UNIVERSITÁ DEGLI STUDI DI SALERNO

DOTTORATO IN MANAGEMENT & INFORMATION TECHNOLOGY

CURRICULUM: INFORMATION SECURITY & INNOVATION SYSTEMS

COORDINATORE: Ch.mo. Prof. Antonelli Valerio

Ciclo XVII N.S.

Novel tools for reproducible Next Generation Sequencing data analysis and integration

Relatori
Ch.mo. Prof. Tagliaferri Roberto
Ch.mo. Prof. Angelini Claudia

Candidato
Righelli Dario
Matr. 8800800010

ANNO ACCADEMICO 2017/2018

When the men on the chessboard
get up and tell you where to go
And you've just had some kind of mushroom,
and your mind is moving low
Go ask Alice, I think she'll know

Add acknowledgements here

Write your abstract here

Contents

Chapter 1

Introduction

1.1 Biological Background

1.2 Sequencing Techniques

1.2.1 RNA-Seq

Since the beginning of modern biology, gene expression is part of central dogma of molecular biology. Of course during the decades some aspects have changed, but the RiboNucleic Acid (RNA) transcripts stile have very high relevance. Nowadays, *RNA-seq* [1–4] is the most widely used technology to understand gene related regulatory mechanisms in response to stress conditions or drug treatments and progressions of several diseases [5].

Main aim of *RNA-seq* experiment is to highlight which genes are increasingly (*up-regulated*) or decreasingly (*down-regulated*) altered when comparing two or more conditions at a specific instant in time or in subsequent time points (time-course experiment), and then identify the biological mechanisms regulating such changes.

The general idea underlying the library preparation of an *RNA-seq* exper-

iment can be viewed as the conversion of long messengers RNA segments in Complementary DNA (cDNA) fragments with RNA or DeoxyriboNucleic Acid (DNA) fragmentation. To each sequence an adapter for the sequencer is added and a short read is obtained with high-throughput sequencing technology (Figure 1.2.1).

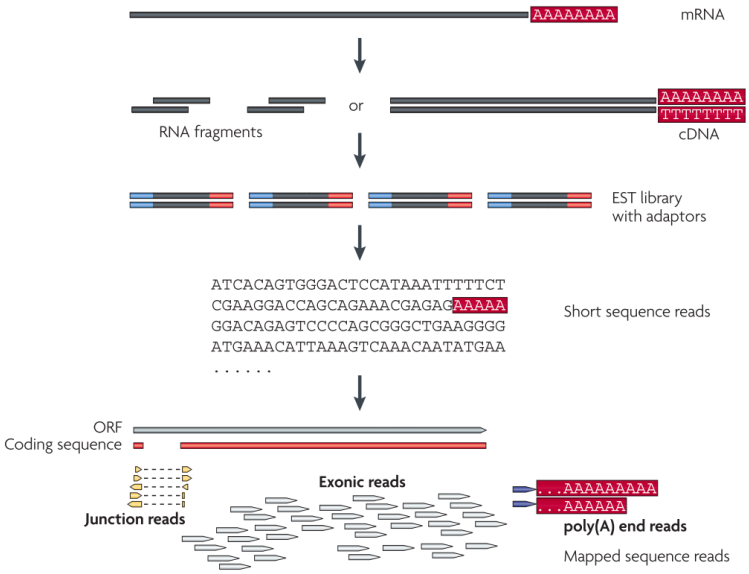


Figure 1.2.1: Representation of an RNA-seq experiment. [2]

Afterwards, the so-obtained reads need to be analyzed with several tools, depending on the particular question the researcher is interested in [6, 7].

In particular we focused on the gene expression quantification in case of multiple biological conditions, due to stress, drug treatments, disease specific, etc., where a typical analysis starts from the alignment of the reads on a specie's reference genome and the quantification of the mapped reads, producing a count matrix of the samples (on columns) and the genes related features (on the rows), typically identifiers depending on the annotation database used by the analyzer. Commonly, the count matrix needs to be filtered from low expressed features

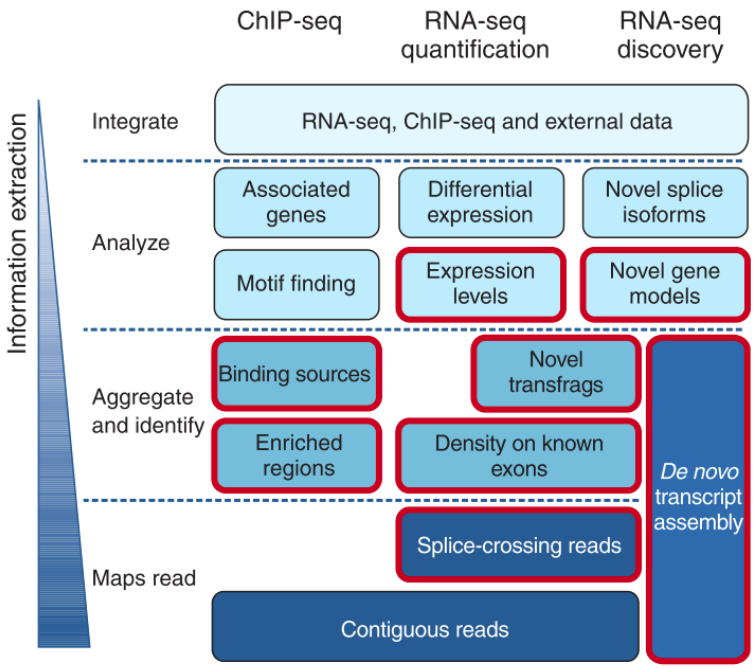


Figure 1.2.2: Representation of RNA-seq analysis complexity. [6]

and then normalized across the samples, to reduce specific bias for each sample. Then, it is possible to choose between several methods for the detection of differential expression of the features between the conditions (see section 2.2.3). Finally, the significant features can be integrated with databases of biological functionalities in order to detect the mostly influenced ones.

1.2.2 Atac-Seq

The chromatin packaging of the genome plays a fundamental role in gene regulation of eukaryotic individuals. To study this aspect of the DNA several technologies have been developed, such as *FAIRE-seq* [8], *DNase-seq* [9] and *ATAC-seq* [10], etc.

ATAC-seq among the others is having a growing interest and diffusion in the last years, because it offers comparable results to the DNASE-seq with less biological material and less library preparation time.

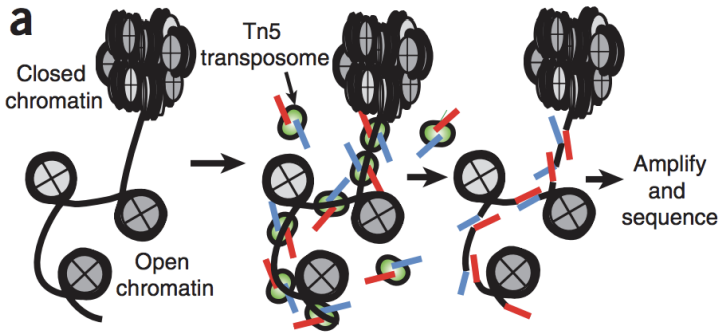


Figure 1.2.3: Representation of ATAC-seq library preparation. [10]

The library preparation adopts a hyperactive Tn5 transposase, modified with adaptors for high-throughput DNA sequencing, which is able to fragment and tag a genome simultaneously. The technology exploit the Tn5 capability of integrating itself into active regulatory elements.

After Tn5 tagmentation, the resulting segments can be amplified and sequenced, producing sequences to map on a reference genome. There is no standard analysis reached for the ATAC-seq analysis, but, inspired by the *ChIP-seq* analysis, the resulting reads, typically, are processed with tools (peak callers) for quantifying their amplification, which produces for each detected open chromatin region a feature, the peak (generally with an associated score).

Depending on the used tool, the peaks can be represented in different data structures, but their representation is given by the genomic coordinates; chromosome, starting and ending point of the region, the strand of the DNA on which the region lies, and additional attributes such as a score, a number of samples on which the regions has been detected, etc.

To obtain a first level of integration, the peaks can be annotated with other relevant features of the genome, such as the Transcription Starting Site (TSS) of

the genes, Untranslated Regions (UTR), promoters, exons, introns, etc. Then, the annotated genes can be used to enrich for GO terms or pathways, reaching a second level of integration.

1.3 Computational Aspects

Chapter 2

Time Course RNA-Seq analyzer **ticorser**

This chapter illustrates the features of Time Course RNA-Seq data Analyzer (*ticorser*), a tool developed for analyzing *RNA-seq* time course data.

2.1 Introduction

ticorser is an R package for complete and fast analysis of time dependent *RNA-seq* data, offering a vast amount of hypothesis tests setup for differential expression between two or more conditions. With aid of *edgeR*, *DESeq2* the tool offers the possiblity to setup the experiment and to obtain differential expression between the experimental biological conditions. The software is developed to assist the user to perform an entire analysis pipeline, from the quantification step, to the functional enrichment analysis, passing through the normalization, filtering and differential expression analysis.

