines of code below allow to plot, from the results of the function **DATAnormalization()**, the expression profile of the 10th gene showing for each biological condition: a box plot, a violin plot, and error bars (standard deviation). Each box plot, violin plot and error bars is associated to the distribution of the expression of all sample belonging to the corresponding biological condition.

```
resEVOmus1500 <- DATAplotExpressionGenes(SEresNorm=resNORMmus1500,
                                         Vector.row.gene=c(10),
                                         Color.Group=NULL,
                                         Plot.Expression=TRUE,
                                         path.result=NULL)
```



If the user wants to select several gene (so several output graphs), for instance the 28th, the 38th, the 39th and the 50th, he needs to set `Vector.row.gene=c(28,38:39,50)`.

The graphs are

- displayed if `Plot.Expression=TRUE`
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

By default (if `Color.Group=NULL`), a color will be automatically applied for each biological condition. The user can change the colors by creating the following data.frame

```
data.col.mus50<-data.frame(Name=c("N1wtT1wt","N1haT1wt","N1haT1ko","N1wtT1ko"),
                           Col=c("black","red","green","blue"))
```

and setting `Color.Group= data.col.mus50`.

Write `?DATAplotExpressionGenes` in your console for more information about the function.

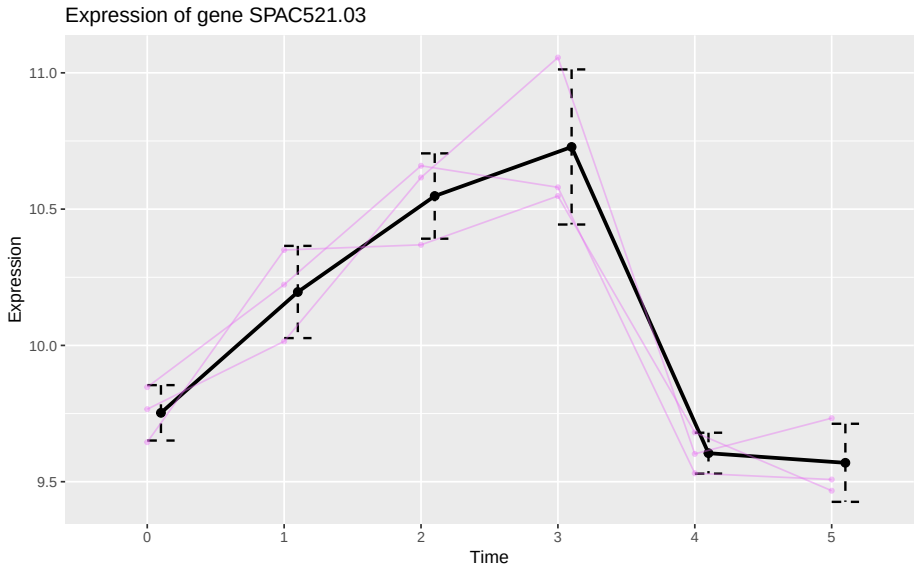
4.4.2 Fission data (case 2)

In this section we use the fission yeast subdataset **RawCounts_Leong2014_FISSIONsub500wt** (see the subsection Fission dataset) in order to explain **DATAplotExpressionGenes()** in **case 2**.

The lines of code below allow to plot, from the results of the function **DATAnormalization()**,

- the evolution of the 17th gene expression of all three replicates across time (purple lines)
- the evolution of the mean and the standard deviation of the 17th gene expression across time (black lines).

```
resEVOyeast500<-DATAplotExpressionGenes(SEresNorm=resNORMyeast500,
                                        Vector.row.gene=c(17),
                                        Plot.Expression=TRUE,
                                        path.result=NULL)
```



The graphs are

- displayed if `Plot.Expression=TRUE`
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

If the user wants to select several genes, for instance the 1st, the 2nd, the 17th and the 19th, he needs to set `Vector.row.gene=c(1,2,17,19)`.

Write `?DATAplotExpressionGenes` in your console for more information about the function.

4.4.3 Leukemia data (case 3 with only two biological conditions)

In this section we use the Chronic lymphocytic leukemia (CLL) subdataset **RawCounts_Schleiss2021_CLLsub500** (see Subsection Leukemia dataset) in order to explain **DATAplotExpressionGenes()** in **case 3** when there only two biological conditions.

The lines of code below allow to plot, from the results of the function **DATAnormalization()**, for each biological condition: the evolution of the 25th gene expression of the three replicates across time and the evolution of the mean and the standard deviation of the 25th gene expression across time. The color of the different lines are different for different biological conditions.

```
resEVOleuk500 <- DATAplotExpressionGenes(SEresNorm=resNORMleuk500,
                                         Vector.row.gene=c(25),
                                         Color.Group = NULL,
                                         Plot.Expression=FALSE,
                                         path.result=NULL,
                                         Name.folder.profile=NULL)
```

The graphs are

- displayed if `Plot.Expression=TRUE`
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

By default (if `Color.Group=NULL`), a color will be automatically assigned to each biological condition. The user can change the colors by creating the following data.frame

```
data.col.Leuk<-data.frame(Name=c("NP","P"),
                          Col=c("black","red"))
```

and setting `Color.Group=data.col.Leuk`.

If the user wants to select several genes, for instance the 97th, the 192th, the 194th and the 494th, he needs to set `Vector.row.gene=c(97,192,194,494)`.

Write `?DATAplotExpressionGenes` in your console for more information about the function.

4.4.4 Mouse 2 data (case 3 with more thant two biological condition)

In this section we use the mouse subdataset **RawCounts_Weger2021_MOUSEsub500** (see Subsection Mouse data 2) in order to explain **DATAplotExpressionGenes()** in **case 3** when there are more than two biological conditions.

The previous lines of code allow to plot, from the results of the function **DATAnormalization()**, for each biological condition: the evolution of the 17th gene expression of the three replicates across time and the evolution of the mean and the standard deviation of the 17th gene expression across time. The color of the different lines are different for different biological conditions.

```
resEVOmus2500 <- DATAplotExpressionGenes(SEresNorm=resNORMmus2500,
                                         Vector.row.gene=c(17),
                                         Color.Group=NULL,
                                         Plot.Expression=FALSE,
                                         path.result=NULL)
```

The graphs are

- displayed if `Plot.Expression=TRUE`
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

By default (if `Color.Group=NULL`), a color will be automatically assigned to each biological condition. The user can change the colors by creating the following data.frame

```
data.col.MusBmCrKoWt<-data.frame(Name=c("BmKo","BmWt","CrKo","CrWt"),
                                 Col=c("black","blue","green","red"))
```

and setting `Color.Group=data.col.MusBmCrKoWt`.

If the user wants to select several genes, for instance the 97th, the 192th, the 194th and the 494th, he needs to set `Vector.row.gene=c(97,192,194,494)`.

Write `?DATAplotExpressionGenes` in your console for more information about the function.

5 Statistical analysis of the transcriptional response for the four dataset (supervised analysis).

5.1 DE analysis with DEanalysisGlobal()

5.1.1 Mouse data (case 1)

In this section we use the mouse subdataset **RawCounts_Antoszewski2022_MOUSEsub500** (see the subsection Mouse dataset 1) in order to explain **DEanalysisGlobal()** in **case 1**.

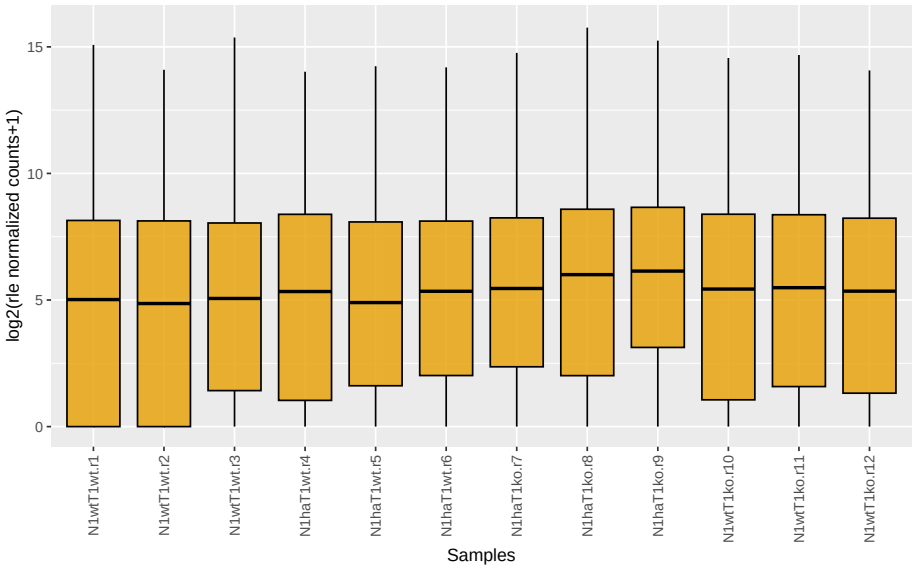
The function below

- returns a data.frame (output `DE.results`). See subection Data.frame summarising all the DE analysis (case 1)
- plots the following graphs
  - an upset graph, realized with the R package UpSetR [Conway et al., 2017], which corresponds to a Venn diagram barplot. If there are only two biological conditions, the graph will not be plotted. See subsection Description of graphs
  - a barplot showing the number of specific genes per biological condition. See subection Description of graphs.

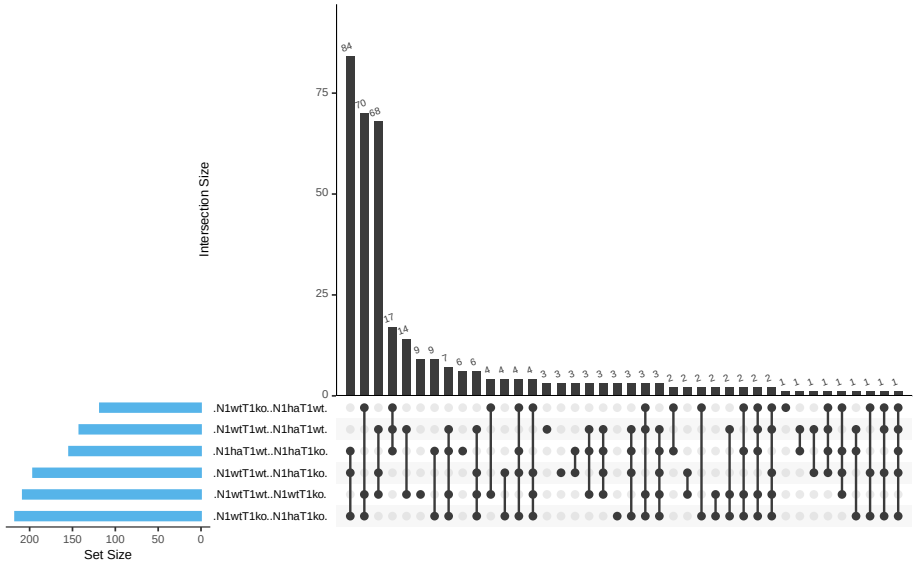
Uncomment lines (`Results_DEanalysis_sub500`) contains the results of `DEanalysisGlobal()` of three dataset which included `RawCounts_Antoszewski2022_MOUSEsub500`. The results of `DEanalysisGlobal()` with `RawCounts_Antoszewski2022_MOUSEsub500` are saved in `Results_DEanalysis_sub500$DE_Antoszewski`

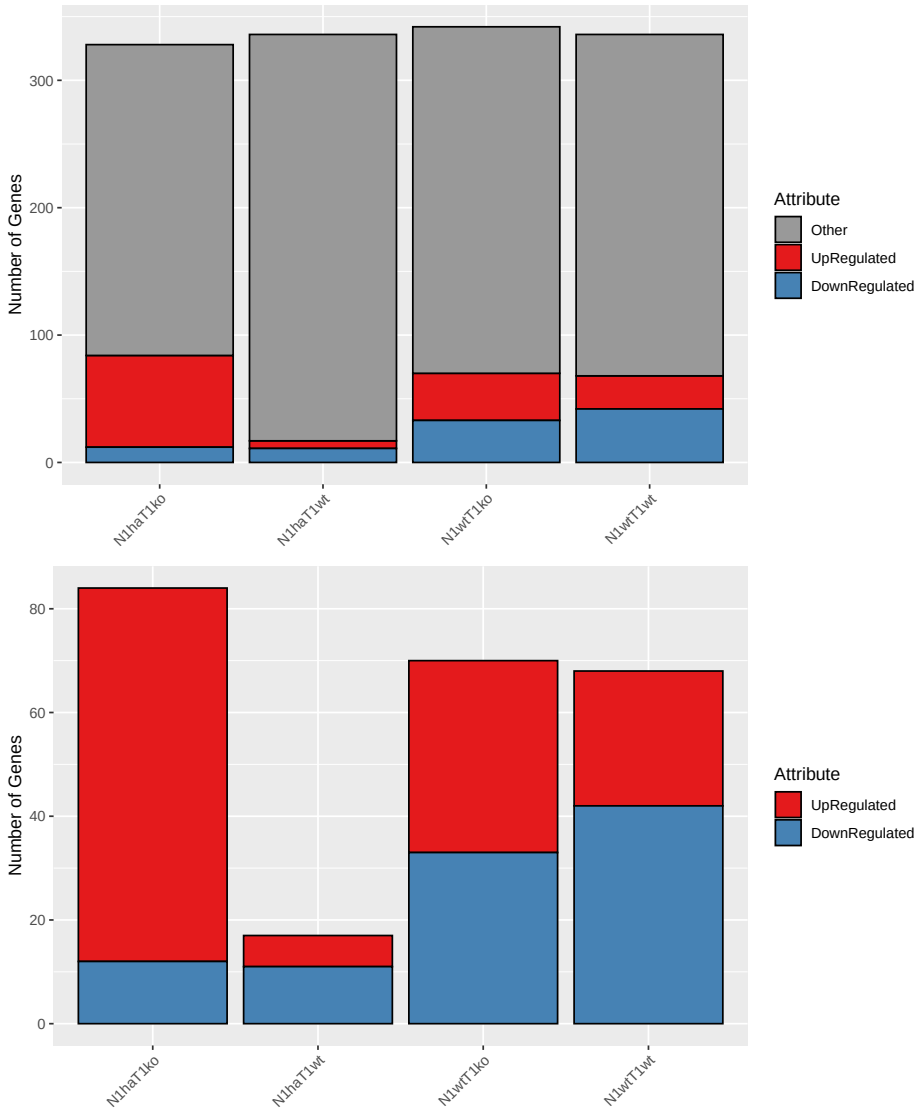
```
ResDEMus500 <- DEanalysisGlobal(SEres=resSEmus1500,
                                pval.min=0.05,
                                pval.vect.t=NULL,
                                log.FC.min=1,
                                LRT.supp.info=FALSE,
                                Plot.DE.graph=TRUE,
                                path.result=NULL,
                                Name.folder.DE=NULL)

## [1] "Preprocessing"
## [1] "Differential expression step with DESeq2::DESeq()"
```




```
## [1] "Case 2 analysis : Biological conditions only"
```






```
# data("Results_DEanalysis_sub500")
# ResDEMus500<-Results_DEanalysis_sub500$DE_Antoszewski2022_MOUSEsub500
```

The graphs are

- displayed if `Plot.DE.graph=TRUE`
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

Write `?DEanalysisGlobal` in your console for more information about the function.

5.1.1.1 Data.frame summarising all the DE analysis (case 1)

The output data.frame `DE.results` can be observed with the following lines of code,

```
Res.DE.results.Mus1 <- SummarizedExperiment::rowData(ResDEMus500)
```

as the glossary of the column names with

```
ResGlossary.Mus1 <- S4Vectors::metadata(ResDEMus500)$DESeq2obj$List.Glossary
```

and then write `Res.DE.results.Mus1` and `ResGlossary.Mus1` in the R console.

The data.frame contains

- gene names
- pvalues, log2 fold change and DE genes between each pairs of biological conditions.
- a binary column (1 and 0) where 1 means the gene is DE between at least one pair of biological conditions.
- $N_{bc} = 4$ binary columns (with N_{bc} the number of biological conditions), which give the genes specific for each biological condition. A '1' in one of these columns means the gene is specific to the biological condition associated to the given column. 0 otherwise. A gene is called specific to a given biological condition BC1, if the gene is DE between BC1 and any other biological conditions, but not DE between any pair of other biological conditions.
- $N_{bc} = 4$ columns filled with -1, 0 and 1, one per biological condition. A '1' in one of these columns means the gene is up-regulated (or over-expressed) for the biological condition associated to the given column. A gene is called up-regulated for a given biological condition BC1 if the gene is specific to the biological condition BC1 and expressions in BC1 are higher than in the other biological conditions. A '-1' in one of these columns means the gene is down-regulated (or under-expressed) for the biological condition associated to the given column. A gene is called down-regulated for a given biological condition BC1 if the gene is specific to the biological condition BC1 and expressions in BC1 are lower than in the other biological conditions. A '0' in one of these columns means the gene is not specific to the biological condition associated to the given column.

5.1.1.2 Description of graphs

The function plots three graphs

The upset graph plots the number of genes corresponding to each possible intersection in a Venn barplot. We say that

- a set of pairs of biological conditions forms an intersection when there is at least one gene which is DE for each of these pairs of biological conditions, but not for the others.
- a gene corresponds to a given intersection if this genes is DE for each pair of biological conditions in the intersection, but not for the others.

For instance, the second column gives the number of genes corresponding to the intersection

$$\Big\{\{\text{N1wtT1wt, N1haT1wt}\},\{\text{N1wtT1wt, N1haT1ko}\},\{\text{N1wtT1wt, N1wtT1ko}\}\Big\}\ .$$

In other words, this is the number of genes DE between

- N1wtT1wt versus N1haT1wt

- N1wtT1wt versus N1haT1ko
- N1wtT1wt versus N1wtT1ko

and not DE between any other pair of biological conditions. Note that this columns actually gives the number of specific genes to the biological condition N1wtT1wt.

The first barplot plots for each biological condition BC1

- The number of up- and down-regulated genes which are specific to the biological condition BC1.
- The number of genes categorized as "Other." A gene belongs to the "Other" category if it is DE between BC1 and at least one other biological condition and not specific.

The second barplot is the same graph without "Other" will be plotted too. If there are only two biological condition, only the second barplot will be plotted.

5.1.2 Fission data (case 2)

In this section we use the fission yeast subdataset **RawCounts_Leong2014_FISSIONsub500wt** (see the subsection Mouse dataset 1) in order to explain **DEanalysisGlobal()** in **case 2**.