For each step, it offers the possiblity of several interactive plots and graphics, such as KEGG-maps and hierarchical GO terms trees.

2.1.1 Time Course RNA-Seq

2.2 Methods

ticorser is a tool fully devoted to the Time Course (TC) *RNA-seq* data offering features to inspect data, to normalize them, to capture differential expression of genes at static time point and overall time points, supporting different experimental designs. It is also possible to compare the results coming from different analysis and to investigate the most influenced biological functions (i.e. Gene Ontology [11, 12] terms and Pathways).

Overall, *ticorser* offers the possibility to analyze data using different R/-Bioconductor [13] packages, to compare the results in order to choose the best combination of tools for the user specific problem. Therefore, *ticorser* implements a vast amount of exploratory and diagnostic interactive plots to explore data not just at pre-processing but also during the post-processing phase.

Starting the analysis of time course *RNA-seq* data from *BAM* files, *ticorser* enables RNA expression quantification through `featureCounts` method [14] producing a count matrix of features per samples, which enables the Differentially Expressed Genes (DEGs) detection with hypothesis statistical methods.

As figure 2.2.1 shows, this is a *design file based* tool, where a file, that describes all the samples by several variables, chaperons the counts matrix through all the analysis, simplifying, in such a way, the user interaction with all the available instruments.

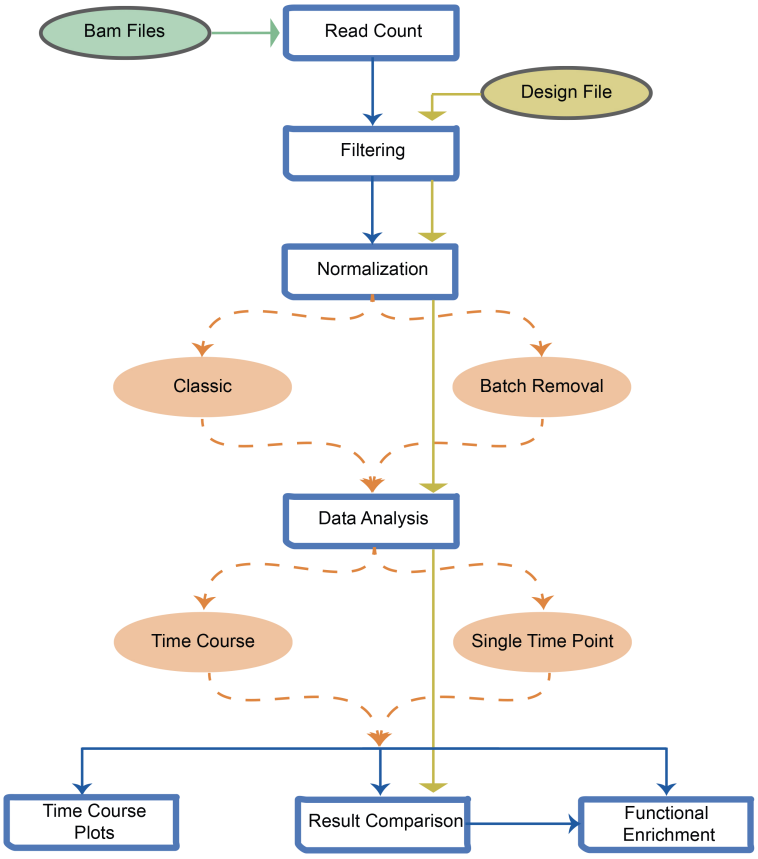


Figure 2.2.1: Main flow of ticorser R package.

Particular attention is given to the normalization phase, providing the possibility not only to use several traditional normalizations methods, but also the methodologies for batch effect removal.

In oder to inspect different approaches to analyze TC data, *ticorser* offers four different methodologies for analyzing TC *RNA-seq* data, and three different methods to analyze different biological conditions at single time point level.

2.2.1 Filtering low counts

Low expressed genes in *RNA-seq* data, always affect the detection of DEGs [15] influencing final results.

In order to address this task, *ticorser* offers multiple statistical methods for low expressed genes filtering. The `filterLowCounts` method is partially based on the `filtered.data` function of *NOISeq* R/Bioconductor package [16]. The method gives the possibility to apply four different filtering methods, the Counts per Million (CPM), the *proportion* and the *Wilcoxon* tests and an our-defined method named *quantile*.

The CPM filters out all those genes with a mean expression between the samples lesser than the `cpm` parameter threshold and, at the same time, a Coefficient of Variation (COV) higher that the `cv.percentage` parameter in all the samples.

The *proportion* test performs the homonym test on the counts, filtering out all those features with a relative expression equal to $cpm/10^6$.

The *Wilcoxon* test filters out all those genes with a median equal to 0.

Finally, the *quantile* method enables to filter out all those genes which express the counts mean between all the samples higher than the `quantile.threshold` argument (default is 99%).

2.2.2 Data Normalization

Normalization is a fundamental aspect in *RNA-seq* data analysis, especially when a comparison between different biological conditions is aimed to highlight the major differences.

For this reason *ticorser* is particularly focused on this aspect. Indeed, through the `normalizeData` function *ticorser* provides five different normalization methods without taking care of particular additional information, except for the normalization method. Thanks to the design matrix based system, the method allows to automatically subset the data only to the relevant factors the user want to normalize.

The function offers the possibility to normalize the data with *Full quantile*,

Upper quartile and *Trimmed Mean of M-values*, setting the `norm.type` parameter to *fqua*, *uqua* or *tmm*.

Moreover, it is possible to apply a batch effect removal normalization method, as described in *RUVSeq* R/Bioconductor package [17], focusin the attention on *RUVg* and *RUVs* normalization methods.

The first one, *RUVg*, can be selected by setting the `norm.type` parameter to *ruvg*, which requires a list of negative controls genes with `negative.controls` parameter. If it's not already available from previous studies, this list can be produced by executing a first *differential expression* analysis, and taking the less significative genes.

In order to facilitate this process we developed a method for creating the negative control genes list from a differential enrichment result matrix. The function `estimateNegativeControlGenesForRUV` takes as input the `de.genes` and the `counts.dataset` dataframes to extrapolate the less significative genes and to redistribute them in bins representing the mean value of the genes. By selecting one or two genes per each bin our method is able to produce a list of negative control genes which have equal distribution for each gene counts trend.

Finally, the `normalizeData` function offers the possibility to remove batch effects with *RUVs* normalization, which is more robust to the negative controls, giving better results when their estimation is approximated. This method taking care of creating the group samples and also the negative controls list, in case this one is `NULL`.

2.2.3 Differential Expression

ticorser offers four different ways for analyzing time course RNA-Seq data.

Depending on the biological question under investigation we designed three different ways of interrogate the data in a time-course experiment.

Moreover, *ticorser* offers three different ways for analyzing different biological conditions in a single time point.

To do so, we took advantage of some of the mostly used and well-performing [18] R/Bioconductor packages, *DESeq2*[19];*MASigPro*[20]; *edgeR*[21]; *NOISeq*[16].

In the following sections we firstly present the Time-Course methods and then the methods for single time point gene differentiation.

Time-Course DE Method 1 - *LRT-TC*

The first method (*LRT-TC*) uses a Likelihood Ratio Test (LRT) to compare two different models in order to extract all those DEGs that invert their expression expression between the conditions across all the time points.

Exploiting the LRT, as implemented in *DESeq2* R/Biocnductor package, we compare two different formulas. The first one defines the *full* model where we put together the timepoints, the conditions and an interaction term between these two variables, while the second one is a reduced model where the interaction term is removed:

In so doing we are able to catch all the genes inverting their expression across the conditions along the time-course experiment.

$$LRT \sim \frac{times + conditions + times : conditions}{times + conditions}$$

Time-Course DE Method 2 - *LRT-T*

The underlying idea of the second method is the same of the first one, where the difference, here, is to remove from the *reduced* formula, not only the interaction term, but also the *conditions* variable. In such a way we are able to extract all those DEGs that have different expression profiles between the conditions across all the time points.

The first formula here defines the same *full* model of the first method, while the second one is the reduced model where only the times appear:

$$LRT \sim \frac{times + conditions + times : conditions}{times}$$

Time-Course DE Method 3 - *LRT_NOInteraction*

Using always the *DESeq2* LRT we defined a third method for the identification of DEGs that have different expression between the conditions across all the

time points, but that maintain the same profile in both conditions.

Here the *full* model defines the time points and the conditions variables without taking into account the interaction term, while the second the *reduced* model presents only the time point variable:

$$LRT \sim \frac{times + conditions}{times}$$

Single DE Methods

To account for fixed time point experiment we implemented functionalities for helping also the exporation of this aspect, using three different methodologies.

By using the `differentiateConditions` function, it is possible to choose between the *edgeR*, *DESeq2*, *NOISeq* and *NOISeqBio*.

In case of *edgeR* we decided to use the *Quasi-Likelihood* method for the differential expression. While when using *DESeq2* for this specific case we choose the *Wald* test, as suggested by the authors.

The *NOISeq* package offers the possibility to discriminate between *biological* and *technical* replicates, computing a posterior probability in both cases.

2.2.4 Data Visualization

While analyzing RNA-Seq data, it is almost mandatory exploring data during each step of the analysis, in order to understand which is the best method to apply for at each analysis step and to be more confident with produced results.

At this scope, we equipped *ticorser* of several useful graphics and plots useful at each analysis step.

Each of them, except when otherwise declared, enables to convert the plot in an interactive *HTML* plot, useful to inspect additional attributes with the mouse pointer.

Exploration Plots on Counts

For exploring counts data we implemented two plots, the *Boxplot* and the Principal Component Analysis (PCA).

The *boxplot* is a graphical representation of the distribution of the samples, in our case we decided to plot a *boxplot* for each sample and to colour them accordingly to the time-point they belongs to (figure 2.2.2).

It is a box divided in two parts, with an outgoing segment from each side. The horizontal upper and lower parts of the box represent the first and third quartile, while the middle horizontal rule represents the median. The upper and lower part of the segments represent the minimum and maximum values of the sample distribution.

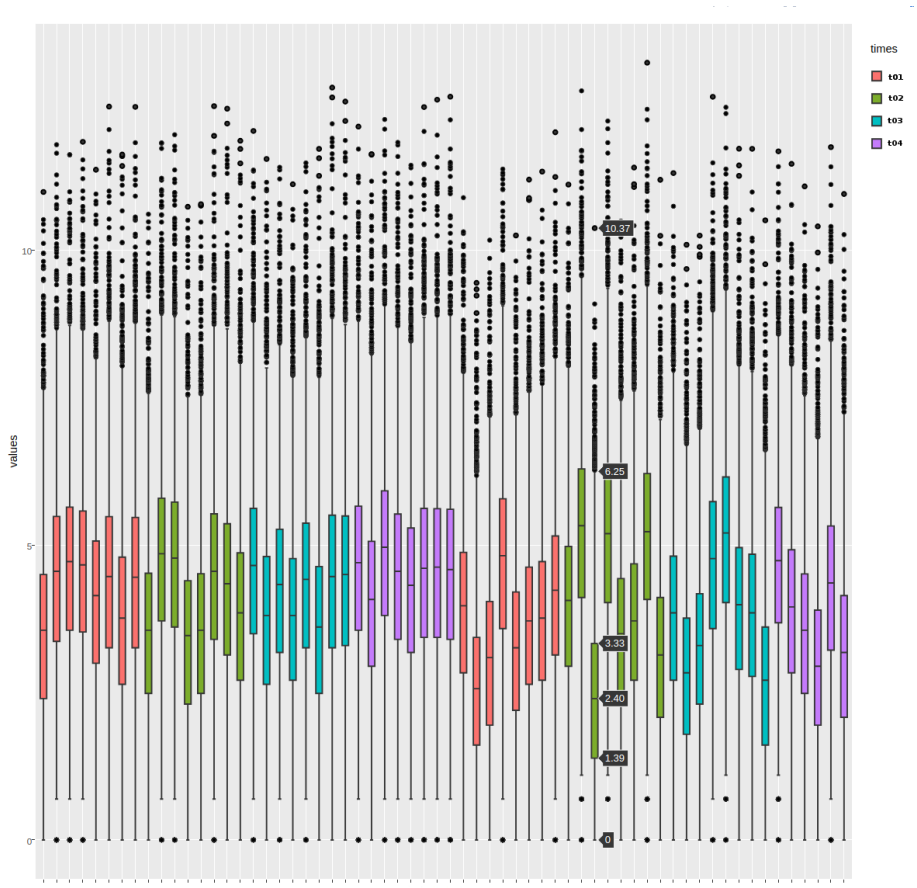


Figure 2.2.2: An example of interactive boxplot made with *ticorser* package. Each boxplot represents a specific sample, while each colour represents a specific time point. When passing the mouse over a boxplot it shows additional information about its quartiles.

Moreover, the `plotBoxplotPlotly` function is based on the design matrix, describing the data counts, which gives the possibility to select the column to use for colouring the samples thanks to the `colorColname` parameter.

Due to the very high dimensionality of RNA-Seq data, it is widely considered

common sense to apply PCA dimensionality reduction technique to visualize the data, limiting their representation to 2 or 3 dimensions. There are several packages allowing to apply a PCA transformation, but we choose to implement it in the `PlotPCAPlotlyFunction` function, by using the `prcomp` from the *stats* package.

The user just needs to give the `counts.data.frame` and the `design.data.frame` by specifying the column name where to find the samples groups. In such a way, the function will compute and plot the PCA, coloring the samples according to the groups identified by the chosen column. Moreover, by setting to `TRUE` the `ellipse.flag` argument, the function will show the ellipses surrounding each sample group, each describing the variance of the groups.

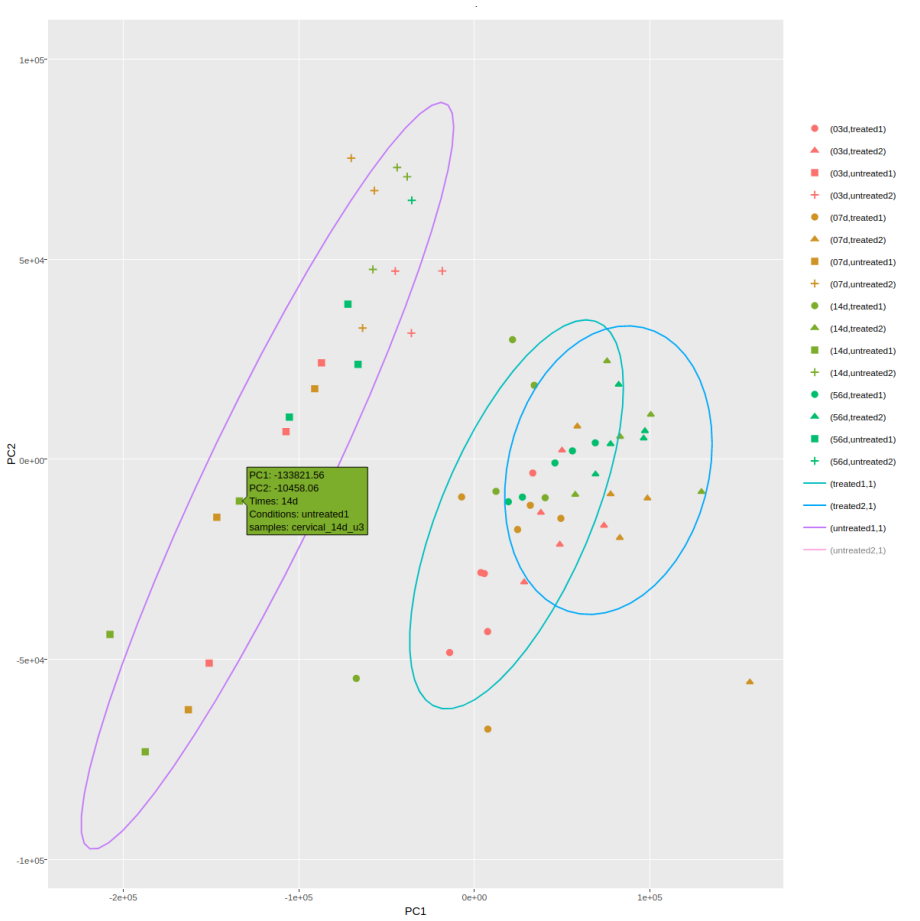


Figure 2.2.3: An example of interactive PCA made with *ticorser* package. Each dot represents a specific sample, while each colour represents a time point, each simbol represents a biological condition group. When passing the mouse over a dot it shows additional information about the selected sample, while from the legend it's possible to show/hide groups or ellipses.

DE Results Plots

In order to inspect the results producted by DE methods, we implemented two kind of very common plots, the *VolcanoPlot* and the *MAPlot*.

Both our implementated methods take as input a DE results data frame automatically recognizing which method produced it. Moreover, it gives the possiblity to add a list of positive control genes, in order to annotate them on the volcano plot with a third colour (figure 2.2.4).

The volcano plot puts in relation the $log_2(FC)$ with $log_{10}(p-value)$ in order to highlight the significant changes inside the data experiment.