The function below

- returns a data.frame (output `DE.results`). See subsection Data.frame summarising all the DE analysis (case 2)
- plots the following graphs
  - an alluvial graph. See subsection Description of graphs
  - a graph showing the number of DE genes as a function of time for each temporal group. By temporal group, we mean the sets of genes which are first DE at the same time. See subsection Description of graphs
  - a barplot showing the number of DE genes (up- or down-regulated) for each time. See subsection Description of graphs.
  - two upset graphs, realized with the R package UpSetR [Conway et al., 2017], showing the number of DE genes belonging to each DE temporal pattern. By temporal pattern, we mean the set of times ti such that the gene is DE between ti and the reference time t0. See subsection Description of graphs.

The lines are commented because they take too much time. Uncomment them in order to use the function `DEanalysisGlobal()`. `Results_DEanalysis_sub500` contains the results of `DEanalysisGlobal()` of three dataset which included `RawCounts_Leong2014_FISSIONsub500wt`. The results of `DEanalysisGlobal()` with `RawCounts_Leong2014_FISSIONsub500wt` are saved in `Results_DEanalysis_sub500$DE_Leong2014_FISSIONsub500wt`.

```
# DEyeast500wt <- DEanalysisGlobal(SEres=resSEfission500,
#                                  pval.min=0.05,
#                                  pval.vect.t=NULL,
#                                  log.FC.min=1,
#                                  LRT.supp.info=FALSE,
#                                  Plot.DE.graph =FALSE,
#                                  path.result=NULL,
#                                  Name.folder.DE=NULL)
data("Results_DEanalysis_sub500")
DEyeast500wt <- Results_DEanalysis_sub500$DE_Leong2014_FISSIONsub500wt
```

The graphs are

- displayed if `Plot.DE.graph=TRUE`
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

Write `?DEanalysisGlobal` in your console for more information about the function.

5.1.2.1 Data.frame summarising all the DE analysis (case 2)

The output data.frame `DE.results` can be observed with the following lines of code,

```
Res.DE.results.Fission <- SummarizedExperiment::rowData(DEyeast500wt)
```

as the glossary of the column names with

```
ResGlossary.Fission <- S4Vectors::metadata(DEyeast500wt)$DESeq2obj$List.Glossary
```

and then write `Res.DE.results.Fission` and `ResGlossary.Fission` in the R console.

The data.frame contains

- gene names
- pvalues, log2 fold change and DE genes between each time ti versus the reference time t0.
- a binary column (1 and 0) where 1 means the gene is DE at at least between one time ti versus the reference time t0.
- a column where each element is succession of 0 and 1. The positions of '1' indicate the set of times ti such that the gene is DE between ti and the reference time t0. So each element of the column is what we called previously, a temporal pattern.

5.1.2.2 Description of graphs

The function returns the following plots

1. An alluvial graph. The x-axis of the alluvial graph is labeled with all times except t0. For each vertical barplot, there are two strata: 1 and 0 whose sizes indicate respectively the number of DE genes and of non DE genes, between the time corresponding to the barplot and the reference time t0.
   The alluvial graph is composed of curves each corresponding to a single gene, which are gathered in alluvia. An alluvium is composed of all genes having the same curve: for example, an alluvium going from the stratum 0 at time t1 (for instance) to the stratum 1 at time t2 corresponds to the set of genes which are not differentially expressed (DE) at t1 and are DE at time t2. Each alluvium connects pairs of consecutive barplots and its thickness gives the number of genes in the set. The color of each alluvium indicates the temporal group, defined as the set of genes which are all first DE at the same time with respect to the reference time t0.
2. A graph giving the time evolution of the number of DE genes within each temporal group. The x-axis labels indicate all times except t0.
3. A barplot showing the number of DE genes per time. The x-axis labels indicate all times except t0. For each DE gene, we compute the sign of the log2 fold change between time ti and time t0. If the sign is positive (resp. negative), the gene is categorized as up-regulated (resp. down-regulated). In the graph, the up-regulated (resp. down-regulated) genes are indicated in red (resp. in blue).
4. An upset graph, realized with the R package UpSetR [Conway et al., 2017], plotting the number of genes in each DE temporal pattern in a Venn barplot. By DE temporal pattern, we mean a subset of times in t1,...,tn. We say that a gene belongs to a DE temporal pattern if the gene is DE versus t0 only at the times in this DE temporal patterns. For each gene in a given DE temporal pattern, we compute the number of DE

times where it is up-regulated and we use a color code in the Venn barplot to indicate the number of genes in a temporal pattern that are up-regulated a given number of times.

5. The same upset graph is also given without colors.

5.1.3 Leukemia data (case 3 with only two biological conditions)

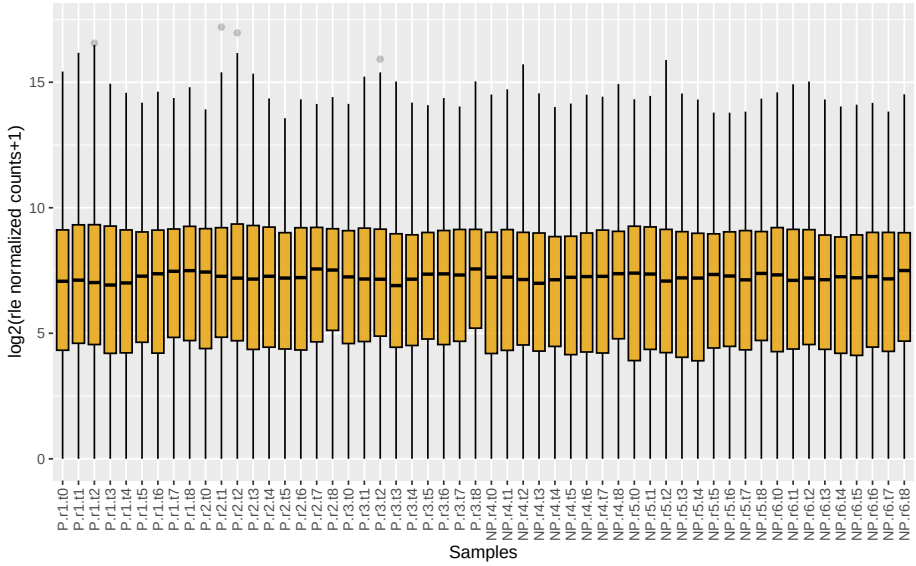
In this section we use the Chronic lymphocytic leukemia (CLL) subdataset **RawCounts_Schleiss2021_CLLsub500** (see Subsection Leukemia dataset) in order to explain **DEanalysisGlobal()** in **case 3** when there only two biological conditions.

The lines of code below

- returns a data.frame (output `DE.results`). See subsection Data.frame summarising all the DE analysis (case 3 with only two biological conditions)
- plots the following graphs
  - *Results from the temporal statistical analysis* (**case 2** for each biological condition). See subsection Graphs from the results of the temporal statistical analysis
  - *Results from the statistical analysis by biological condition* (**case 1** for each fixed time). See subsection Graphs from the results of the biological condition analysis.
  - *Results from the combination of temporal and biological statistical analysis*. See subsection Graphs from the results of the combination of temporal and biological statistical analysis

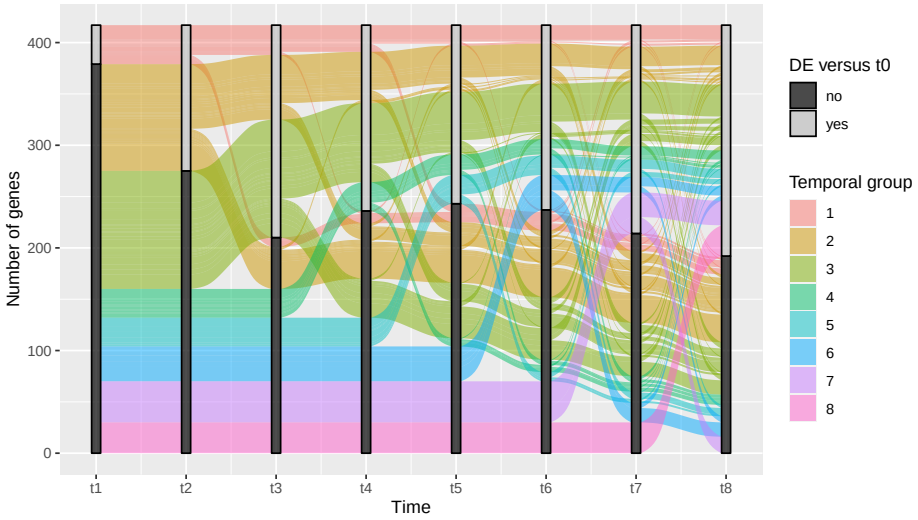
```
ResDELeuk500 <- DEanalysisGlobal(SEres=resSEleuk500,
                                 pval.min=0.05,
                                 pval.vect.t=NULL,
                                 log.FC.min=1,
                                 LRT.supp.info=FALSE,
                                 Plot.DE.graph =TRUE,
                                 path.result=NULL,
                                 Name.folder.DE=NULL)
```

```
## [1] "Preprocessing"
## [1] "Differential expression step with DESeq2::DESeq()"
```

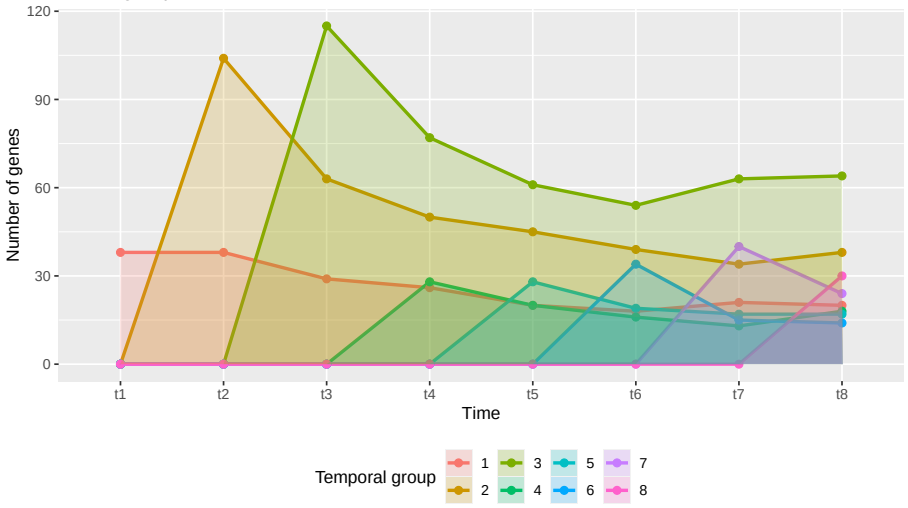


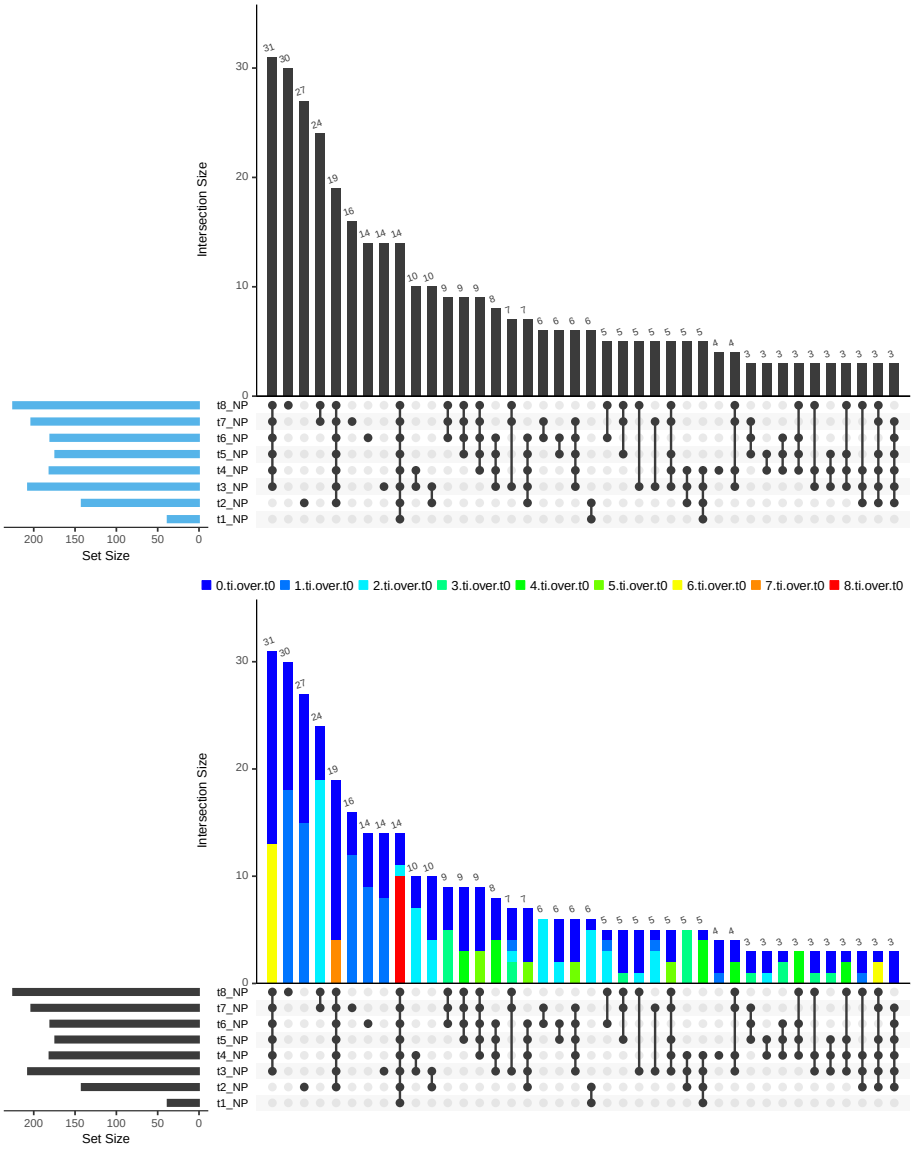

```
## [1] "Case 3 analysis : Biological conditions and Times."
## [1] "DE time analysis for each biological condition."
```

Alluvial graph of genes which are DE at at least one time,
for the group NP

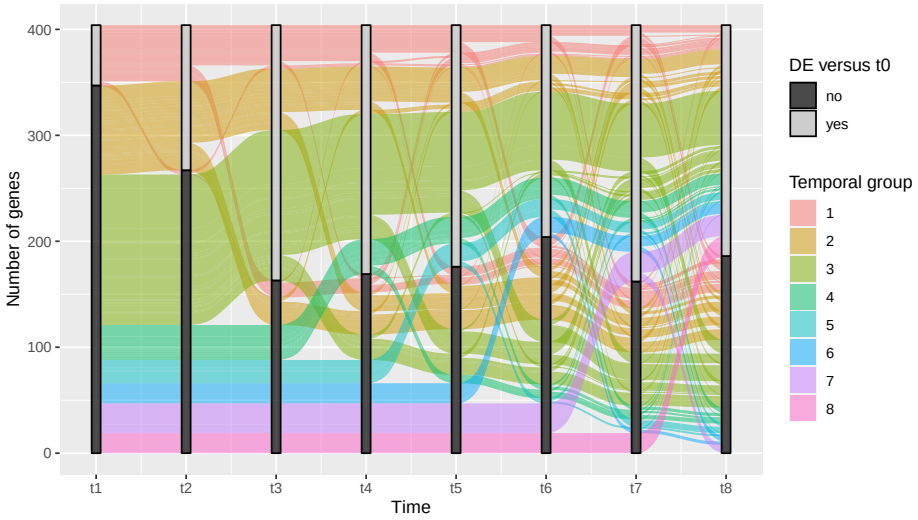


Time evolution of the number of genes which are DE at at least one time within each temporal
for the group NP

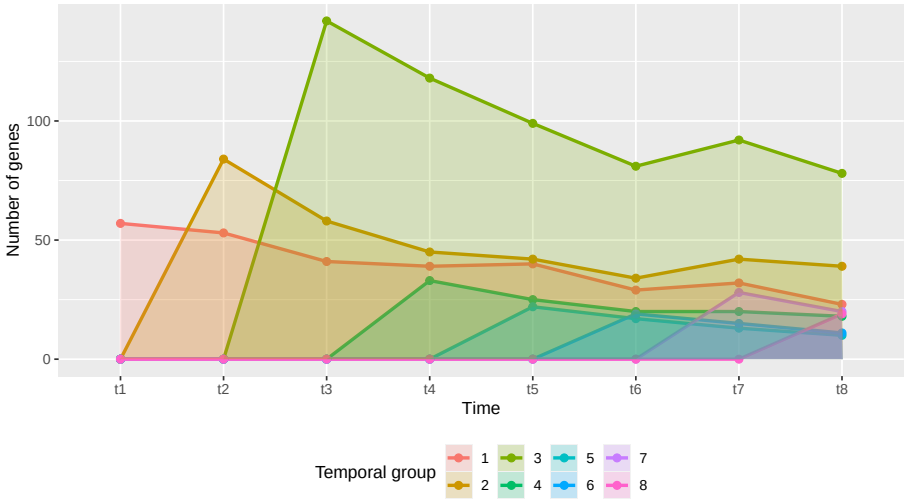


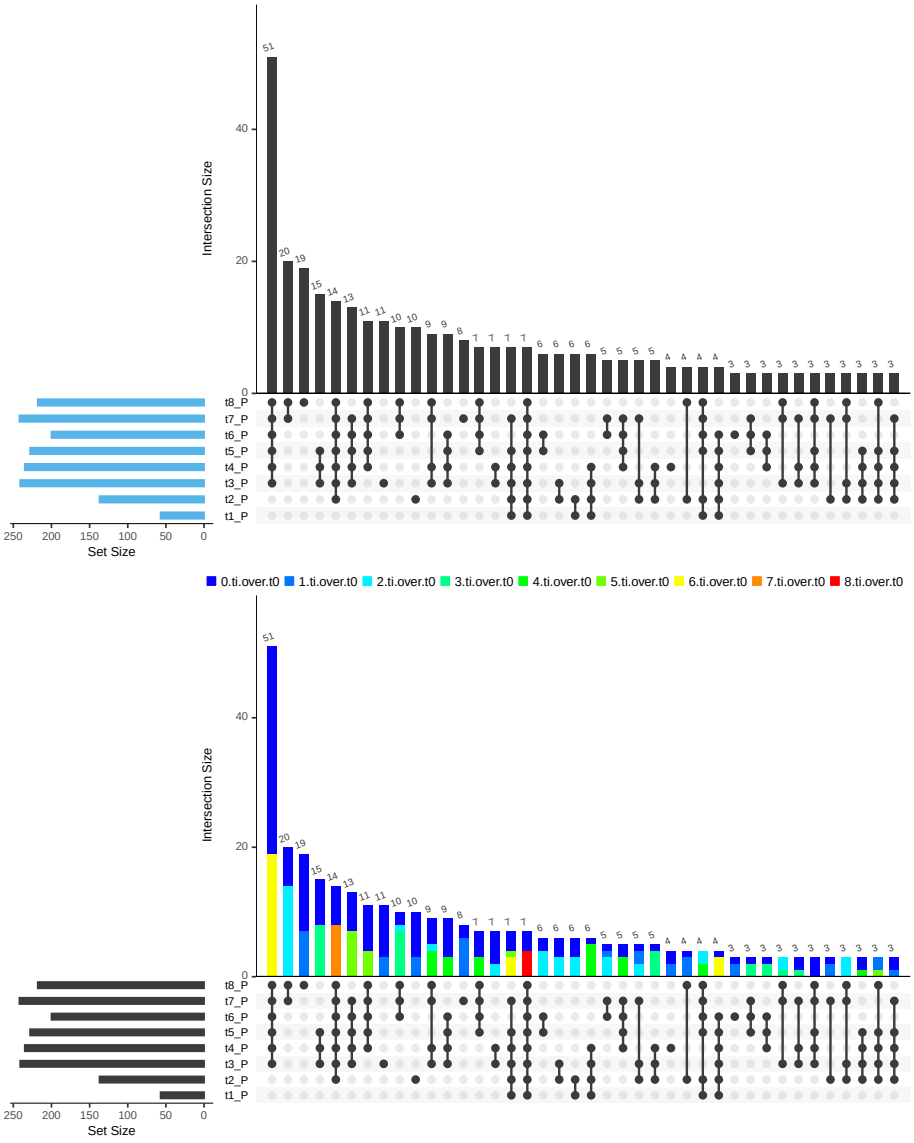

Intersection Size
Set Size
t8_NP
t7_NP
t6_NP
t5_NP
t4_NP
t3_NP
t2_NP
t1_NP
0.ti.over.t0
1.ti.over.t0
2.ti.over.t0
3.ti.over.t0
4.ti.over.t0
5.ti.over.t0
6.ti.over.t0
7.ti.over.t0
8.ti.over.t0