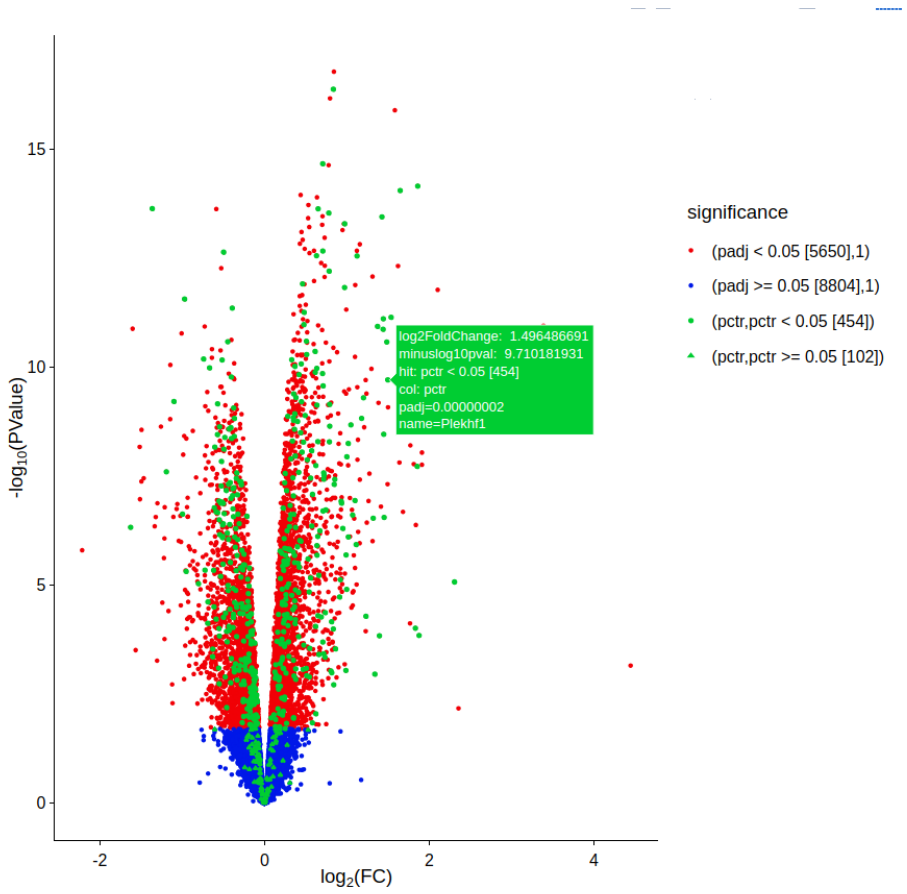


Figure 2.2.4: An example of interactive volcano plot made with *ticorser* package. Each dot represents a gene, while blue and red colours highlights the significance of the genes. In green there are those genes coming from the positive control list. When passing the mouse over a dot it shows additional information about the selected gene.

The MA-Plot puts in relation two quantities useful to understand the differences between the measurements in two conditions. On the x-axis there is represented the $log_2(FC)$ where FC is the fold change computed as the ratio of

the treatment on the reference. It is not mandatory to have a DE results data-frame to plot an MA-plot, but it's pretty useful to have it in order to understand the distribution of the significant genes (figure 2.2.5).

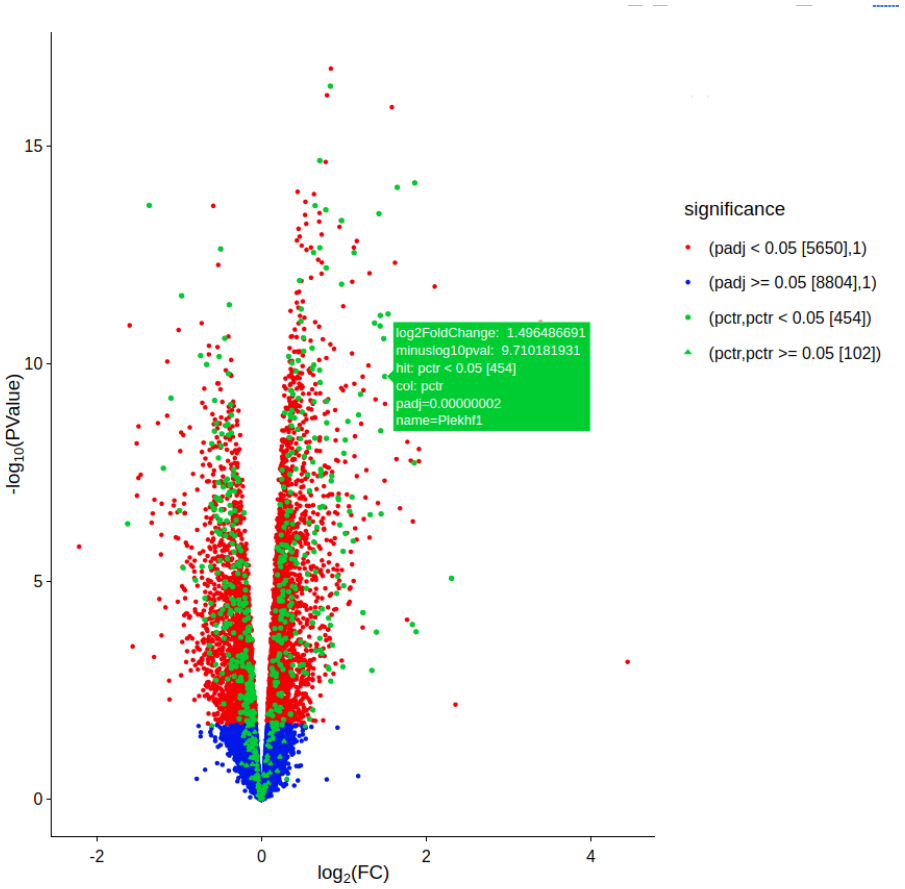


Figure 2.2.5: An example of interactive MA-plot made with *ticorser* package. Each dot represents a gene, while blue and red colours highlights the significance of the genes. When passing the mouse over a dot it shows additional information about the selected gene.

Gene Profiles plot

For highlighting the gene expression of a gene across multiple time points and different conditions, we implemented the function `plotGeneProfile`, which takes as input a count matrix, its linked design matrix and a gene name, . In such a way the function is able to plot the profile of a gene showing its trend throught all time points, and between the different conditions (figure 2.4.6).

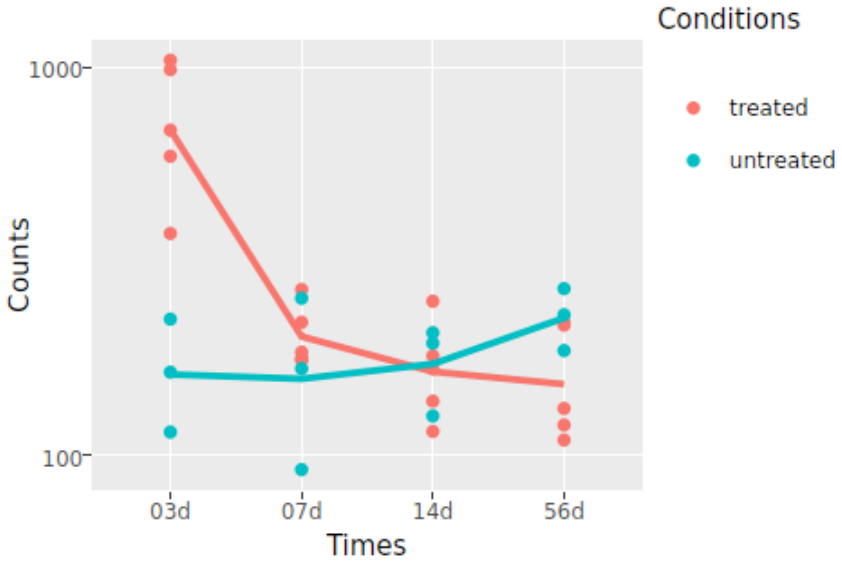


Figure 2.2.6: An example of interactive gene profile made with *ticorser* package. Each dot represents the counts value of the gene in a sample. Colours identifies the conditions of the samples. The lines represent the gene trend over all time points.

keggmap

ticorser offers the possibility to plot *keggmaps*[22] taking into account the $log_2(FC)$ of the genes involved in the graphical representation, thought all the timepoints.

Indeed, using the `plotKeggmap` function it enables to plot a *keggmap*, using as input the counts and the design matrices, computing the $log_2(FC)$ at each time point, and showing it in the gene box inside the kegg map.