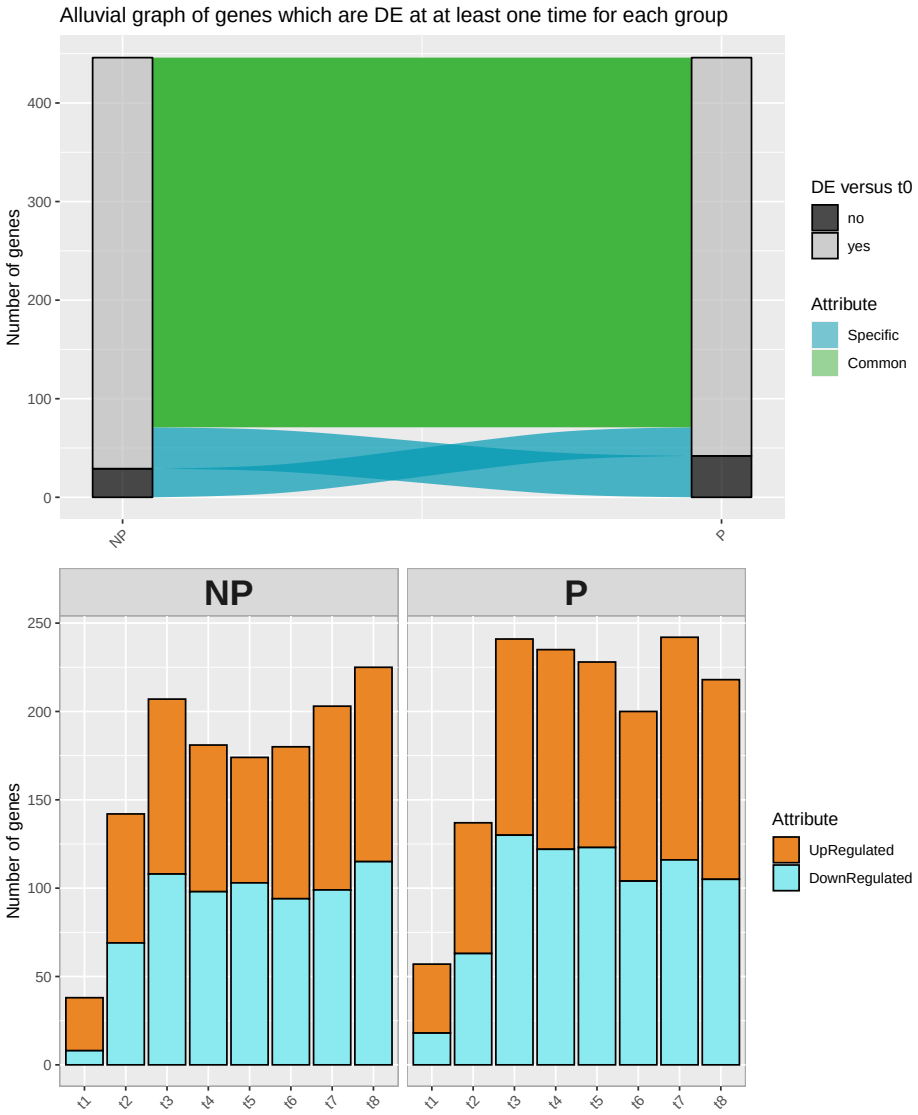
Alluvial graph of genes which are DE at at least one time,
for the group P



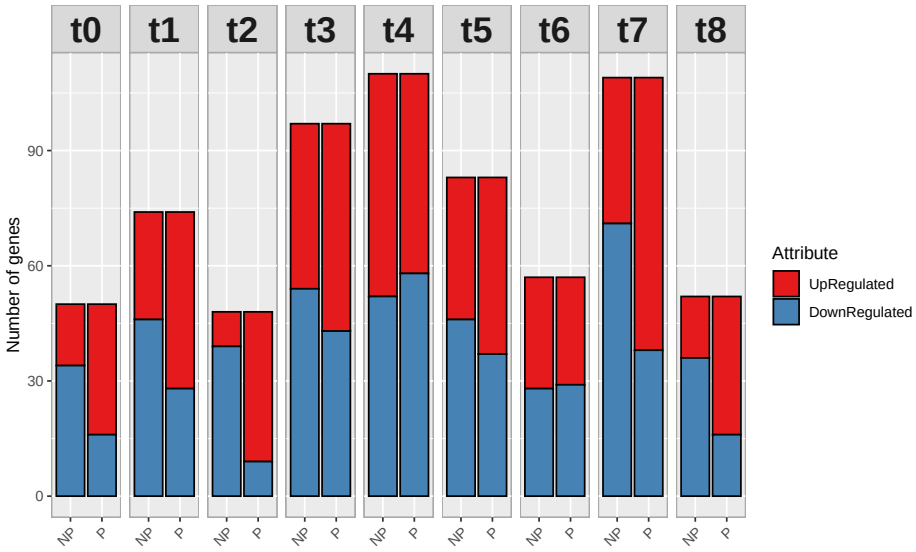
Time evolution of the number of genes which are DE at at least one time within each temporal
for the group P



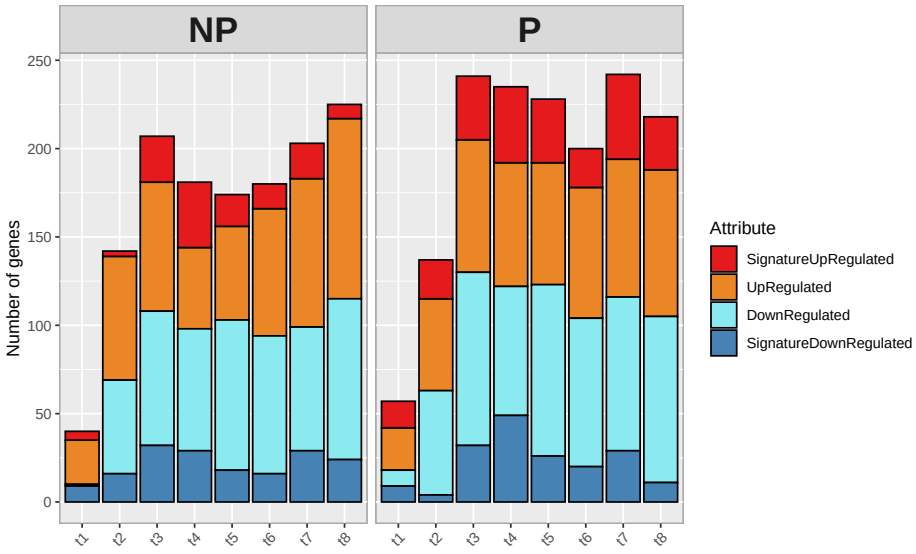

Intersection Size
Set Size
t8_P
t7_P
t6_P
t5_P
t4_P
t3_P
t2_P
t1_P
0.ti.over.t0
1.ti.over.t0
2.ti.over.t0
3.ti.over.t0
4.ti.over.t0
5.ti.over.t0
6.ti.over.t0
7.ti.over.t0
8.ti.over.t0

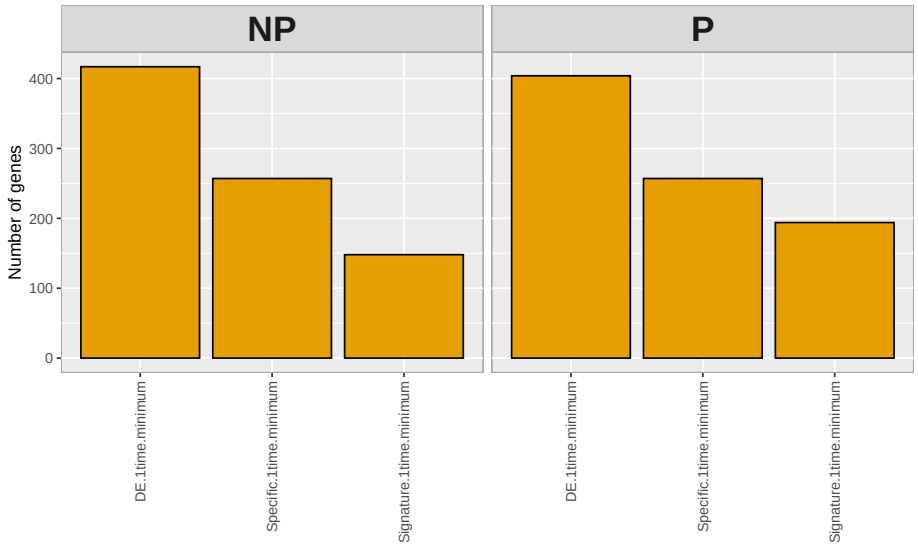



```
## [1] "DE group analysis for each time measurement."
```




```
## [1] "Combined time and group results."
```






```
# data("Results_DEanalysis_sub500")
# ResDELeuk500<-Results_DEanalysis_sub500$DE_Schleiss2021_CLLsub500
```

`Results_DEanalysis_sub500` contains the results of `DEanalysisGlobal()` of three dataset which included `RawCounts_Schleiss2021_CLLsub500`. The results of `DEanalysisGlobal()` with `RawCounts_Schleiss2021_CLLsub500` are saved in `Results_DEanalysis_sub500$DE_Schleiss2021_CLLsub500`.

The graphs are

- displayed if `Plot.DE.graph=TRUE`
- saved in a folder if the user selects a folder path in `path.result`. If `path.result=NULL` the results will not be saved in a folder.

Write `?DEanalysisGlobal` in your console for more information about the function.

5.1.3.1 Data.frame summarising all the DE analysis (case 3 with only two biological conditions)

The output data.frame `DE.results` can be observed with the following lines of code,

```
Res.DE.results.Leuk <- SummarizedExperiment::rowData(ResDELeuk500)
```

as the glossary of the column names with

```
ResGlossary.Leuk <- S4Vectors::metadata(ResDELeuk500)$DESeq2obj$List.Glossary
```

and then write `Res.DE.results.Leuk` and `ResGlossary.Leuk` in the R console.

The data.frame contains

- gene names
- *Results from the temporal statistical analysis* (**case 2** for each biological condition)
  - pvalues, log2 fold change and DE genes between each time ti versus the reference time t0, for each biological condition.

  - $N_{bc} = 2$ binary columns (1 and 0), one per biological condition (with N_{bc} the number of biological conditions). A '1' in one of these two columns means the gene is DE at at least between one time ti versus the reference time t0, for the biological condition associated to the given column.
  - $N_{bc} = 2$ columns where each element is succession of 0 and 1, one per biological condition. The positions of '1,' in one of these two columns, indicate the set of times ti such that the gene is DE between ti and the reference time t0, for the biological condition associated to the given column. So each element of the column is what we called previously, a temporal pattern.
- *Results from the statistical analysis by biological condition* (**case 1** for each fixed time)
  - pvalues, log2 fold change and DE genes between each pairs of biological conditions, for each fixed time.
  - $T = 9$ binary columns (1 and 0), one per time (with T the number of time measurements). A '1,' in one of these columns, means the gene is DE between at least one pair of biological conditions, for the fixed time associated to the given column.
  - $N_{bc} \times T = 2 \times 9 = 18$ binary columns, which give the genes specific for each biological condition at each time ti. A '1' in one of these columns means the gene is specific to the biological condition at a fixed time associated to the given column. '0' otherwise. A gene is called specific to a given biological condition BC1 at a time ti, if the gene is DE between BC1 and any other biological conditions at time ti, but not DE between any pair of other biological conditions at time ti.
  - $N_{bc} \times T = 2 \times 9 = 18$ columns filled with -1, 0 and 1. A '1' in one of these columns means the gene is up-regulated (or over-expressed) for the biological condition at a fixed time associated to the given column. A gene is called up-regulated for a given biological condition BC1 at time ti if the gene is specific to the biological condition BC1 at time ti and expressions in BC1 at time ti are higher than in the other biological conditions at time ti. A '-1' in one of these columns means the gene is down-regulated (or under-expressed) for the biological condition at a fixed time associated to the given column. A gene is called down-regulated for a given biological condition at a time ti BC1 if the gene is specific to the biological condition BC1 at time ti and expressions in BC1 at time ti are lower than in the other biological conditions at time ti. A '0' in one of these columns means the gene is not specific to the biological condition at a fixed time associated to the given column.
  - $N_{bc} = 2$ binary columns (1 and 0). A '1' in one of these columns means the gene is specific at at least one time ti, for the biological condition associated to the given column. '0' otherwise.
- *Results from the combination of temporal and biological statistical analysis*
  - $N_{bc} \times T = 2 \times 9 = 18$ binary columns, which give the signatures genes for each biological condition at each time ti. A '1' in one of these columns means the gene is signature gene to the biological condition at a fixed time associated to the given column. '0' otherwise. A gene is called signature of a biological condition BC1 at a given time ti, if the gene is specific to the biological condition BC1 at time ti and DE between ti versus the reference time t0 for the biological condition BC1.

- $N_{bc} = 2$ binary columns (1 and 0). A '1' in one of these columns means the gene is signature at at least one time ti, for the biological condition associated to the given column. '0' otherwise.

5.1.3.2 Graphs from the results of the temporal statistical analysis

The function plots

- $N_{bc} = 2$ alluvial graphs, one per biological condition (with N_{bc} the number of biological conditions). The x-axis of the graph is labeled with all times except t0. For each vertical barplot, there are two strata: 1 and 0 whose sizes indicate respectively the number of DE genes and of non DE genes, between the time corresponding to the barplot and the reference time t0. The alluvial graph is composed of curves each corresponding to a single gene, which are gathered in alluvia. An alluvium is composed of all genes having the same curve: for example, an alluvium going from the stratum 0 at time t1 (for instance) to the stratum 1 at time t2 corresponds to the set of genes which are not differentially expressed (DE) at t1 and are DE at time t2. Each alluvium connects pairs of consecutive barplots and its thickness gives the number of genes in the set. The color of each alluvium indicates the temporal group, defined as the set of genes which are all first DE at the same time with respect to the reference time t0.
- $N_{bc} = 2$ graphs showing the number of DE genes as a function of time for each temporal group, one per biological condition. By temporal group, we mean the sets of genes which are first DE at the same time. The graphs plot the time evolution of the number of DE genes within each temporal group, one per biological condition. The x-axis labels indicate all times except t0.
- a barplot showing the number of DE genes up-regulated and down-regulated per time, for each biological condition. The graph shows the number of DE genes per time, for each biological condition. The x-axis labels indicate all times except t0. For each DE gene, we compute the sign of the log2 fold change between time ti and time t0. If the sign is positive (resp. negative), the gene is categorized as up-regulated (resp. down-regulated). In the graph, the up-regulated (resp. down-regulated) genes are indicated in red (resp. in blue).
- $2 \times N_{bc} = 4$ upset graphs, realized with the R package UpSetR [Conway et al., 2017], showing the number of DE genes belonging to each DE temporal pattern, for each biological condition. By temporal pattern, we mean the set of times ti such that the gene is DE between ti and the reference time t0. The graphs, realized with the R package UpSetR [Conway et al., 2017], plot the number of genes in each DE temporal pattern in a Venn barplot, one per biological condition. By DE temporal pattern, we mean a subset of times in t1,...,tn. We say that a gene belongs to a DE temporal pattern if the gene is DE versus t0 only at the times in this DE temporal patterns. For each gene in a given DE temporal pattern, we compute the number of DE times where it is up-regulated and we use a color code in the Venn barplot to indicate the number of genes in a temporal pattern that are up-regulated a given number of times. The same graph is also given without colors.
- an alluvial graph for DE genes which are DE at at least one time for each group. The graph shows common and no common genes between which are DE at at least one time for the two biological conditions.

5.1.3.3 Graphs from the results of the biological condition analysis

The function plots

- a barplot showing the number of specific genes per biological condition, for each time. A gene is called specific to a given biological condition BC1 at a time ti, if the gene is DE between BC1 and any other biological conditions at time ti, but not DE between any pair of other biological conditions at time ti.
- only if there are more than two biological conditions (which is not the case here), $\binom{N_{bc}}{2} = \binom{4}{2} = 6$ upset graph, realized with the R package UpSetR [Conway et al., 2017], which corresponds to Venn diagram barplots for each time ti. Each graph the number of genes corresponding to each possible intersection in a Venn barplot at a given time. We say that
  - a set of pairs of biological conditions forms an intersection at a given time ti when there is at least one gene which is DE for each of these pairs of biological conditions at time ti, but not for the others at time ti.
  - a gene corresponds to a given intersection if this genes is DE for each pair of biological conditions in the intersection at time ti, but not for the others at time ti.
- only if there are more than two biological conditions (which is not the case here), an alluvial graph for DE genes which are specific at at least one time for each group. The plots allows to detect genes that are specific only for one biological condition.

5.1.3.4 Graphs from the results of the combination of temporal and biological statistical analysis

The function plots

- a barplot showing the number of signature genes and DE genes (but not signature) per time, for each biological condition. A gene is called signature of a biological condition BC1 at a given time ti, if the gene is specific to the biological condition BC1 at time ti and DE bewteen ti versus the reference time t0 for the biological condition BC1.
- a barplot showing the number of genes which are DE at at least one time, specific at at least one time and signature at at least one time, for each biological condition. The barplot summarizes well the combination of temporal and biological analysis because the figure gives for each biological condition the number of genes which are DE at at least one time, the number of specific genes at at least one time and the number of signature at at least one time.
- only if there are more than two biological conditions (which is not the case here), an alluvial graph for DE genes which are signature at at least one time for each group. The plots allows to detect genes that are signature only for one biological condition.

5.1.4 Mouse data 2 (case 3 with more than two biological conditions)

In this section we use the mouse subdataset **RawCounts_Weger2021_MOUSEsub500** (see Subsection 1.6.4) in order to explain **DEanalysisGlobal()** in **case 3** when there are more than two biological conditions.

For the speed of the algorithm, we will take only three biological conditions and three times

```
Sub3bc3T <- RawCounts_Weger2021_MOUSEsub500[, seq_len(73)]
SelectTime <- grep("_t0_", colnames(Sub3bc3T))
SelectTime <- c(SelectTime, grep("_t1_", colnames(Sub3bc3T)))
SelectTime <- c(SelectTime, grep("_t2_", colnames(Sub3bc3T)))
Sub3bc3T <- Sub3bc3T[, c(1, SelectTime)]

resSEmus25003b3t <- DATAprepSE(RawCounts=Sub3bc3T,
                               Column.gene=1,
                               Group.position=1,
                               Time.position=2,
                               Individual.position=3)
```

The lines of code below

- returns a data.frame (output `DE.results`). See subsection Data.frame summarising all the DE analysis (case 3 with only two biological conditions)
- plots the following graphs (similar to those described in section Leukemia data (case 3 with only two biological conditions))
  - *Results from the temporal statistical analysis* (**case 2** for each biological condition). See subsection Graphs from the results of the temporal statistical analysis
  - *Results from the statistical analysis by biological condition* (**case 1** for each fixed time). See subsection Graphs from the results of the biological condition analysis.
  - *Results from the combination of temporal and biological statistical analysis*. See subsection Graphs from the results of the combination of temporal and biological statistical analysis

```
ResDEMusBmCrKoWt500<-DEanalysisGlobal(SEres=resSEmus25003b3t,
                                      pval.min=0.05,
                                      pval.vect.t=NULL,
                                      log.FC.min=1,
                                      LRT.supp.info=FALSE,
                                      Plot.DE.graph=FALSE,
                                      path.result=NULL, Name.folder.DE=NULL)

## [1] "Preprocessing"
## [1] "Differential expression step with DESeq2::DESeq()"
## [1] "Case 3 analysis : Biological conditions and Times."
## [1] "DE time analysis for each biological condition."
## [1] "DE group analysis for each time measurement."
## [1] "Combined time and group results."
```

As we are in the similar case described in section Leukemia data (case 3 with only two biological conditions), the results will not be described here.

5.2 Volcano plots, ratio intensity (MA) plots and Heatmaps with DEplotVolcanoMA() and DEplotHeatmaps()

5.2.1 Mouse data (case 1)

In this section we use the mouse subdataset **RawCounts_Antoszewski2022_MOUSEsub500** (see the subsection Mouse dataset 1) in order to explain **DEplotVolcanoMA()** and **DEplotHeatmaps()** in **case 1**.

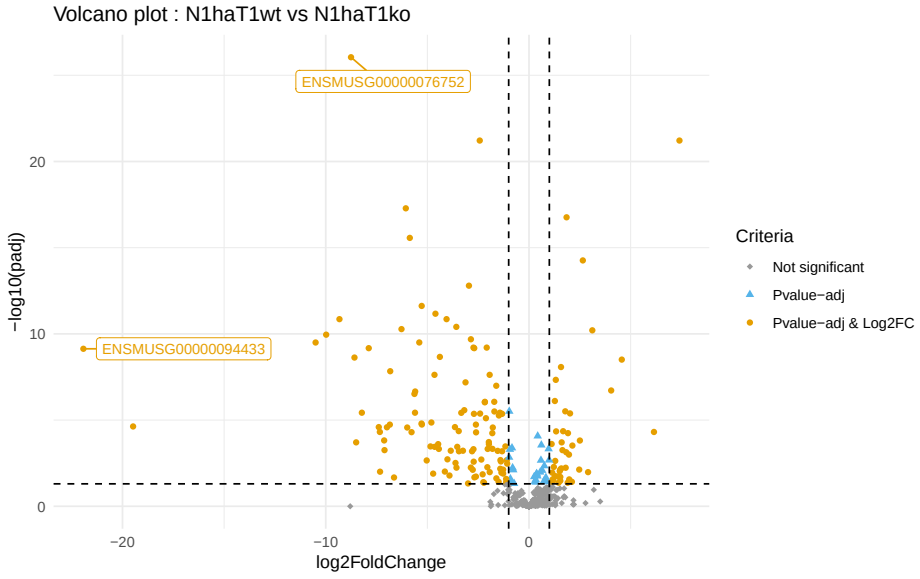
5.2.1.1 Volcano and MA plots (case 1)

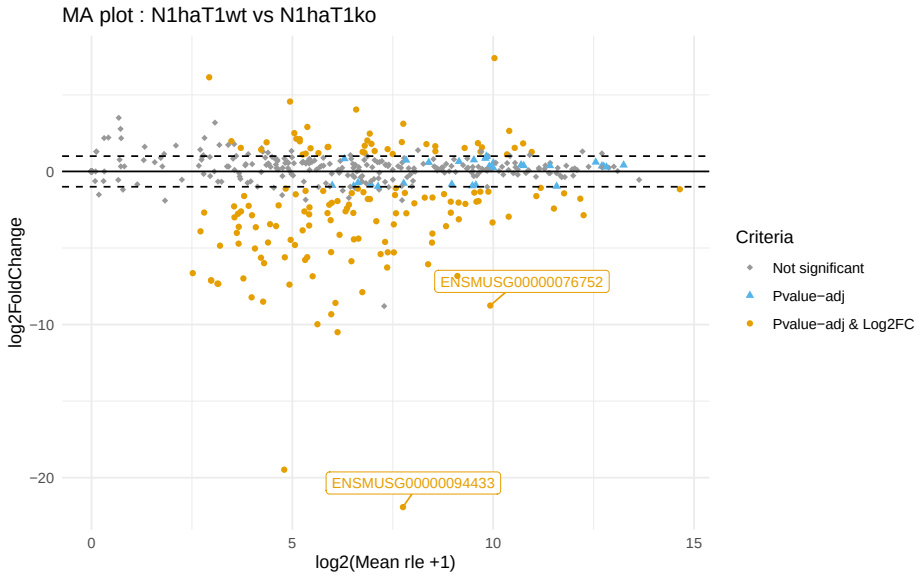
The following lines of code allow to plot

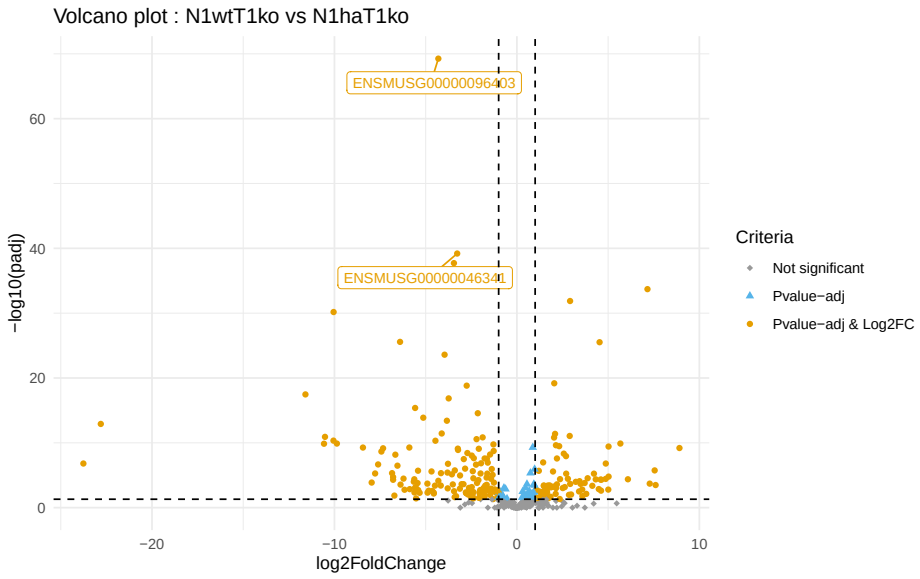
- $\binom{N_{bc}}{2} = \frac{N_{bc}(N_{bc}-1)}{2} = 6$ volcano plots (with $N_{bc} = 4$ the number of biological conditions).
- $\frac{N_{bc}(N_{bc}-1)}{2} = 6$ MA plots.

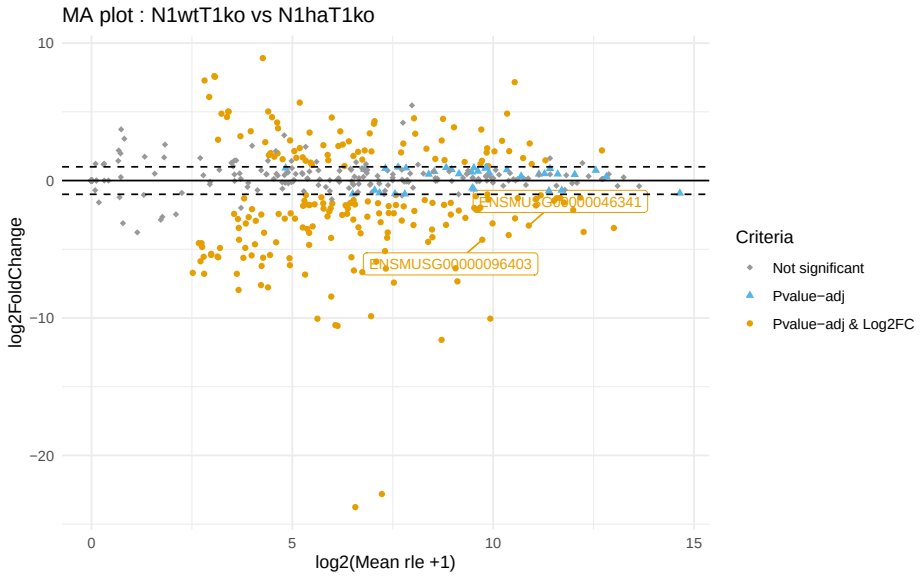
allowing to separate non DE genes, DE genes below a threshold of log2 fold change and DE genes above a threshold of log2 fold change.

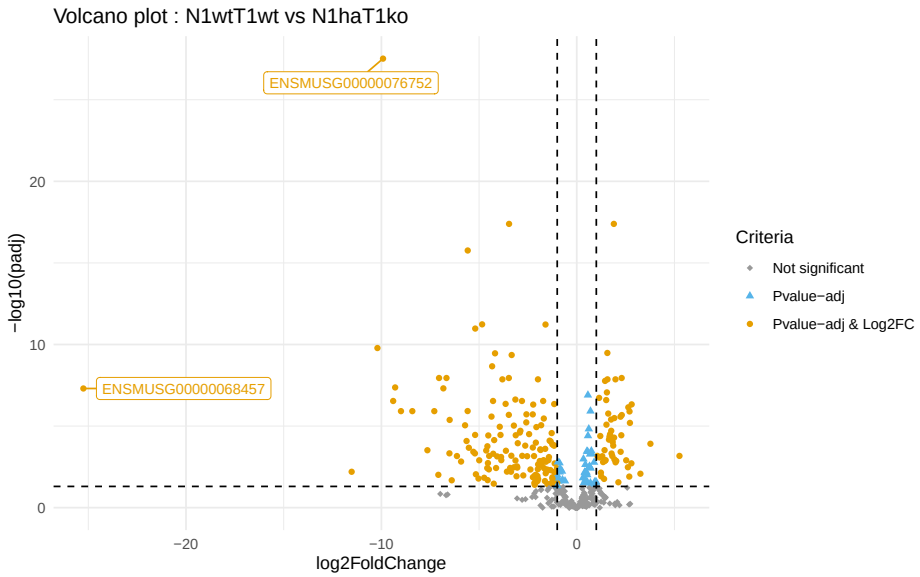
```
resVolcanoMAMouse <- DEplotVolcanoMA(SEresDE=ResDEMus500,
                                     NbGene.plotted=2,
                                     SizeLabel=3,
                                     Display.plots=TRUE,
                                     Save.plots=FALSE)
```

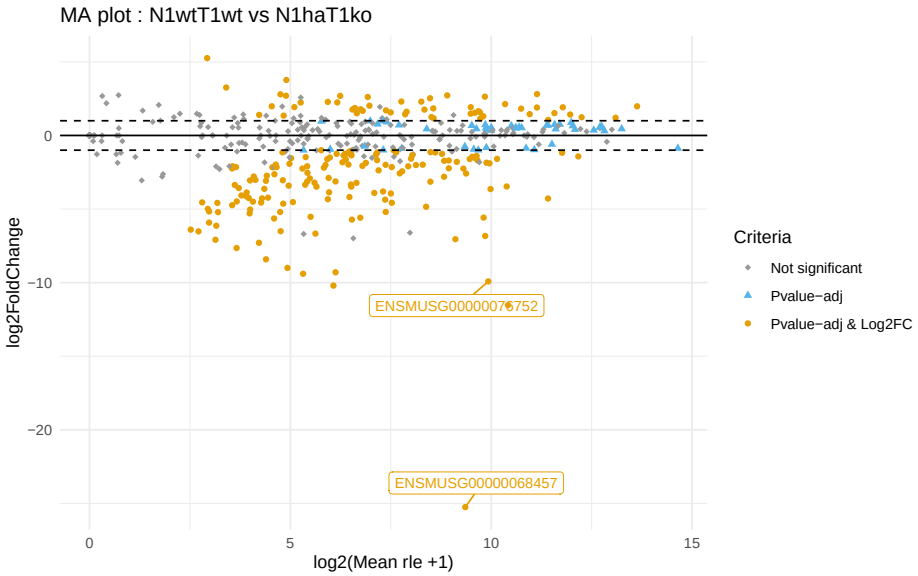


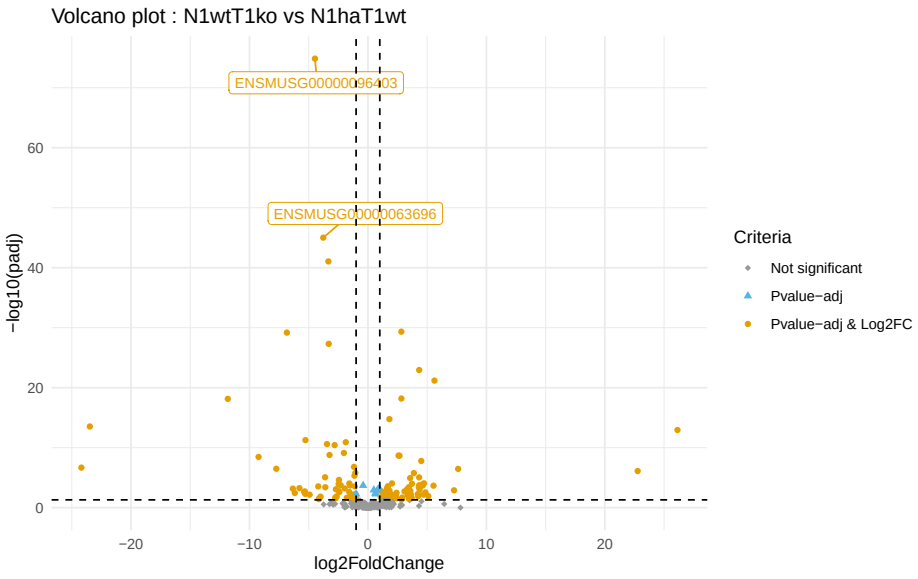

MA plot : N1haT1wt vs N1haT1ko
log2FoldChange
log2(Mean rle +1)
ENSMUSG00000076752
ENSMUSG00000094433
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC

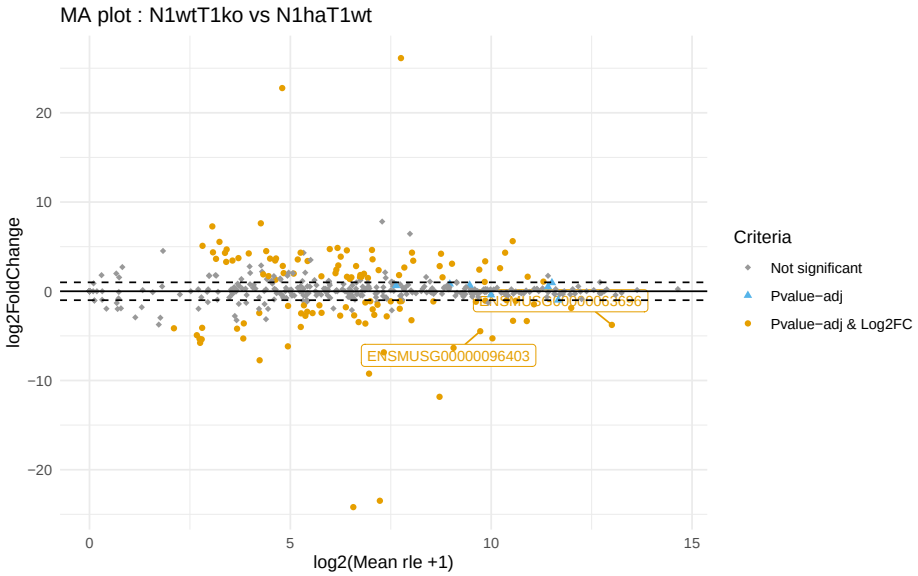

Volcano plot : N1wtT1ko vs N1haT1ko
−log10(padj)
log2FoldChange
ENSMUSG00000096403
ENSMUSG00000046341
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC