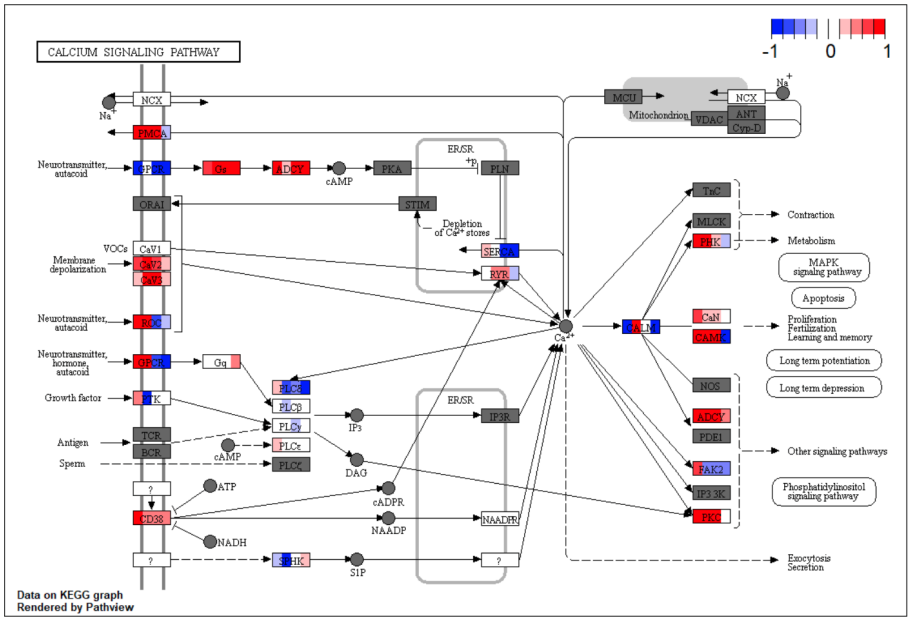


Figure 2.2.7: add description

2.3 Additional Features

ticorser offers multiple additional features to help the user during the differential enrichment analysis of time-course RNA-Seq data.

Gene quantification

A fundamental aspect during *RNA-Seq* analysis is the quantification of the features (genes). To account for this, *ticorser* give the possibility to quantify the

gene expression, starting from samples mapping files, using the `countBamFilesFeatureCounts` function with the aim to guide and facilitate this operation to the user.

The feature is based on the `featureCounts` method available through the *Rsubread* R/Bioconductor package [23], hard coding some parameters as `useMetafeatures` to `TRUE` and `allowMultiOverlap` to `FALSE`. The user can give as input a list of *BAM* files and a Gene Transfer Format (GTF) file, choosing the `gtf.attr.type` and `gtf.feat.type` to quantify the samples expression on its needs.

Results Comparison

Usually during a differential expression analysis it is pretty common to have the need to compare multiple DEGs results list. In order to facilitate this need we developed three different functions for venn diagrams plots, `Venn2de`, `Venn3de` and `Venn4de`. During this process it is often required not just to show the graphical plot, but it's most important to show the gene lists resulting from the intersections and disjunctions of the venns. To afford this aim these functions take as input the gene lists to compare and automatically produces as output lists of files within the resulting lists of all the areas of the produced venns.

Gene Identifiers Conversion

The *ticorser* package exports multiple functions for the gene names manipulation. Indeed, it is very common the need to convert DEGs list from a specific identifier to another. We developed `convertGenesViaMouseDb` and `convertGenesViaBiomart` which convert a DEGs list using the *org.Mm.eg.db* [24] for *Mouse* and using the *biomaRt* R/Bioconductor packages for *Human, Mouse* and *Rat*. Additionally, it's possible to easily attach the resulting list to a *dataframe*, by using the `attachGeneColumnToDf` function, which takes care of adding a new column within the mapped identifiers in the right places of the original *dataframe*.

Input/Output File Handling

To speed up the reading and writing of input/output files, *ticorser* offers two main functions, `readDataFrameFromTSV` and `writeDataFrameAsTSV`.

In the first case there is only one mandatory parameter, the `file.name.path`, even if gives the possibility to change the classical parameters as `row.names.col , headed.flag, sep, quote.char`.

While in the second case the function requires the `data.frame.to.save` and the `file.name.path` where to store the object. Also in this case it is possible to set the classical parameters as `col.names` and `row.names`.

Moreover, we implemented a method for creating a folder path in a recursive way. The `updateFolderPath` method takes as input a starting path and a list of strings. It is useful when using a recursive function call or an automated nested function call process. Of course if a user need to create a single path by itself, he/she can simply create it using the `dir.create` R base function.

2.4 Case Study

For testing our package we selected a not yet published, but very complex dataset for traumatic Spinal Cord Injury (SCI), that is a neurological condition occurring mainly at the thoracic and cervical levels. The dataset is constituted of 62 samples of *RNA-seq*, divided in groups of two different tissues at four different time points, with treatments and controls for each time point. Because the dataset is not yet accessible, the genes and the samples have been masked during the analysis and no further details will be provided, but we only use it as illustrative example.

Features quantification

ticorser gives the possibility to quantify the gene expression by using the `featureCounts` method of the `rsubread` R/Bioconductor package, by using the `countBamFiltesFeatureCounts` method with the path of the *BAM* files and a GTF[1] file within the desired annotation features.

It's really important the choice of the GTF file, in terms of version and release, because it affects the further analysis, that's why we suggest to always use

[1] https://www.ensembl.org/info/website/upload/gff.html

the GTF associated to latest version of the genome for the under investigation specie [2].

After the gene quantification, the method produces a count matrix with the features (genes) on the rows and the samples on the columns. Each cell of the matrix is an integer value indicating the amount of reads quantified for the feature on the row in the column sample.

The design matrix

From this point afterwards, *ticorser* requires a design file illustrating their characteristics of each sample, in order to speed up the computations and the interactions with the user. In particular, the design matrix must have a column specifying the sample names, which have to be equal to the column names in the count matrix. Table 2.1 shows an example of a typical design matrix useful to work with *ticorser* package.

[2]Two main resources for genome download are https://www.ensembl.org/index.html and https://www.ncbi.nlm.nih.gov/grc

rownames	Times	Conditions	Tissue
s01_t01_t1	01h	treated1	tissue1
s02_t01_t1	01h	treated1	tissue1
s03_t01_t1	01h	treated1	tissue1
s01_t01_u1	01h	untreated1	tissue1
s02_t01_u1	01h	untreated1	tissue1
s03_t01_u1	01h	untreated1	tissue1
s01_t02_t1	02h	treated1	tissue1
s02_t02_t1	02h	treated1	tissue1
s03_t02_t1	02h	treated1	tissue1
s01_t02_u1	02h	untreated1	tissue1
s02_t02_u1	02h	untreated1	tissue1
s03_t02_u1	02h	untreated1	tissue1
s01_t01_t2	01h	treated2	tissue2
s02_t01_t2	01h	treated2	tissue2
s03_t01_t2	01h	treated2	tissue2
s01_t01_u2	01h	untreated2	tissue2
s02_t01_u2	01h	untreated2	tissue2
s03_t01_u2	01h	untreated2	tissue2
s01_t02_t2	02h	treated2	tissue2
s02_t02_t2	02h	treated2	tissue2
s03_t02_t2	02h	treated2	tissue2
s01_t02_u2	02h	untreated2	tissue2
s02_t02_u2	02h	untreated2	tissue2
s03_t02_u2	02h	untreated2	tissue2

Table 2.1: An example of design matrix required for the right working of *ticorser* package.

Filtering & Normalization

A so obtained count matrix could reflect the effects of one or more bias due to experimental passages during the library preparation or to different sequencing batches. **(need reference)** In order to account for this, is a good practice to normalize data, a step which affects the DEGs detection[25–27]. Before normalizing, it might also be useful to filter out low expressed features, since less significant and may lead to biased results introducing unwanted noise in the

analysis.

Figure 2.4.1 shows the effects of the filtering step on the row counts (in the upper left corner). What emerges from this comparison is that the low expressed features are in high amount in raw data, while it is quite un-important the method used for filtering them.