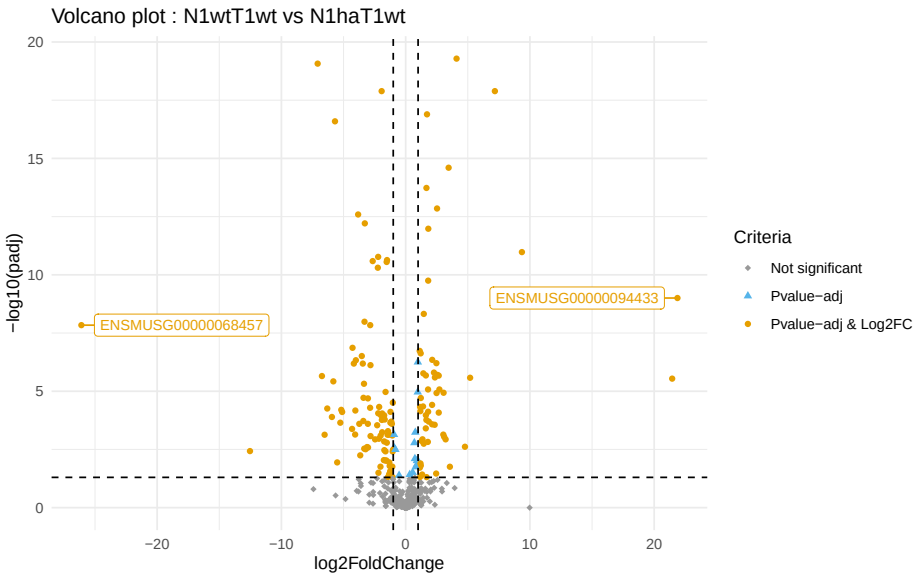

MA plot : N1wtT1ko vs N1haT1ko
log2FoldChange
log2(Mean rle +1)
ENSMUSG00000046341
ENSMUSG00000096403
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC

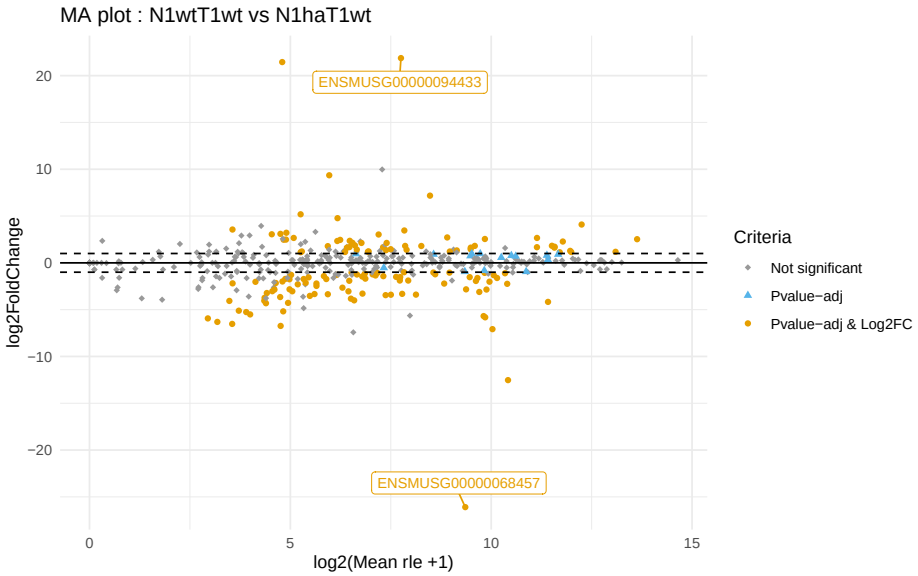

Volcano plot : N1wtT1wt vs N1haT1ko
−log10(padj)
log2FoldChange
ENSMUSG00000076752
ENSMUSG00000068457
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC

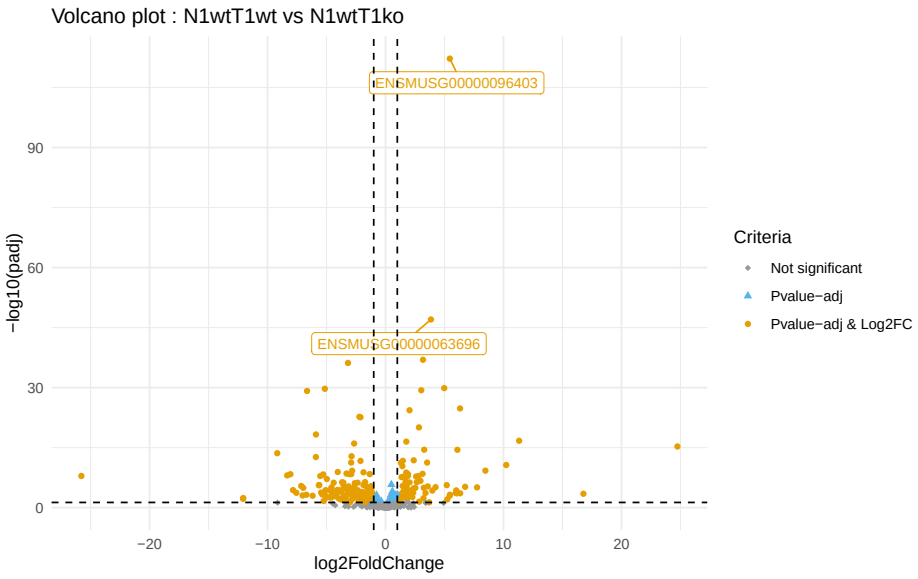

MA plot : N1wtT1wt vs N1haT1ko
log2FoldChange
log2(Mean rle +1)
ENSMUSG00000076752
ENSMUSG00000068457
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC

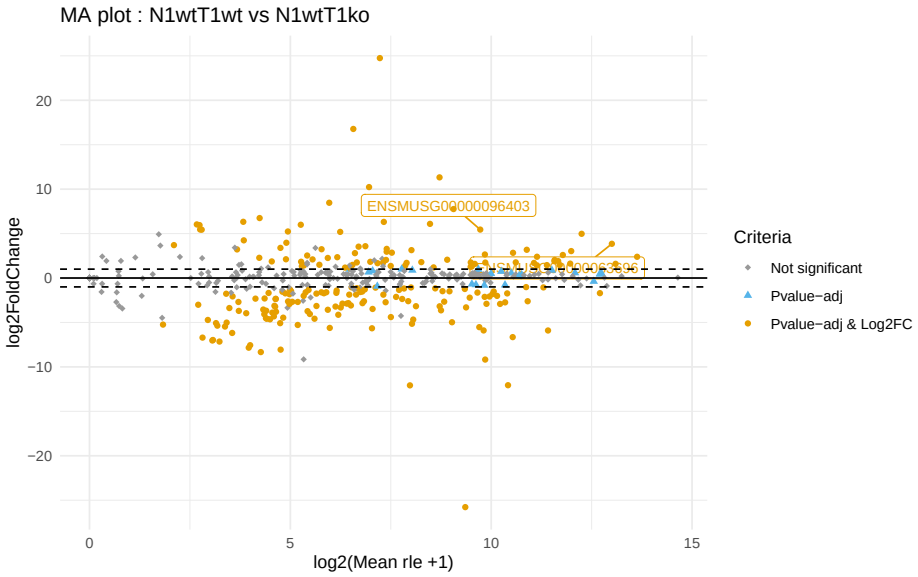

Volcano plot : N1wtT1ko vs N1haT1wt
−log10(padj)
log2FoldChange
ENSMUSG00000096403
ENSMUSG00000063696
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC


MA plot : N1wtT1ko vs N1haT1wt
log2FoldChange
log2(Mean rle +1)
ENSMUSG00000096403
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC


Volcano plot : N1wtT1wt vs N1haT1wt
−log10(padj)
log2FoldChange
ENSMUSG00000068457
ENSMUSG00000094433
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC


MA plot : N1wtT1wt vs N1haT1wt
ENSMUSG00000094433
ENSMUSG00000068457
log2FoldChange
log2(Mean rle +1)
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC


Volcano plot : N1wtT1wt vs N1wtT1ko
ENSMUSG00000096403
ENSMUSG00000063696
−log10(padj)
log2FoldChange
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC



If the user wants to save the graphs, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

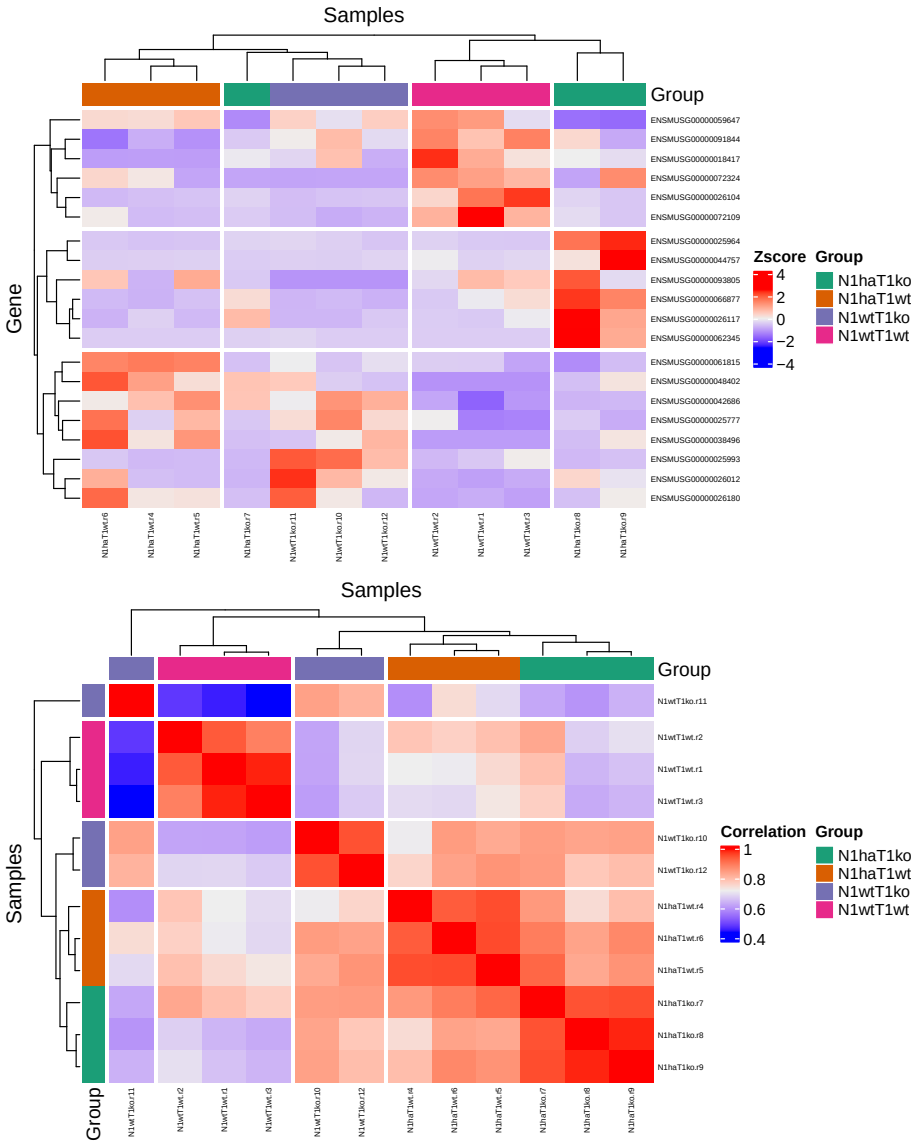
If the user wants to display the graph, the user must set `Display.plots=TRUE`.

Write `?DEplotVolcanoMA` in your console for more information about the function.

5.2.1.2 Heatmaps (case 1)

The following lines of code allow to plot a correlation heatmap between samples and a heatmap accross samples and genes for a subset of genes that can be selected by the user.

```
resplotHeatmapMus <- DEplotHeatmaps(SEresDE=ResDEMus500,
                                    ColumnsCriteria=2,#c(2,4),
                                    Set.Operation="union",
                                    NbGene.analysis=20,
                                    SizeLabelRows=5,
                                    SizeLabelCols=5,
                                    Display.plots=TRUE,
                                    Save.plots=FALSE)
```



For the rle heatmap (so the heatmap accross samples and subset of genes), the subset of genes is selected as follow

1. the user select one or more binary column with the input `ColumnsCriteria` which corresponds to the column number of `rowData(ResDEMus500)`
2. Then the selected genes will be
   - If `Set.Operation="union"` then the rows extracted from the different datasets included in `ResDEMus500` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(ResDEMus500)` by `ColumnsCriteria` is >0.
   - If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `ResDEMus500` are those such that the product of the selected columns of `SummarizedExperiment::rowData(ResDEMus500)` by `ColumnsCriteria` is >0.

- If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `ResDEMus500` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(ResDEMus500)` by `ColumnsCriteria` is >0.

3. Finally, the subset of genes will be the `NbGene.analysis` genes, among the selected genes, which have the highest sum of absolute log2 fold change.

If the user wants to save the graphs, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

If the user wants to display the graph, the user must set `Display.plots=TRUE`.

Write `?DEplotHeatmaps` in your console for more information about the function.

5.2.2 Fission data (case 2)

In this section we use the fission yeast subdataset **RawCounts_Leong2014_FISSIONsub500wt** (see the subsection Fission dataset) in order to explain **DEplotVolcanoMA()** and **DEplotHeatmaps()** in **case 2**.

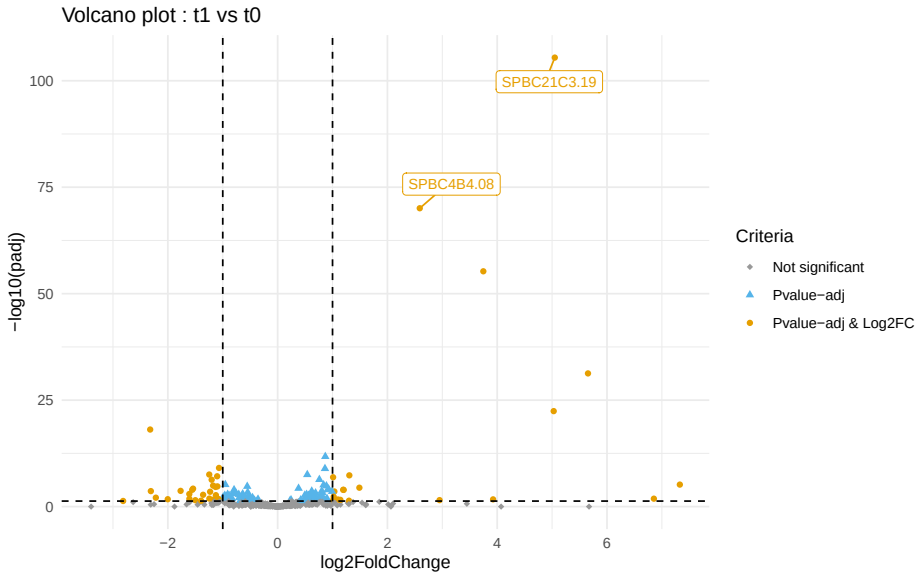
5.2.2.1 Volcano and MA plots (case 2)

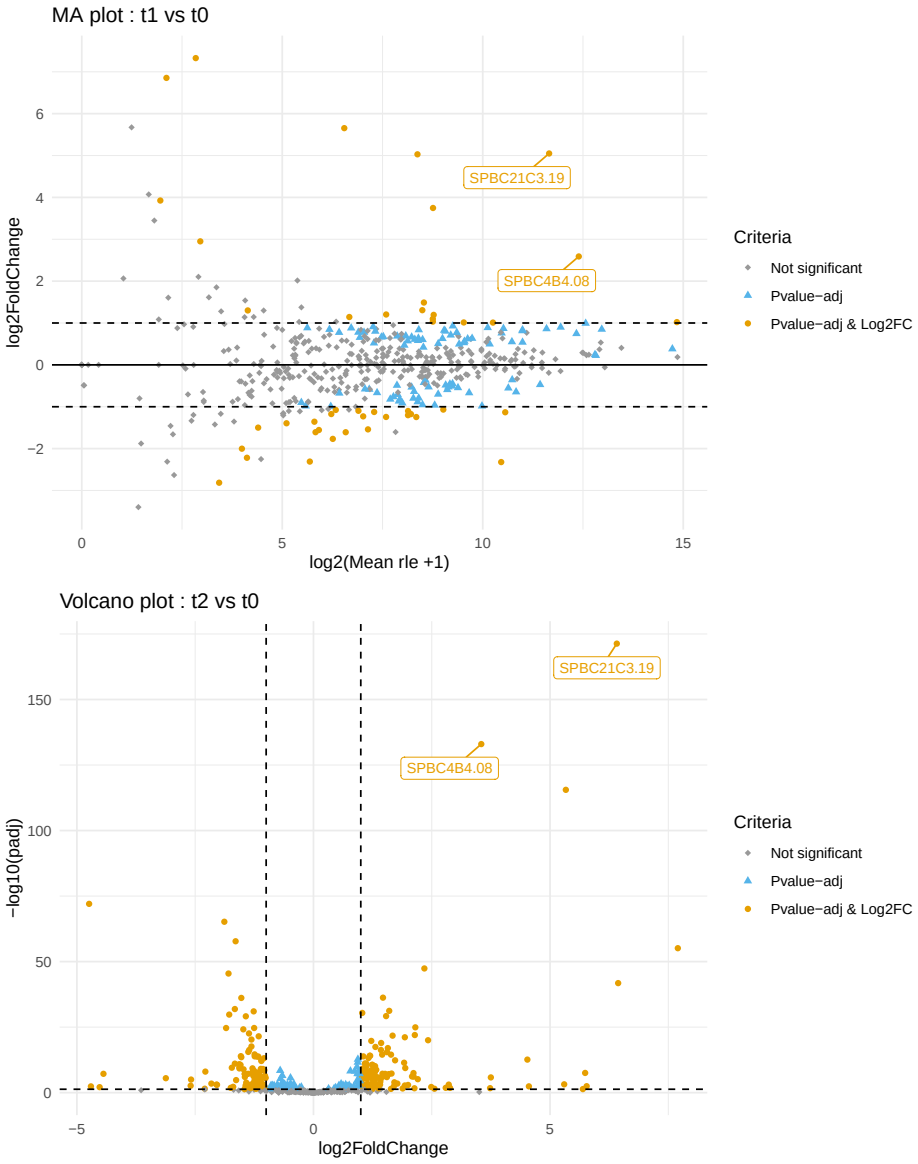
The following lines of code allow to plot

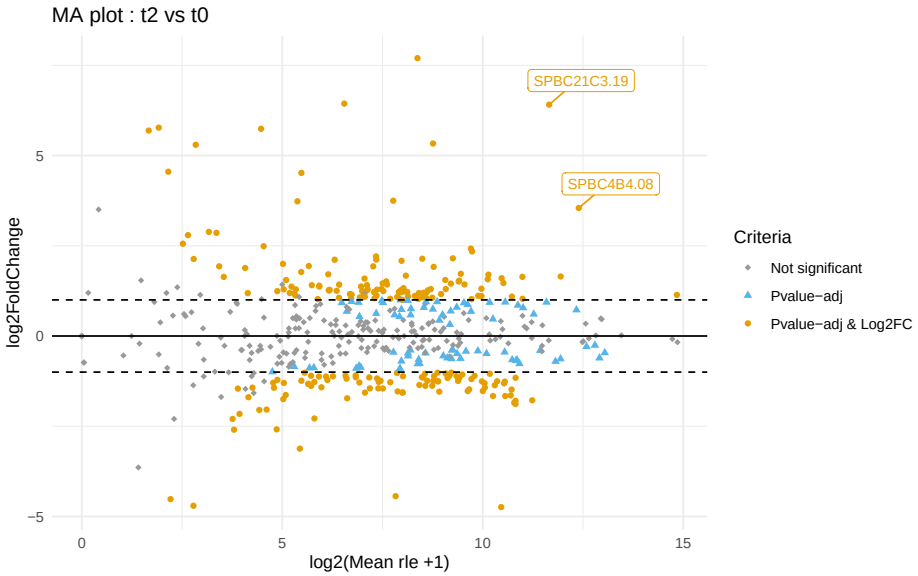
- $T-1=6-1=5$ volcano plots (with $T=6$ the number time points)
- $T-1=5$ MA plots with.

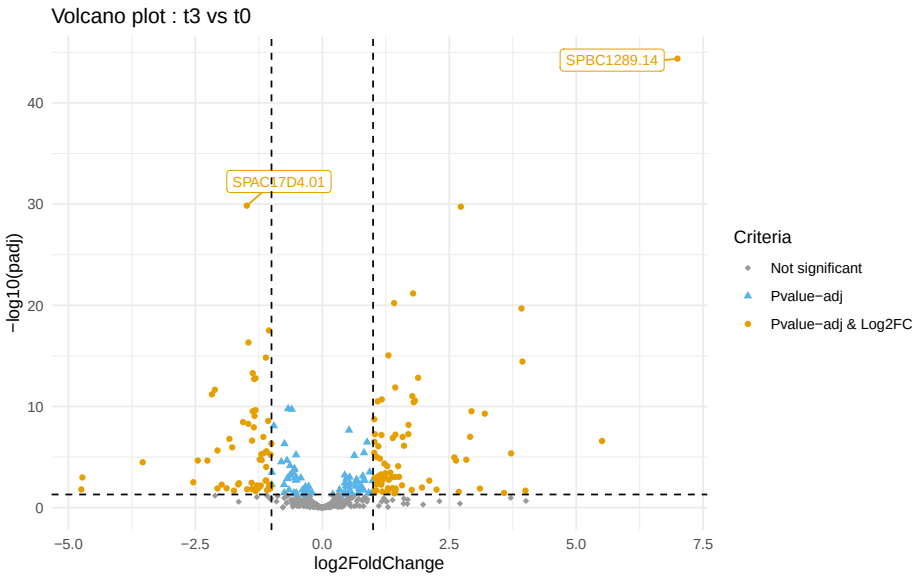
allowing to separate non DE genes, DE genes below a threshold of log2 fold change and DE genes above a threshold of log2 fold change.

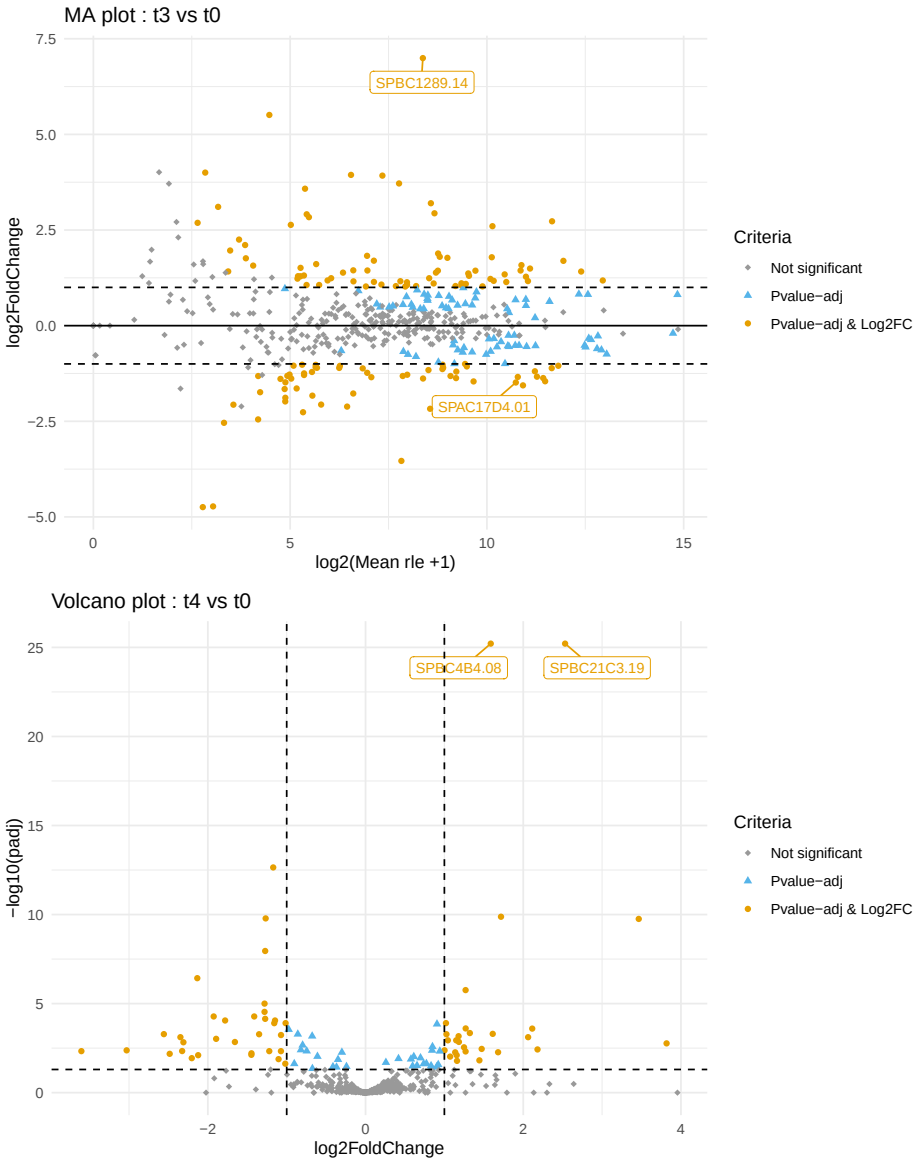
```
resVolcanoMAFission <- DEplotVolcanoMA(SEresDE=DEyeast500wt,
                                       NbGene.plotted=2,
                                       SizeLabel=3,
                                       Display.plots=TRUE,
                                       Save.plots=TRUE)
```

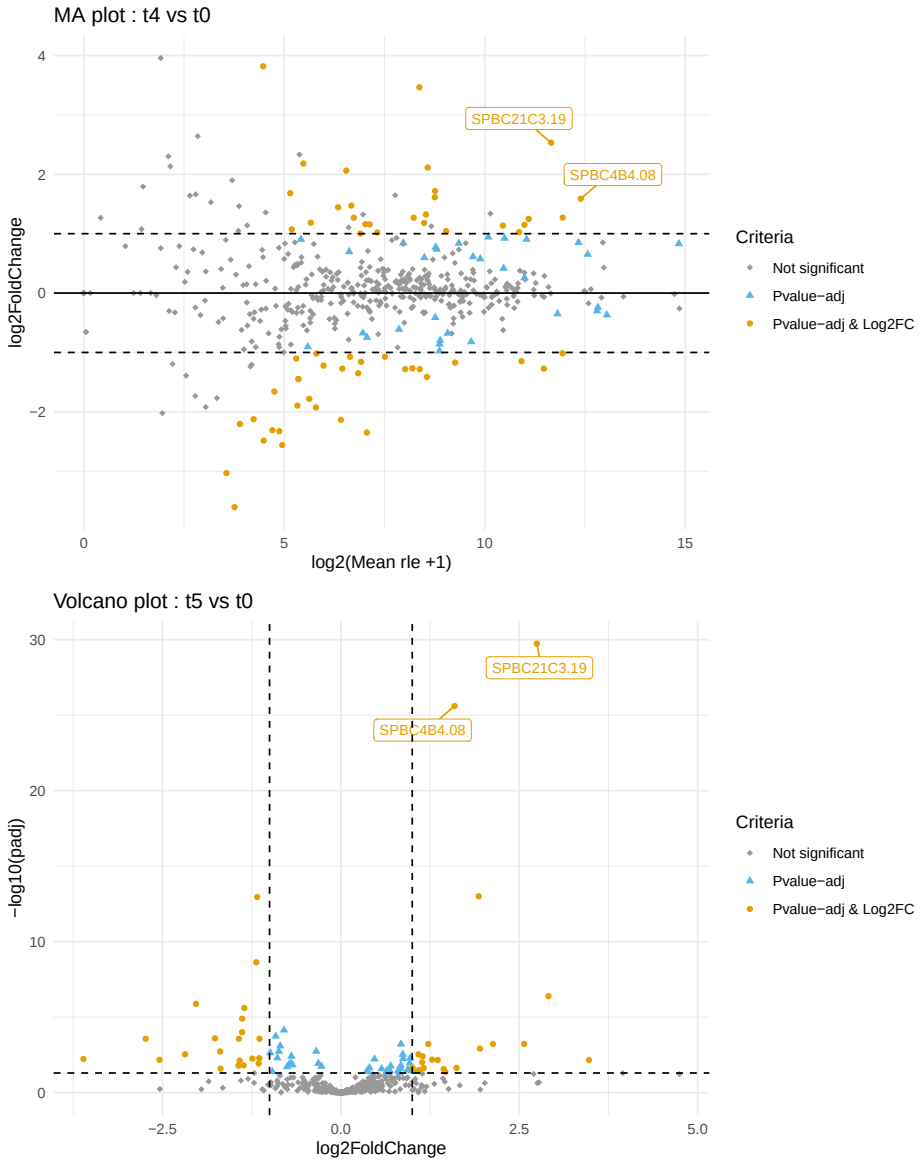


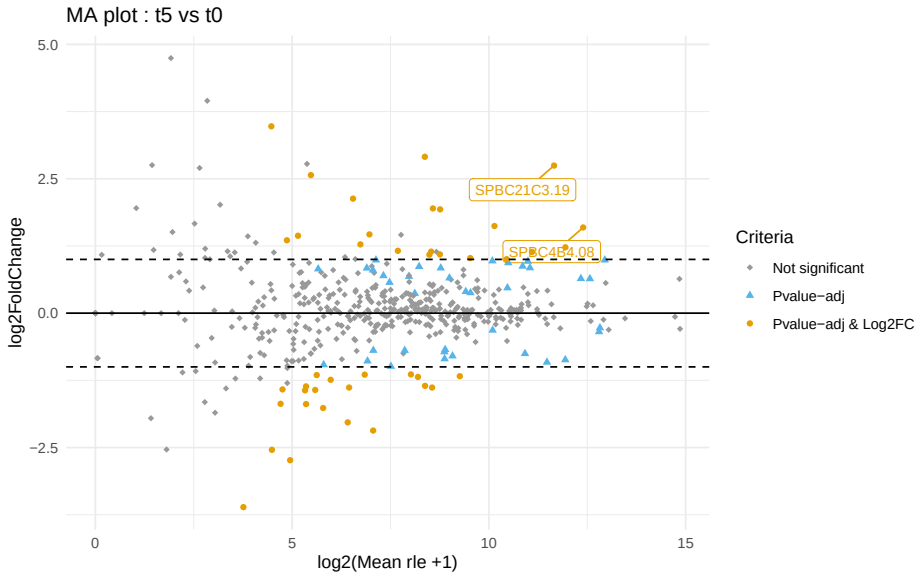

MA plot : t1 vs t0
log2FoldChange
log2(Mean rle +1)
SPBC21C3.19
SPBC4B4.08
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC
Volcano plot : t2 vs t0
−log10(padj)
log2FoldChange
SPBC21C3.19
SPBC4B4.08
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC


MA plot : t2 vs t0
SPBC21C3.19
SPBC4B4.08
log2FoldChange
log2(Mean rle +1)
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC


Volcano plot : t3 vs t0
SPBC1289.14
SPAC17D4.01
−log10(padj)
log2FoldChange
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC


MA plot : t3 vs t0
SPBC1289.14
SPAC17D4.01
log2FoldChange
log2(Mean rle +1)
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC
Volcano plot : t4 vs t0
SPBC4B4.08
SPBC21C3.19
−log10(padj)
log2FoldChange
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC


MA plot : t4 vs t0
log2FoldChange
log2(Mean rle +1)
SPBC21C3.19
SPBC4B4.08
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC
Volcano plot : t5 vs t0
−log10(padj)
log2FoldChange
SPBC21C3.19
SPBC4B4.08
Criteria
Not significant
Pvalue−adj
Pvalue−adj & Log2FC



If the user wants to save the graphs, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

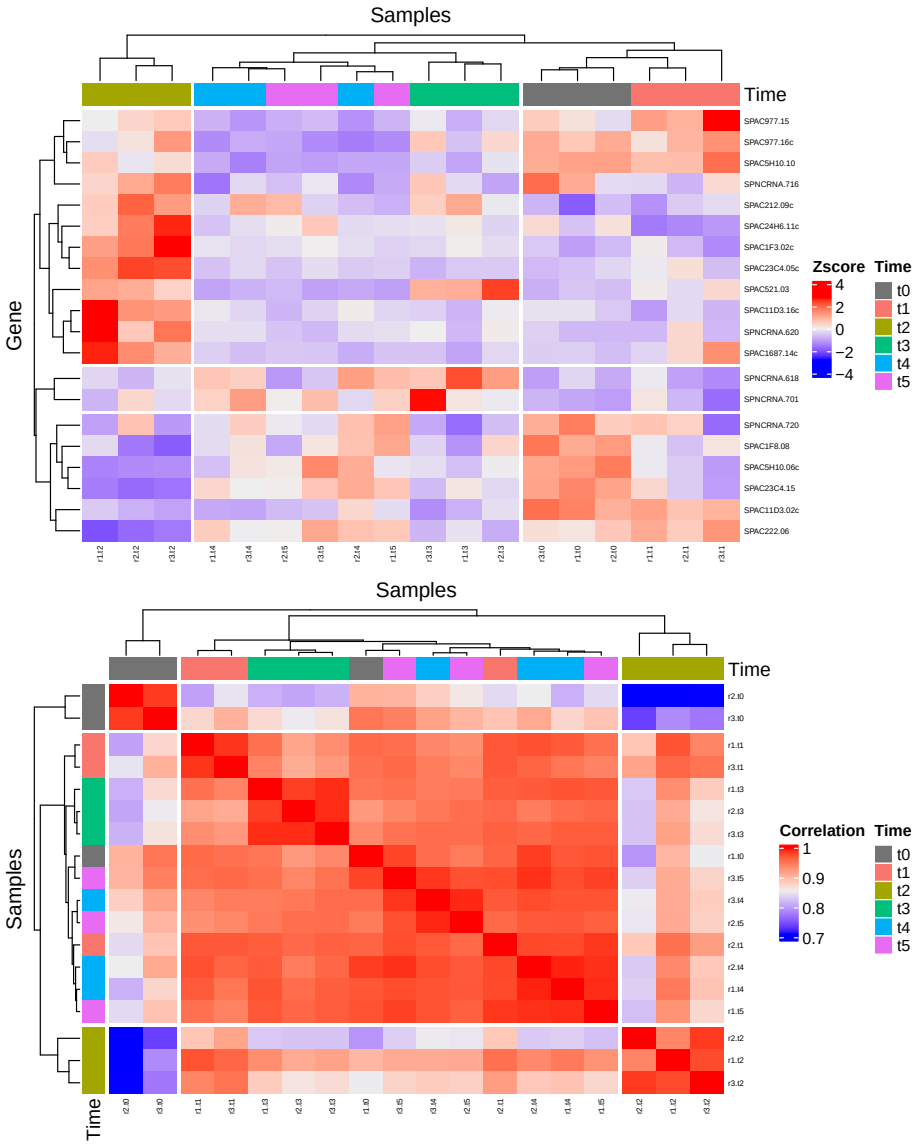
If the user wants to display the graph, the user must set `Display.plots=TRUE`.

Write `?DEplotVolcanoMA` in your console for more information about the function.

5.2.2.2 Heatmaps (case 2)

The following lines of code allow to plot a correlation heatmap between samples and a heatmap across samples and genes for a subset of genes that can be selected by the user.

```
resplotHeatmapFission <- DEplotHeatmaps(SEresDE=DEyeast500wt,
                                        ColumnsCriteria=2,
                                        Set.Operation="union",
                                        NbGene.analysis=20,
                                        Color.Group=NULL,
                                        SizeLabelRows=5,
                                        SizeLabelCols=5,
                                        Display.plots=TRUE,
                                        Save.plots=FALSE)
```



For the rle heatmap (so the heatmap accross samples and subset of genes), the subset of genes is selected as follow

1. the user select one or more binary column with the input `ColumnsCriteria` which corresponds to the column number of `rowData(DEyeast500wt)`
2. Then the selected genes will be
   - If `Set.Operation="union"` then the rows extracted from the different datasets included in `DEyeast500wt` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(DEyeast500wt)` by `ColumnsCriteria` is >0.
   - If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `DEyeast500wt` are those such that the product of the selected columns of `SummarizedExperiment::rowData(DEyeast500wt)` by `ColumnsCriteria` is >0.

- If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `DEyeast500wt` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(DEyeast500wt)` by `ColumnsCriteria` is >0.

3. Finally, the subset of genes will be the `NbGene.analysis` genes, among the selected genes, which have the highest sum of absolute log2 fold change.

If the user wants to save the graphs, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

If the user wants to display the graph, the user must set `Display.plots=TRUE`.

Write `?DEplotHeatmaps` in your console for more information about the function.

5.2.3 Leukemia data (case 3 with only two biological conditions)

In this section we use the Chronic lymphocytic leukemia (CLL) subdataset **RawCounts_Schleiss2021_CLLsub500** (see Subsection Leukemia dataset) in order to explain **DEplotVolcanoMA()** and **DEplotHeatmaps()** in **case 3** when there only two biological conditions.