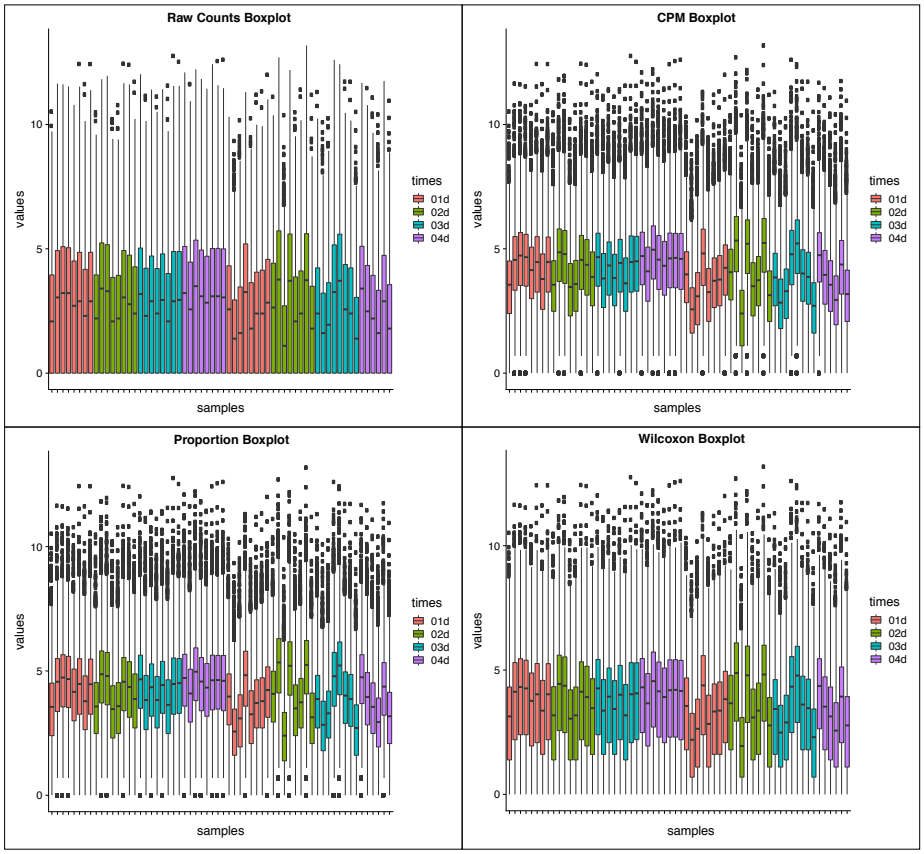


Figure 2.4.1: A comparison of how the filtering methods available in *ticorser* affect the data. The first panel shows the row counts with low expressed genes, while the other boxes shows the filtering effect on the samples.

Moreover, looking at the *Wilcoxon* test, it is more conservative, because it preserves 17039 features, respect to the *Proportion* test and CPM, that preserves, respectively, 14122 and 14171 features. We decided to use the count matrix filtered with *Proportion* test and default parameter.

In order to improve the results for the differential expression analysis, it is crucial to correct for between-sample distributional differences in read counts. To compare the differences of normalized data we decided to use not only the boxplots for the samples, as shown in figure 2.4.2, but also the PCA representation, enabling us to understand which normalization better remove biases from the samples by clusterizing the samples of the same group (figure 2.4.3).

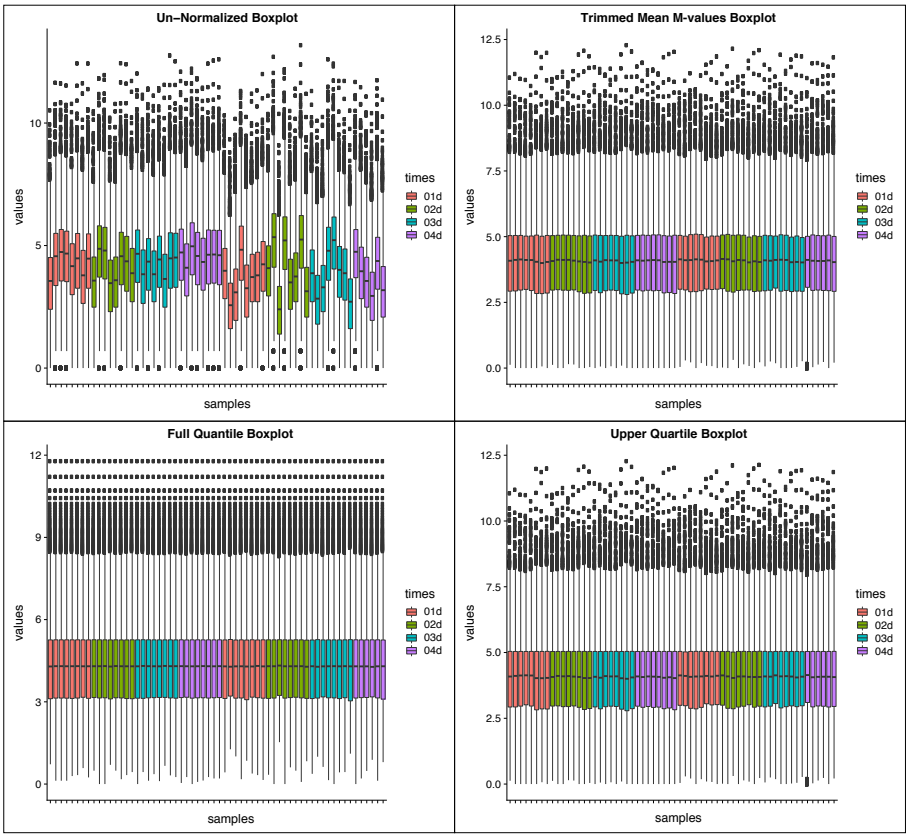


Figure 2.4.2: A comparison of how the filtering methods available in *ticorser* affect the data. The first panel shows the row counts with low expressed genes, while the other boxes shows the filtering effect on the samples.

Figure 2.4.2 perfectly shows the different effects of different normalization types on the count data. In fact we can see that while *Upper quartile* aligns all the samples on third percentile leaving the medians not aligned, the *Full quantile* totally aligns the samples, not only on the third percentile, but also the medians as such as the all the upper distributed outliers. At the same time,

the *TMM* aligns the samples, but in a not so stringent manner, well interpreting and leaving the variability of each sample.

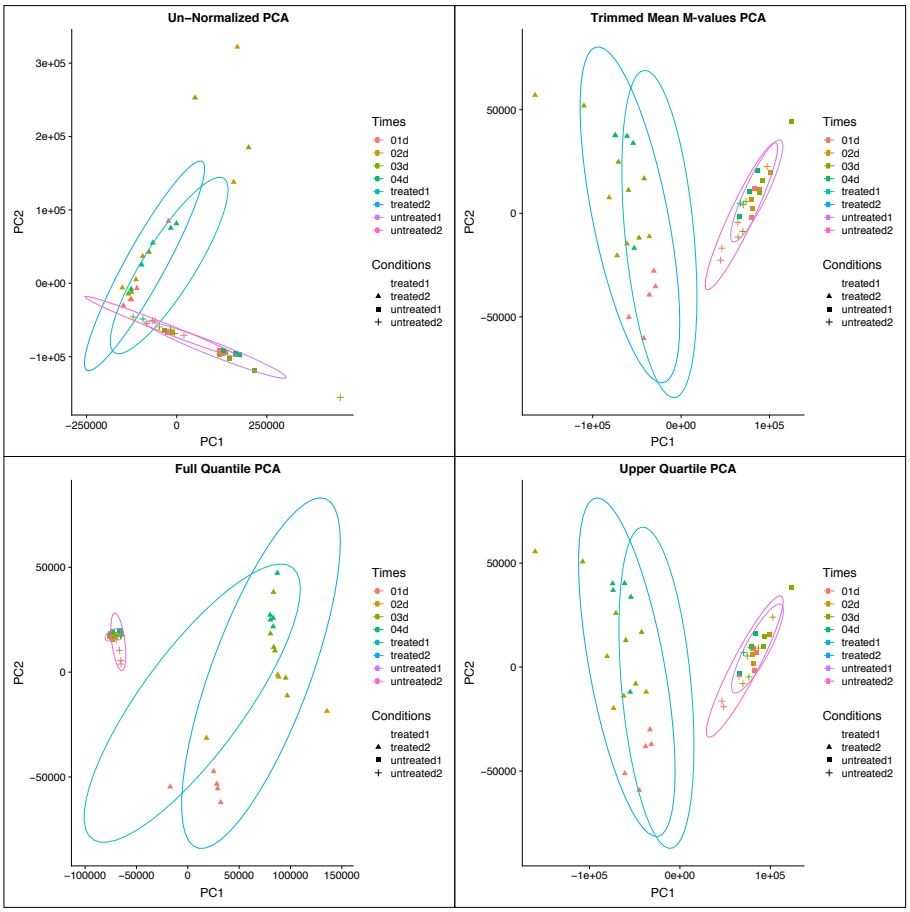


Figure 2.4.3: A comparison of how the normalizations methods available in *ticorser* affect the data. The first panel shows the clustering of row samples, while the other boxes shows the normalization effect on the samples.

As already mentioned a boxplot data inspection could not be enough to discriminate between the normalization methods. Indeed, looking at *PCAs* of

normalized counts (figure 2.4.3 the *TMM* and the *Upper quartile* normalizations are able to well discriminate the treated and the untreated samples, but still preserving some variability inside each group. While the *Full quantile*, even if it is able to help discriminating the treated and untreated groups, it smashes too much the variability inside the untreated samples.

Differential Expression

Once the data are well normalized, in order to be able to well discriminate between the groups, simply by changing our design matrix with only the interested samples, we can focus on the differential expression step. Here we focus only on half of the total dataset, checking differences between the treated and untreated samples across experimental time points.

For detecting DEGs across time points taking into account the conditions we designed *ticorser* with several methods (see section 2.2.3 for further details). Depending on the biological question under investigation, using the `ApplyDeSeq2` function and using the count matrix with the design matrix, the package automatically detects the samples to discriminate for the differential expression. Moreover, depending on the method selected, it is able to detect different types of genes between the samples. When selecting `DeSeqTime_TC` method, it detects all the genes which changes their expression across all time points between two conditions in the `Conditions` column of the design matrix, while using `DeSeqTime_T`, it recognizes all the genes which have different expression between the conditions across all the time points. Furthermore, using `DeSeqTime_NoInteraction`, the method is able to detect all those genes demonstrating an oscillating behaviour across the time points in both the conditions.

Additionally, for each of the previous described methods we produce an additional output, obtained with a *Wald Test*, in order to detect all those genes which express differential expression between the two conditions.

For each method we produce a list of two lists, within the results for the LRT and for the *LRT_ Wald*. Each of this already divided for all differential expressed gene, only UP genes, only DOWN genes and the results table with all the genes and their statistics.

Table 2.2 illustrates differences in catching DEGs between the three different methods on the same dataset, highlighting the total number of genes reported, with UP and DOWN regulated. It is relevant to see that, when applying the *Wald Test* on the results obtained by LRT the DEGs for this test are much higher in number.

	LRT			Wald on LRT		
	Total	UP	DOWN	Total	UP	DOWN
TC	825	454	371	7066	3606	3460
T	8812	4672	4140	7066	3606	3460
No-Int	9560	4893	4667	9567	4900	4667

Table 2.2

It is also possible to inspect the *DE* results by plotting a *Volcano Plot* or an MA plot, as figure 2.4.4 shows.

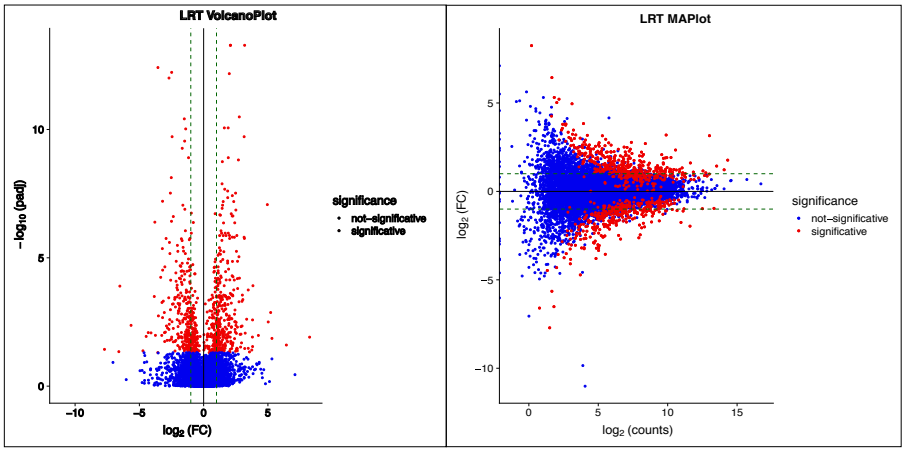


Figure 2.4.4

In order to better compare the results coming from different analysis and methodologies approaches we can use one of the VENN diagrams available in *ticorser*, which, while intersecting the results, saves all the gene lists coming

from the intersections and also from exclusions. Moreover, by setting the `enrich.lists.flag` to `TRUE`, the methods automatically starts functional enrichment for both pathways (on Reactome and KEGG databases) and Gene Ontology (on Biological Process, Cellular Components and Molecular Function), storing all the results in the `output.folder` specified (we remind to next section for further details on Functional Enrichment Analysis).

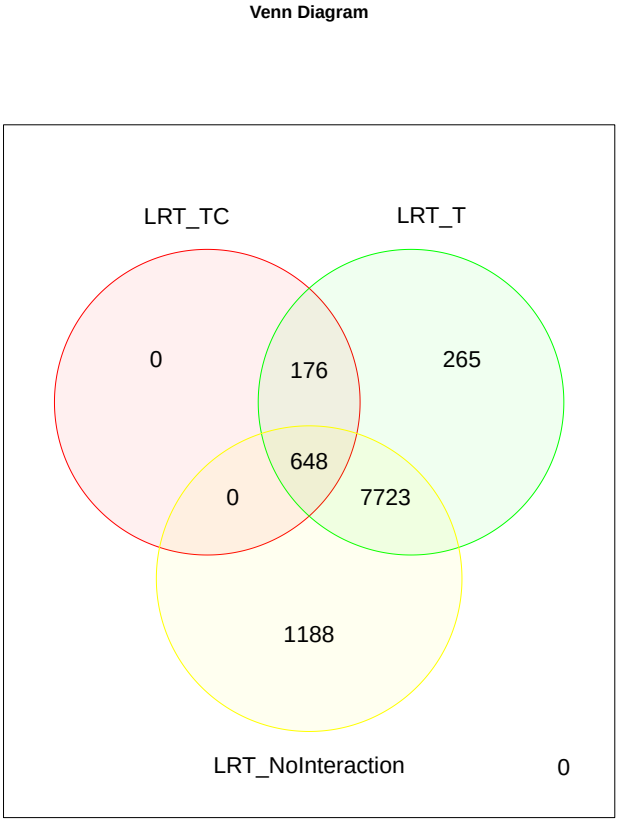


Figure 2.4.5

Depending on the biological question under investigation, it can be useful to identify the unique genes for each method used in the time course differential expresison analysis, or the genes coming from their intersections.

To look at the gene expression, *ticorser* has a specific function to explore the trend of a specific gene across the time points between the two conditions. By the usage of the `PlotCountsAlongTimes` and by specifying the name of a gene that is present in the count matrix, we are able to explore its behaviour.

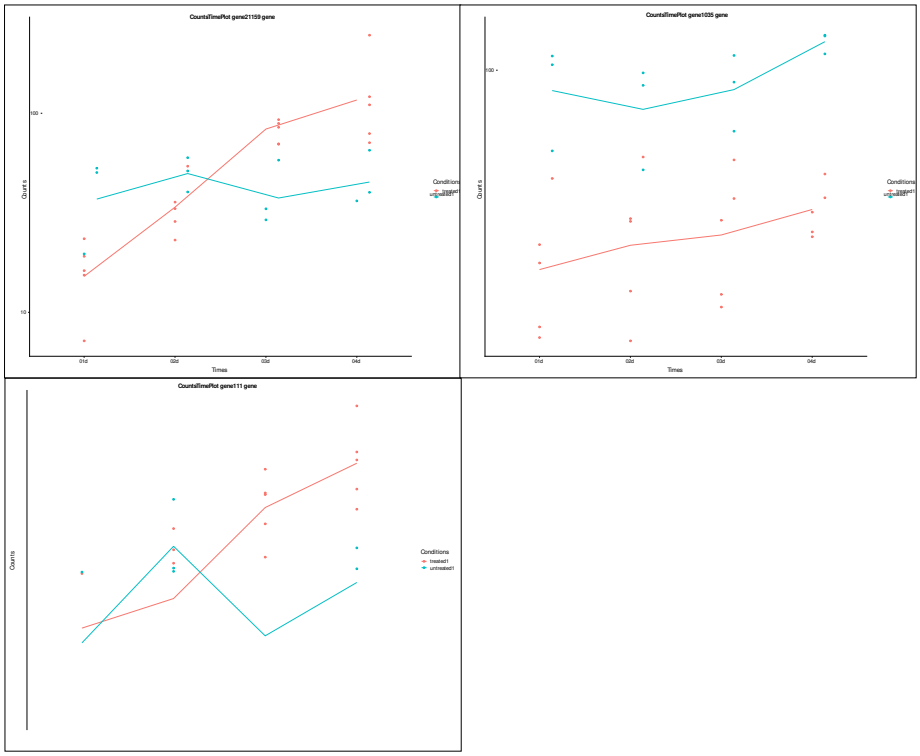


Figure 2.4.6

For example looking at figure 2.4.6 we identified three different genes, one for each DE method, that show different trends over time points in conditions.

The first one, is a gene detected by the `DeSeqTime_TC` method, showing a complete inversion of the expression between the conditions during the time course experiment, otherwhise, the second gene, detected with `DeSeqTime_T` method, in the right upper corner shows a difference between the conditions that remains uninvariate across the experiment. Finally, using the `DeSeqTime_NoInteraction` method, we detected a gene with a s̈trange b̈ehaviour, which is very oscillating across the time points and the conditions.

Once inspected the results for the time course experiment analysis, an investigator could be interested in exploring the data on singular time points, here we report an analysis example on the first time point.

For the singular time point analysis we can choose between three different methodologies, two present in the *NOISeq* package (accessible through the `ApplyNoiseq` function), while the third one is the *Wald test* present in the *DESeq2* package. It is a good practice to use more than one method, while working with transcriptomic data in order to be more confident on the final results. Table 2.3 illustrates differences between the methods used for DEGs detection.