5.2.3.1 Volcano and MA plots (case 3 two conditions)

The following lines of code allow to plot

- $\binom{N_{bc}}{2} \times T + (T-1) \times N_{bc} = \frac{N_{bc}(N_{bc}-1)}{2} \times T + (T-1) \times N_{bc} = 25$ volcano plots (with $N_{bc} = 2$ the number of biological conditions and $T = 9$ the number of time points).
- $\binom{N_{bc}}{2} \times T + (T-1) \times N_{bc} = 25$ MA plots.

allowing to separate non DE genes, DE genes below a threshold of log2 fold change and DE genes above a threshold of log2 fold change.

```
resVolcanoMACLL<-DEplotVolcanoMA(SEresDE=ResDELeuk500,
                                 NbGene.plotted=2,
                                 SizeLabel=3,
                                 Display.plots=FALSE,
                                 Save.plots=FALSE)
```

The graphs are a combination of the results described in subsection Volcano and MA plots (case 1) and subsection **??**. If the user wants to save the graphs, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

If the user wants to display the graph, the user must set `Display.plots=TRUE`.

Write `?DEplotVolcanoMA` in your console for more information about the function.

5.2.3.2 Heatmaps (case 3 two conditions)

The following lines of code allow to plot a correlation heatmap between samples and a heatmap across samples and genes for a subset of genes that can be selected by the user.

```
resplotHeatmapCLL<-DEplotHeatmaps(SEresDE=ResDELeuk500,
                                  ColumnsCriteria=c(12, 13),
                                  Set.Operation="union",
                                  NbGene.analysis=20,
                                  SizeLabelRows=5,
                                  SizeLabelCols=5,
                                  Display.plots=FALSE,
                                  Save.plots=FALSE)
```

Both graphs are similar to those described in subsection Heatmaps (case 2) and Heatmaps (case 1). For the rle heatmap (so the heatmap accross samples and subset of genes), the subset of genes is selected as follow

1. the user select one or more binary column with the input `ColumnsCriteria` which corresponds to the column number of `rowData(ResDELeuk500)`
2. Then the selected genes will be
   - If `Set.Operation="union"` then the rows extracted from the different datasets included in `ResDELeuk500` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(ResDELeuk500)` by `ColumnsCriteria` is >0.
   - If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `ResDELeuk500` are those such that the product of the selected columns of `SummarizedExperiment::rowData(ResDELeuk500)` by `ColumnsCriteria` is >0.
   - If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `ResDELeuk500` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(ResDELeuk500)` by `ColumnsCriteria` is >0.
3. Finally, the subset of genes will be the `NbGene.analysis` genes, among the selected genes, which have the highest sum of absolute log2 fold change.

If the user wants to save the graphs, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

If the user wants to display the graph, the user must set `Display.plots=TRUE`.

Write `?DEplotHeatmaps` in your console for more information about the function.

5.2.4 Mouse data 2 (case 3 with more than two biological conditions)

In this section we use the mouse subdataset **RawCounts_Weger2021_MOUSEsub500** (see Subsection Mouse data 2) in order to explain **DEplotVolcanoMA()** and **DEplotHeatmaps** in **case 3** when there are more than two biological conditions.

5.2.4.1 Volcano and MA plots (case 3 with more than two biological conditions)

The following lines of code allow to plot

- $\binom{N_{bc}}{2} \times T + (T-1) \times N_{bc} = \frac{N_{bc}(N_{bc}-1)}{2} \times T + (T-1) \times N_{bc} = 56$ volcano plots (with $N_{bc} = 4$ the number of biological conditions and $T = 6$ the number of time points).
- $\binom{N_{bc}}{2} \times T + (T-1) \times N_{bc} = 56$ MA plots.

allowing to separate non DE genes, DE genes below a threshold of log2 fold change and DE genes above a threshold of log2 fold change.

```
resVolcanoMAMouse2<-DEplotVolcanoMA(SEresDE=ResDEMusBmCrKoWt500,
                                    NbGene.plotted=2,
                                    SizeLabel=3,
                                    Display.plots=FALSE,
                                    Save.plots=FALSE)
```

The graphs are a combination of the results described in subsection Volcano and MA plots (case 1) and subsection **??**. If the user wants to save the graphs, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

If the user wants to display the graph, the user must set `Display.plots=TRUE`.

Write `?DEplotVolcanoMA` in your console for more information about the function.

5.2.4.2 Heatmaps (case 3 with more than two biological conditions)

The following lines of code allow to plot a correlation heatmap between samples and a heatmap across samples and genes for a subset of genes that can be selected by the user.

```
resplotHeatmapMouse2<-DEplotHeatmaps(SEresDE=ResDEMusBmCrKoWt500,
                                     ColumnsCriteria=2:5,
                                     Set.Operation="union",
                                     NbGene.analysis=20,
                                     SizeLabelRows=5,
                                     SizeLabelCols=5,
                                     Display.plots=FALSE,
                                     Save.plots=FALSE)
```

Both graphs are similar to those described in subsection Heatmaps (case 2) and Heatmaps (case 1). For the rle heatmap (so the heatmap accross samples and subset of genes), the subset of genes is selected as follow

1. the user select one or more binary column with the input `ColumnsCriteria` which corresponds to the column number of `rowData(ResDEMusBmCrKoWt500)`
2. Then the selected genes will be
   - If `Set.Operation="union"` then the rows extracted from the different datasets included in `ResDEMusBmCrKoWt500` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(ResDEMusBmCrKoWt500)` by `ColumnsCriteria` is >0.
   - If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `ResDEMusBmCrKoWt500` are those such that the product of the selected columns of `SummarizedExperiment::rowData(ResDEMusBmCrKoWt500)` by `ColumnsCriteria` is >0.
   - If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `ResDEMusBmCrKoWt500` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(ResDEMusBmCrKoWt500)` by `ColumnsCriteria` is >0.
3. Finally, the subset of genes will be the `NbGene.analysis` genes, among the selected genes, which have the highest sum of absolute log2 fold change.

If the user wants to save the graphs, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

If the user wants to display the graph, the user must set `Display.plots=TRUE`.

Write `?DEplotHeatmaps` in your console for more information about the function.

5.3 Gene Ontology (GO) analysis with GSEAQuickAnalysis() and GSEApreprocessing()

5.3.1 Mouse data (case 1)

In this section we use the mouse subdataset **RawCounts_Antoszewski2022_MOUSEsub500** (see the subsection Mouse dataset 1) in order to explain **GSEAQuickAnalysis()** and **GSEApreprocessing()** in **case 1**.

5.3.1.1 Gene ontology with the R package gprofiler2

The lines of code below realize an enrichment analysis with the R package gprofiler2 for a selection of genes. Beware, an internet connection is needed. The function returns

- a data.frame (output `metadata(resGSEAgprofiler2Mus)$Rgprofiler2$GSEAresults`) giving information about all detected gene ontologies the list of associated genes.
- a lollipop graph (see section Gene ontology and gene enrichment). The y-axis indicates the `MaxNumberGO` most significant gene ontologies and pathways associated to the selected DE genes. The gene ontologies and patways are sorted into descending order. The x-axis indicates the -log10(pvalues). The higher is a lollipop the more significant is a gene ontology or pathway. A lollipop is yellow if the pvalues is smaller than 0.05 (significant) and blue otherwise.
- A mahattan plot (see section Gene ontology and gene enrichment) indicating all genes ontologies ordered according to the functional database (G0::BP, G0::CC, G0::MF and KEGG)

```
resGSEAgprofiler2Mus<-GSEAQuickAnalysis(Internect.Connection=FALSE,
                                        SEresDE=ResDEMus500,
                                        ColumnsCriteria=2,
                                        ColumnsLog2ordered=NULL,
                                        Set.Operation="union",
                                        Organism="mmusculus",
                                        MaxNumberGO=10,
                                        Background=FALSE,
                                        Display.plots=FALSE,
                                        Save.plots=FALSE)
#
# head(4Vectors::metadata(resGSEAgprofiler2Mus)$Rgprofiler2$GSEAresults)
```

As **GSEAQuickAnalysis()** requires an internet connection, the input `Internect.Connection` is set by default to `FALSE`. Once the user is sure to have an internet connection, the user may set `Internect.Connection=TRUE` in order to realize the enrichment analysis.

The subset of genes is selected as follow

1. the user select one or more binary column with the input `ColumnsCriteria` which corresponds to the column number of `rowData(ResDEMus500)`
2. Then the subset of genes will be
   - If `Set.Operation="union"` then the rows extracted from the different datasets included in `ResDEMus500` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(ResDEMus500)` by `ColumnsCriteria` is >0.

- If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `ResDEMus500` are those such that the product of the selected columns of `SummarizedExperiment::rowData(ResDEMus500)` by `ColumnsCriteria` is >0.
- If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `ResDEMus500` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(ResDEMus500)` by `ColumnsCriteria` is >0.

If `ColumnsLog2ordered` is a vector of integers, it corresponds to the columns number of `rowData(ResDEMus500)`, the output of `DEanalysisGlobal()`, which must contains log_2 fold change values. The rows of `rowData(ResDEMus500)` (corresponding to genes) will be decreasingly ordered according to the sum of absolute log_2 fold change (the selected columns must contain log_2 fold change values) before the enrichment analysis. The enrichment analysis will take into account the genes order as the first genes will be considered to have the highest biological importance and the last genes the lowest.

If the user wants to save the graphs, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

Write `?GSEAQuickAnalysis` in your console for more information about the function.

5.3.1.2 Preprocessing for GSEA, DAVID, WebGestalt, gProfiler, Panther, ShinyGO, Enrichr and GOrilla

The following lines of code will generate all files, for a selection of genes, in order to use the following software and online tools : GSEA, DAVID, WebGestalt, gProfiler, Panther, ShinyGO, Enrichr and GOrilla.

```
resGSEApreprocessingFission<-GSEApreprocessing(SEresDE=ResDEMus500,
                                               ColumnsCriteria=2,
                                               Set.Operation="union",
                                               Rnk.files=FALSE,
                                               Save.files=FALSE)
```

The subset of genes is selected as follow

1. the user select one or more binary column with the input `ColumnsCriteria` which corresponds to the column number of `rowData(ResDEMus500)`
2. Then the subset of genes will be
   - If `Set.Operation="union"` then the rows extracted from the different datasets included in `ResDEMus500` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(ResDEMus500)` by `ColumnsCriteria` is >0.
   - If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `ResDEMus500` are those such that the product of the selected columns of `SummarizedExperiment::rowData(ResDEMus500)` by `ColumnsCriteria` is >0.

- If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `ResDEMus500` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(ResDEMus500)` by `ColumnsCriteria` is >0.

If the user wants to save the files, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

Write `?GSEApreprocessing` in your console for more information about the function.

5.3.2 Fission data (case 2)

In this section we use the fission yeast subdataset **RawCounts_Leong2014_FISSIONsub500wt** (see the subsection Fission dataset) in order to explain **GSEAQuickAnalysis()** and **GSEApreprocessing()** in **case 2**.

5.3.2.1 Gene ontology with the R package gprofiler2

The lines of code below realize an enrichment analysis with the R package gprofiler2 for a selection of genes. Beware, an internet connection is needed. The function returns

- a data.frame (output `metadata(resGSEAgprofiler2Fission)$Rgprofiler2$GSEAresults`) giving information about all detected gene ontologies the list of associated genes.
- a lollipop graph (see section Gene ontology and gene enrichment). The y-axis indicates the `MaxNumberGO` most significant gene ontologies and pathways associated to the selected DE genes. The gene ontologies and pathways are sorted into descending order. The x-axis indicates the $-log10(pvalues)$. The higher is a lollipop the more significant is a gene ontology or pathway. A lollipop is yellow if the pvalues is smaller than 0.05 (significant) and blue otherwise.
- A Manhattan plot (see section Gene ontology and gene enrichment) indicating all genes ontologies ordered according to the functional database (G0::BP, G0::CC, G0::MF and KEGG)

```
resGSEAgprofiler2Fission<-GSEAQuickAnalysis(Internect.Connection=FALSE,
                                            SEresDE=DEyeast500wt,
                                            ColumnsCriteria=2,
                                            ColumnsLog2ordered=NULL,
                                            Set.Operation="union",
                                            Organism="spombe",
                                            MaxNumberGO=20,
                                            Background=FALSE,
                                            Display.plots=FALSE,
                                            Save.plots=FALSE)
#
# head(resGSEAgprofiler2Fission$GSEAresults)
```

As **GSEAQuickAnalysis()** requires an internet connection, the input `Internect.Connection` is set by default to `FALSE`. Once the user is sure to have an internet connection, the user may set `Internect.Connection=TRUE` in order to realize the enrichment analysis.

The subset of genes is selected as follow

1. the user select one or more binary column with the input `ColumnsCriteria` which corresponds to the column number of `rowData(DEyeast500wt)`
2. Then the subset of genes will be
   - If `Set.Operation="union"` then the rows extracted from the different datasets included in `DEyeast500wt` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(DEyeast500wt)` by `ColumnsCriteria` is >0.
   - If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `DEyeast500wt` are those such that the product of the selected columns of `SummarizedExperiment::rowData(DEyeast500wt)` by `ColumnsCriteria` is >0.
   - If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `DEyeast500wt` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(DEyeast500wt)` by `ColumnsCriteria` is >0.

If `ColumnsLog2ordered` is a vector of integers, it corresponds to the columns number of `rowData(DEyeast500wt)`, the output of `DEanalysisGlobal()`, which must contains log_2 fold change values. The rows of `rowData(DEyeast500wt)` (corresponding to genes) will be decreasingly ordered according to the sum of absolute log_2 fold change (the selected columns must contain log_2 fold change values) before the enrichment analysis. The enrichment analysis will take into account the genes order as the first genes will be considered to have the highest biological importance and the last genes the lowest.

If the user wants to save the graphs, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

Write `?GSEAQuickAnalysis` in your console for more information about the function.

5.3.2.2 Preprocessing for GSEA, DAVID, WebGestalt, gProfiler, Panther, ShinyGO, Enrichr and GOrilla

The following lines of code will generate all files, for a selection of genes, in order to use the following software and online tools : GSEA, DAVID, WebGestalt, gProfiler, Panther, ShinyGO, Enrichr and GOrilla.

```
resGSEApreprocessingFission<-GSEApreprocessing(SEresDE=DEyeast500wt,
                                               ColumnsCriteria=2,
                                               Set.Operation="union",
                                               Rnk.files=FALSE,
                                               Save.files=FALSE)
```

The subset of genes is selected as follow

1. the user select one or more binary column with the input `ColumnsCriteria` which corresponds to the column number of `rowData(DEyeast500wt)`
2. Then the subset of genes will be

- If `Set.Operation="union"` then the rows extracted from the different datasets included in `DEyeast500wt` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(DEyeast500wt)` by `ColumnsCriteria` is >0.
- If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `DEyeast500wt` are those such that the product of the selected columns of `SummarizedExperiment::rowData(DEyeast500wt)` by `ColumnsCriteria` is >0.
- If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `DEyeast500wt` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(DEyeast500wt)` by `ColumnsCriteria` is >0.

If the user wants to save the files, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

Write `?GSEApreprocessing` in your console for more information about the function.

5.3.3 Leukemia data (case 3 with only two biological conditions)

In this section we use the Chronic lymphocytic leukemia (CLL) subdataset **RawCounts_Schleiss2021_CLLsub500** (see Subsection Leukemia dataset) in order to explain **GSEAQuickAnalysis()** and **GSEApreprocessing()** in **case 3** when there only two biological conditions.

5.3.3.1 Gene ontology with the R package gprofiler2

The lines of code below realize an enrichment analysis with the R package gprofiler2 for a selection of genes. Beware, an internet connection is needed. The function returns

- a data.frame (output `metadata(resGSEAgprofiler2Leuk)$Rgprofiler2$GSEAresults`) giving information about all detected gene ontologies the list of associated genes.
- a lollipop graph (see section Gene ontology and gene enrichment). The y-axis indicates the `MaxNumberGO` most significant gene ontologies and pathways associated to the selected DE genes. The gene ontologies and patways are sorted into descending order. The x-axis indicates the -log10(pvalues). The higher is a lollipop the more significant is a gene ontology or pathway. A lollipop is yellow if the pvalues is smaller than 0.05 (significant) and blue otherwise.
- A manhattan plot (see section Gene ontology and gene enrichment) indicating all genes ontologies ordered according to the functional database (G0::BP, G0::CC, G0::MF and KEGG)

```
resGSEAgprofiler2Leuk<-GSEAQuickAnalysis(Internect.Connection=FALSE,
                                         SEresDE=ResDELeuk500,
                                         ColumnsCriteria=c(12,13),
                                         ColumnsLog2ordered=NULL,
                                         Set.Operation="union",
```

```
                                        Organism="hsapiens",
                                        MaxNumberGO=20,
                                        Background=FALSE,
                                        Display.plots=FALSE,
                                        Save.plots=FALSE)
#
# head(resGSEAgprofiler2Leuk$GSEAresults)
```

As **GSEAQuickAnalysis()** requires an internet connection, the input `Internect.Connection` is set by default to `FALSE`. Once the user is sure to have an internet connection, the user may set `Internect.Connection=TRUE` in order to realize the enrichment analysis.

The subset of genes is selected as follow

1. the user select one or more binary column with the input `ColumnsCriteria` which corresponds to the column number of `rowData(ResDELeuk500)`
2. Then the subset of genes will be

- If `Set.Operation="union"` then the rows extracted from the different datasets included in `ResDELeuk500` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(ResDELeuk500)` by `ColumnsCriteria` is >0.
- If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `ResDELeuk500` are those such that the product of the selected columns of `SummarizedExperiment::rowData(ResDELeuk500)` by `ColumnsCriteria` is >0.
- If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `ResDELeuk500` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(ResDELeuk500)` by `ColumnsCriteria` is >0.

If `ColumnsLog2ordered` is a vector of integers, it corresponds to the columns number of `rowData(ResDELeuk500)`, the output of `DEanalysisGlobal()`, which must contains log_2 fold change values. The rows of `rowData(ResDELeuk500)` (corresponding to genes) will be decreasingly ordered according to the sum of absolute log_2 fold change (the selected columns must contain log_2 fold change values) before the enrichment analysis. The enrichment analysis will take into account the genes order as the first genes will be considered to have the highest biological importance and the last genes the lowest.

If the user wants to save the graphs, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

Write `?GSEAQuickAnalysis` in your console for more information about the function.

5.3.3.2 Preprocessing for GSEA, DAVID, WebGestalt, gProfiler, Panther, ShinyGO, Enrichr and GOrilla

The following lines of code will generate all files, for a selection of genes, in order to use the following software and online tools : GSEA, DAVID, WebGestalt, gProfiler, Panther, ShinyGO, Enrichr and GOrilla.

```
resGSEApreprocessingLeuk<-GSEApreprocessing(SEresDE=ResDELeuk500,
                                            ColumnsCriteria=c(12,13),
                                            Set.Operation="union",
                                            Rnk.files=FALSE,
                                            Save.files=FALSE)
```

The subset of genes is selected as follow

1. the user select one or more binary column with the input `ColumnsCriteria` which corresponds to the column number of `ResDELeuk500$DE.results`
2. Then the subset of genes will be
   - If `Set.Operation="union"` then the rows extracted from the different datasets included in `ResDELeuk500` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(ResDELeuk500)` by `ColumnsCriteria` is >0.
   - If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `ResDELeuk500` are those such that the product of the selected columns of `SummarizedExperiment::rowData(ResDELeuk500)` by `ColumnsCriteria` is >0.
   - If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `ResDELeuk500` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(ResDELeuk500)` by `ColumnsCriteria` is >0.

If the user wants to save the files, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

Write `?GSEApreprocessing` in your console for more information about the function.

5.3.4 Mouse data 2 (case 3 with more than two biological conditions)

In this section we use the mouse subdataset **RawCounts_Weger2021_MOUSEsub500** (see Subsection Mouse data 2) in order to explain **GSEAQuickAnalysis()** and **GSEApreprocessing()** in **case 3** when there are more than two biological conditions.

5.3.4.1 Gene ontology with the R package gprofiler2

The lines of code below realize an enrichment analysis with the R package gprofiler2 for a selection of genes. Beware, an internet connection is needed. The function returns

- a data.frame (output `metadata(resGSEAgprofiler2Mouse2)$Rgprofiler2$GSEAresults`) giving information about all detected gene ontologies the list of associated genes.
- a lollipop graph (see section Gene ontology and gene enrichment). The y-axis indicates the `MaxNumberGO` most significant gene ontologies and pathways associated to the selected DE genes. The gene ontologies and patways are sorted into descending order. The x-axis indicates the -log10(pvalues). The higher is a lollipop the more significant is a gene ontology or pathway. A lollipop is yellow if the pvalues is smaller than 0.05 (significant) and blue otherwise.

- A manhattan plot (see section Gene ontology and gene enrichment) indicating all genes ontologies ordered according to the functional database (G0::BP, G0::CC, G0::MF and KEGG)

```
resGSEAgprofiler2Mouse2<-GSEAQuickAnalysis(Internect.Connection=FALSE,
                                           SEresDE=ResDEMusBmCrKoWt500,
                                           ColumnsCriteria=2:5,
                                           ColumnsLog2ordered=NULL,
                                           Set.Operation="union",
                                           Organism="mmusculus",
                                           MaxNumberGO=20,
                                           Background=FALSE,
                                           Display.plots=FALSE,
                                           Save.plots=FALSE)
#
# head(resGSEAgprofiler2Mouse2$GSEAresults)
```

As **GSEAQuickAnalysis()** requires an internet connection, the input `Internect.Connection` is set by default to `FALSE`. Once the user is sure to have an internet connection, the user may set `Internect.Connection=TRUE` in order to realize the enrichment analysis.

The subset of genes is selected as follow

1. the user select one or more binary column with the input `ColumnsCriteria` which corresponds to the column number of `resGSEAgprofiler2Mouse2$DE.results`
2. Then the subset of genes will be

- If `Set.Operation="union"` then the rows extracted from the different datasets included in `ResDEMusBmCrKoWt500` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(ResDEMusBmCrKoWt500)` by `ColumnsCriteria` is >0.
- If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `ResDEMusBmCrKoWt500` are those such that the product of the selected columns of `SummarizedExperiment::rowData(ResDEMusBmCrKoWt500)` by `ColumnsCriteria` is >0.
- If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `ResDEMusBmCrKoWt500` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(ResDEMusBmCrKoWt500)` by `ColumnsCriteria` is >0.

If `ColumnsLog2ordered` is a vector of integers, it corresponds to the columns number of `rowData(ResDEMusBmCrKoWt500)`, the output of `DEanalysisGlobal()`, which must contains log_2 fold change values. The rows of `rowData(ResDEMusBmCrKoWt500)` (corresponding to genes) will be decreasingly ordered according to the sum of absolute log_2 fold change (the selected columns must contain log_2 fold change values) before the enrichment analysis. The enrichment analysis will take into account the genes order as the first genes will be considered to have the highest biological importance and the last genes the lowest.

If the user wants to save the graphs, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

Write `?GSEAQuickAnalysis` in your console for more information about the function.

5.3.4.2 Preprocessing for GSEA, DAVID, WebGestalt, gProfiler, Panther, ShinyGO, Enrichr and GOrilla

The following lines of code will generate all files, for a selection of genes, in order to use the following software and online tools : GSEA, DAVID, WebGestalt, gProfiler, Panther, ShinyGO, Enrichr and GOrilla.

```
resGSEApreprocessing<-GSEApreprocessing(SEresDE=ResDEMusBmCrKoWt500,
                                        ColumnsCriteria=2:5,
                                        Set.Operation="union",
                                        Rnk.files=FALSE,
                                        Save.files=TRUE)
```

The subset of genes is selected as follow

1. the user select one or more binary column with the input `ColumnsCriteria` which corresponds to the column number of `ResDEMusBmCrKoWt500$DE.results`
2. Then the subset of genes will be

- If `Set.Operation="union"` then the rows extracted from the different datasets included in `ResDEMusBmCrKoWt500` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(ResDEMusBmCrKoWt500)` by `ColumnsCriteria` is >0.
- If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `ResDEMusBmCrKoWt500` are those such that the product of the selected columns of `SummarizedExperiment::rowData(ResDEMusBmCrKoWt500)` by `ColumnsCriteria` is >0.
- If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `ResDEMusBmCrKoWt500` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(ResDEMusBmCrKoWt500)` by `ColumnsCriteria` is >0.

If the user wants to save the files, the input `Save.plots` must be

- either `Save.plots=TRUE`, and the graph will be saved in the same location than the input `path.result` of the function `DEanalysisGlobal()`.
- either a strings of characters giving the path to a folder where the graphs will be saved. The user then choose the path of the folder where results can be saved.

Write `?GSEApreprocessing` in your console for more information about the function.

6 Session info

Here is the output of `sessionInfo()` on the system on which this document was compiled.

- R version 4.3.1 Patched (2023-06-17 r84564), `x86_64-apple-darwin20`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Time zone: `America/New_York`
- TZcode source: `internal`
- Running under: `macOS Monterey 12.6.5`
- Matrix products: default
- BLAS: `/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRblas.0.dylib`
- LAPACK: `/Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRlapack.dylib` ; LAPACK version3.11.0
- Base packages: base, datasets, grDevices, graphics, methods, stats, tcltk, utils
- Other packages: Biobase 2.62.0, BiocGenerics 0.48.0, BiocStyle 2.30.0, DESeq2 1.42.0, DynDoc 1.80.0, GenomicRanges 1.54.0, IRanges 2.36.0, Mfuzz 2.62.0, MultiRNAflow 1.0.0, S4Vectors 0.40.0, SummarizedExperiment 1.32.0, e1071 1.7-13, knitr 1.44, widgetTools 1.80.0
- Loaded via a namespace (and not attached): BiocManager 1.30.22, BiocParallel 1.36.0, Cairo 1.6-1, ComplexHeatmap 2.18.0, DT 0.30, DelayedArray 0.28.0, FactoMineR 2.9, GenomeInfoDb 1.38.0, GenomeInfoDbData 1.2.11, GetoptLong 1.0.5, GlobalOptions 0.1.2, MASS 7.3-60, Matrix 1.6-1.1, MatrixGenerics 1.14.0, R6 2.5.1, RColorBrewer 1.1-3, RCurl 1.98-1.12, Rcpp 1.0.11, S4Arrays 1.2.0, SparseArray 1.2.0, TH.data 1.1-2, UpSetR 1.4.0, XVector 0.42.0, abind 1.4-5, backports 1.4.1, base64enc 0.1-3, bitops 1.0-7, bookdown 0.36, broom 1.0.5, bslib 0.5.1, cachem 1.0.8, car 3.1-2, carData 3.0-5, circlize 0.4.15, class 7.3-22, cli 3.6.1, clue 0.3-65, cluster 2.1.4, coda 0.19-4, codetools 0.2-19, colorspace 2.1-0, compiler 4.3.1, crayon 1.5.2, data.table 1.14.8, dendextend 1.17.1, digest 0.6.33, doParallel 1.0.17, dplyr 1.1.3, emmeans 1.8.9, estimability 1.4.1, evaluate 0.22, factoextra 1.0.7, fansi 1.0.5, farver 2.1.1, fastmap 1.1.1, flashClust 1.01-2, foreach 1.5.2, generics 0.1.3, ggalluvial 0.12.5, ggplot2 3.4.4, ggpubr 0.6.0, ggrepel 0.9.4, ggsci 3.0.0, ggsignif 0.6.4, glue 1.6.2, gprofiler2 0.2.2, grid 4.3.1, gridExtra 2.3, gtable 0.3.4, highr 0.10, htmltools 0.5.6.1, htmlwidgets 1.6.2, httr 1.4.7, iterators 1.0.14, jquerylib 0.1.4, jsonlite 1.8.7, labeling 0.4.3, lattice 0.22-5, lazyeval 0.2.2, leaps 3.1, lifecycle 1.0.3, locfit 1.5-9.8, magick 2.8.1, magrittr 2.0.3, matrixStats 1.0.0, misc3d 0.9-1, multcomp 1.4-25, multcompView 0.1-9, munsell 0.5.0, mvtnorm 1.2-3, parallel 4.3.1, pillar 1.9.0, pkgconfig 2.0.3, plot3D 1.4, plot3Drgl 1.0.4, plotly 4.10.3, plyr 1.8.9, png 0.1-8, proxy 0.4-27, purrr 1.0.2, reshape2 1.4.4, rgl 1.2.1, rjson 0.2.21, rlang 1.1.1, rmarkdown 2.25, rstatix 0.7.2, sandwich 3.0-2, sass 0.4.7, scales 1.2.1, scatterplot3d 0.3-44, shape 1.4.6, splines 4.3.1, stats4 4.3.1, stringi 1.7.12, stringr 1.5.0, survival 3.5-7, tibble 3.2.1, tidyr 1.3.0, tidyselect 1.2.0, tkWidgets 1.80.0, tools 4.3.1, utf8 1.2.4, vctrs 0.6.4, viridis 0.6.4, viridisLite 0.4.2, withr 2.5.1, xfun 0.40, xtable 1.8-4, yaml 2.3.7, zlibbioc 1.48.0, zoo 1.8-12

References

[Anders and Huber, 2010] Anders, S. and Huber, W. (2010). Differential expression analysis for sequence count data. 11(10):R106.

[Antoszewski et al., 2022] Antoszewski, M., Fournier, N., Ruiz Buendía, G. A., Lourenco, J., Liu, Y., Sugrue, T., Dubey, C., Nkosi, M., Pritchard, C. E. J., Huijbers, I. J., Segat, G. C., Alonso-Moreno, S., Serracanta, E., Belver, L., Ferrando, A. A., Ciriello, G., Weng, A. P., Koch, U., and Radtke, F. (2022). Tcf1 is essential for initiation of oncogenic notch1-driven chromatin topology in t-ALL. 139(16):2483–2498.

[Bar-Joseph et al., 2012] Bar-Joseph, Z., Gitter, A., and Simon, I. (2012). Studying and modelling dynamic biological processes using time-series gene expression data. 13(8):552–564.

[Chao et al., 2021] Chao, K.-H., Hsiao, Y.-W., Lee, Y.-F., Lee, C.-Y., Lai, L.-C., Tsai, M.-H., Lu, T.-P., and Chuang, E. Y. (2021). RNASeqR: An r package for automated two-group RNA-seq analysis workflow. 18(5):2023–2031.

[Conway et al., 2017] Conway, J. R., Lex, A., and Gehlenborg, N. (2017). UpSetR: an r package for the visualization of intersecting sets and their properties. 33(18):2938–2940.

[Eden et al., 2009] Eden, E., Navon, R., Steinfeld, I., Lipson, D., and Yakhini, Z. (2009). GOrilla: a tool for discovery and visualization of enriched GO terms in ranked gene lists. 10(1):48.

[Futschik and Carlisle, 2005] Futschik, M. E. and Carlisle, B. (2005). Noise-robust soft clustering of gene expression time-course data. 03(4):965–988.

[Ge et al., 2020] Ge, S. X., Jung, D., and Yao, R. (2020). ShinyGO: a graphical gene-set enrichment tool for animals and plants. 36(8):2628–2629.

[Gu et al., 2016] Gu, Z., Eils, R., and Schlesner, M. (2016). Complex heatmaps reveal patterns and correlations in multidimensional genomic data. 32(18):2847–2849.

[Kolberg et al., 2020] Kolberg, L., Raudvere, U., Kuzmin, I., Vilo, J., and Peterson, H. (2020). gprofiler2 – an r package for gene list functional enrichment analysis and namespace conversion toolset g:profiler. 9:709.

[Kuleshov et al., 2016] Kuleshov, M. V., Jones, M. R., Rouillard, A. D., Fernandez, N. F., Duan, Q., Wang, Z., Koplev, S., Jenkins, S. L., Jagodnik, K. M., Lachmann, A., McDermott, M. G., Monteiro, C. D., Gundersen, G. W., and Ma'ayan, A. (2016). Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. 44:W90–W97.

[Kumar and Futschik, 2007] Kumar, L. and Futschik, M. E. (2007). Mfuzz: A software package for soft clustering of microarray data. 2(1):5–7.

[Lataretu and Hölzer, 2020] Lataretu, M. and Hölzer, M. (2020). RNAflow: An effective and simple RNA-seq differential gene expression pipeline using nextflow. 11(12):1487.

[Leong et al., 2014] Leong, H. S., Dawson, K., Wirth, C., Li, Y., Connolly, Y., Smith, D. L., Wilkinson, C. R. M., and Miller, C. J. (2014). A global non-coding RNA system modulates fission yeast protein levels in response to stress. 5(1):3947.

[Li et al., 2010] Li, B., Ruotti, V., Stewart, R. M., Thomson, J. A., and Dewey, C. N. (2010). RNA-seq gene expression estimation with read mapping uncertainty. 26(4):493–500.

[Liao et al., 2019] Liao, Y., Wang, J., Jaehnig, E. J., Shi, Z., and Zhang, B. (2019). WebGestalt 2019: gene set analysis toolkit with revamped UIs and APIs. 47:W199–W205.

[Love et al., 2014] Love, M. I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. 15(12):550.

[Lê et al., 2008] Lê, S., Josse, J., and Husson, F. (2008). **FactoMineR** : An *R* package for multivariate analysis. 25(1).

[Marini et al., 2020] Marini, F., Linke, J., and Binder, H. (2020). ideal: an r/bioconductor package for interactive differential expression analysis. 21(1):565.

[Mortazavi et al., 2008] Mortazavi, A., Williams, B. A., McCue, K., Schaeffer, L., and Wold, B. (2008). Mapping and quantifying mammalian transcriptomes by RNA-seq. 5(7):621–628.

[Raudvere et al., 2019] Raudvere, U., Kolberg, L., Kuzmin, I., Arak, T., Adler, P., Peterson, H., and Vilo, J. (2019). g:profiler: a web server for functional enrichment analysis and conversions of gene lists (2019 update). 47:W191–W198.

[Robinson et al., 2010] Robinson, M. D., McCarthy, D. J., and Smyth, G. K. (2010). edgeR: a bioconductor package for differential expression analysis of digital gene expression data. 26(1):139–140.

[Schleiss et al., 2021] Schleiss, C., Carapito, R., Fornecker, L.-M., Muller, L., Paul, N., Tahar, O., Pichot, A., Tavian, M., Nicolae, A., Miguet, L., Mauvieux, L., Herbrecht, R., Cianferani, S., Freund, J.-N., Carapito, C., Maumy-Bertrand, M., Bahram, S., Bertrand, F., and Vallat, L. (2021). Temporal multiomic modeling reveals a b-cell receptor proliferative program in chronic lymphocytic leukemia. 35(5):1463–1474.

[Seelbinder et al., 2019] Seelbinder, B., Wolf, T., Priebe, S., McNamara, S., Gerber, S., Guthke, R., and Linde, J. (2019). GEO2rnaseq: An easy-to-use r pipeline for complete pre-processing of RNA-seq data.

[Sherman et al., 2022] Sherman, B. T., Hao, M., Qiu, J., Jiao, X., Baseler, M. W., Lane, H. C., Imamichi, T., and Chang, W. (2022). DAVID: a web server for functional enrichment analysis and functional annotation of gene lists (2021 update). 50:W216–W221.

[Subramanian et al., 2005] Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S., and Mesirov, J. P. (2005). Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. 102(43):15545–15550.

[Team, 2021] Team, R. C. (2021). R: a language and environment for statistical computing.

[Thomas et al., 2022] Thomas, P. D., Ebert, D., Muruganujan, A., Mushayahama, T., Albou, L., and Mi, H. (2022). Panther: Making genome-scale phylogenetics accessible to all. 31(1):8–22.

[Wagner et al., 2012] Wagner, G. P., Kin, K., and Lynch, V. J. (2012). Measurement of mRNA abundance using RNA-seq data: RPKM measure is inconsistent among samples. 131(4):281–285.

[Weger et al., 2021] Weger, B. D., Gobet, C., David, F. P. A., Atger, F., Martin, E., Phillips, N. E., Charpagne, A., Weger, M., Naef, F., and Gachon, F. (2021). Systematic analysis of differential rhythmic liver gene expression mediated by the circadian clock and feeding rhythms. 118(3):e2015803118.

[Yosef et al., 2013] Yosef, N., Shalek, A. K., Gaublomme, J. T., Jin, H., Lee, Y., Awasthi, A., Wu, C., Karwacz, K., Xiao, S., Jorgolli, M., Gennert, D., Satija, R., Shakya, A., Lu, D. Y., Trombetta, J. J., Pillai, M. R., Ratcliffe, P. J., Coleman, M. L., Bix, M., Tantin, D., Park, H., Kuchroo, V. K., and Regev, A. (2013). Dynamic regulatory network controlling TH17 cell differentiation. 496(7446):461–468.

[Zhao et al., 2020] Zhao, S., Ye, Z., and Stanton, R. (2020). Misuse of RPKM or TPM normalization when comparing across samples and sequencing protocols. 26(8):903–909.