	Genes		
	Total	UP	DOWN
DESeq2	5239	2684	2555
NOISeq	24	24	0
NOISeqBio	6315	3053	3262

Table 2.3

Figure 2.4.7 shows differences in the detection of DEGs using the three different methods. In particular, `NOISeqBio` is really useful when working with biological replicates (like in this case), indeed it is able to detect much more DEGs than the other methods. Indeed, when using `NOISeq` on the same data, the DEGs are only 24, while *DESeq2* is able to detect quite the same amount of genes.

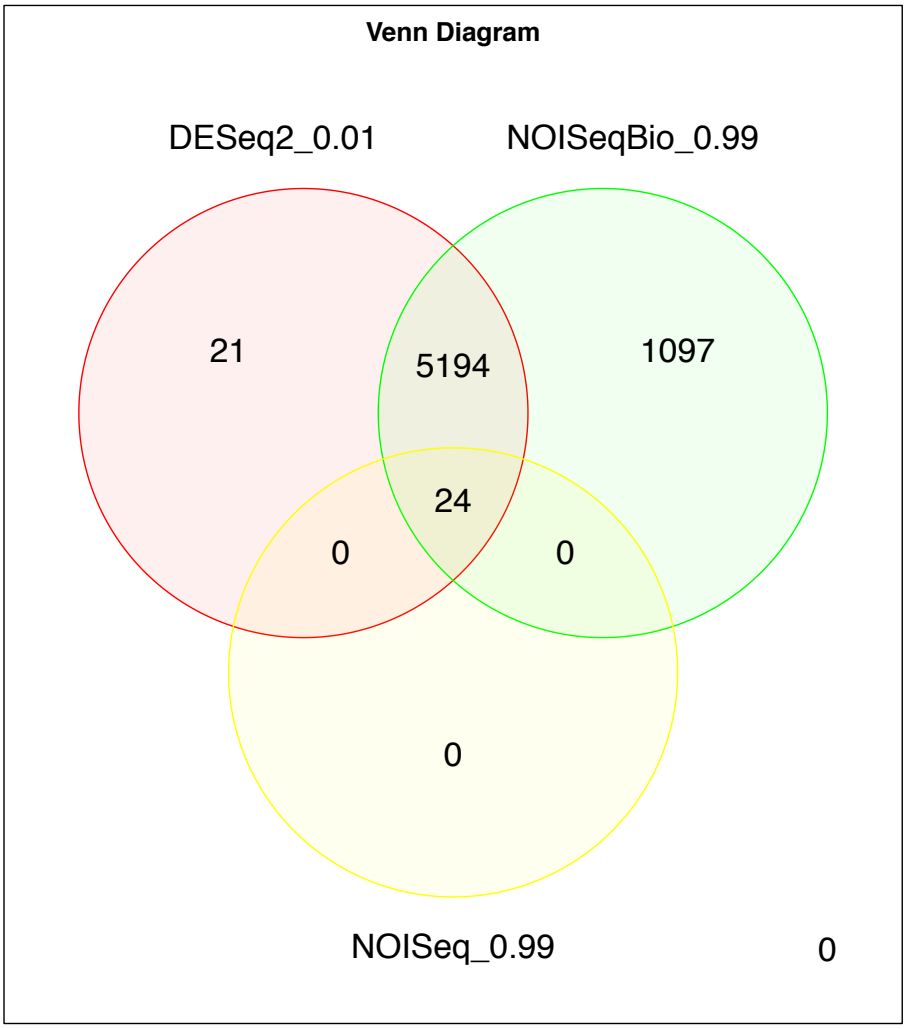


Figure 2.4.7

It is also possible to use a singular function, `PerformDEAnalysis`, for applying the preferred method in the differential expression phase. The tool automatic-

ally stores the results in the `output.folder` creating an articulate, but intuitive, folder tree with all the results and the plots (Volcano and MA Plots), and performing functional enrichment analysis on the computed results, simply by setting the `enrich.results.flag` to `TRUE`.

Functional Enrichment Analysis

For the Functional Analysis *ticorser* has a set of functions helping to perform it with *GProfiler* and *ClusterProfiler* tools.

In order to perform pathway analysis with *GProfiler* we used `enrichPathwayGProfiler` function, twice, one for the *Reactome* database and one for the *KEGG* database enrichment. Then, we performed the same analysis with *ClusterProfiler* by using the `enrichKEGGFunction`, which performs only on KEGG database, but is able to produce a graphical representation of the network of the pathways. (see figure **??**)

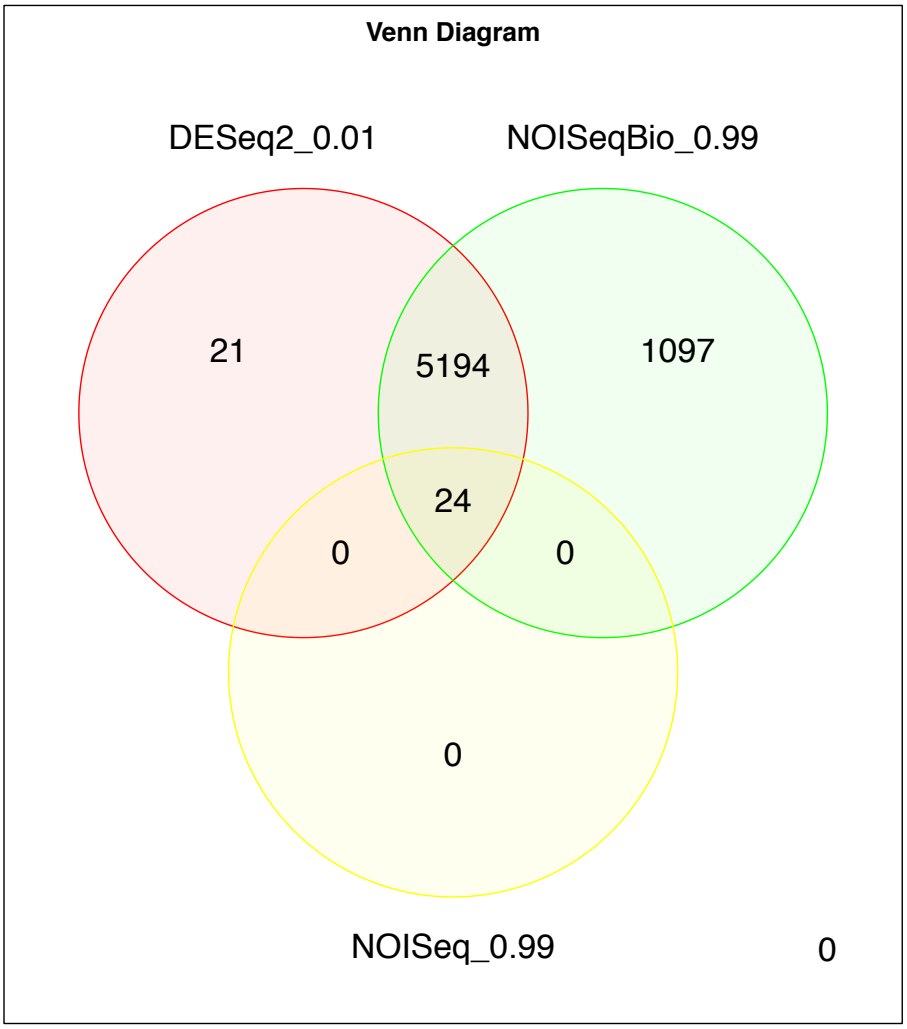


Figure 2.4.8: CHANGE THE FIGURE WITH PATHWAY NETWORK

On the other hand, to perform a Gene Ontology functional enrichment analysis we use `enrichGOGProfiler` three times, one for each class of the `ontology`,

`CC` for Cellular Components, `BP` for Biological Processes and `MF` for Molecular Functions. Analogously, we use `enrichGOFunction` for performing the same analysis with *ClusterProfiler*, which produces not only the table of the results, but also a tree for the Gene Ontology terms significative in our dataset. (see figure **ADD FIGURE**)

Finally, if we have identified a specific KEGG pathway which is of our interest, we can plot a *KEGG map* representation with `PlotKeggMapTimeCoursePathview` , which takes as input a dataframe of expression values of the relevant genes for the pathway interested. In such a way a keggmap with the expression values for each time point for those genes will be visualized.

2.5 Conclusions and Future Works

Chapter 3

Differential Enriched Scan 2 DEScan2

Epigenetics, as shown in the introduction (cite), is a pretty wide and complex field, and the sequencing technology to adopt depends on the biological question under investigation.

Some studies [28, 29] have demonstrated the importance of genome-wide chromatin accessibility of a broad spectrum of chromatin phenomena activation using sequencing techniques as *ATAC-seq*, *Sono-seq*, etc. Even if there are some methods for the analysis of these omic data types, there still is lack of them, in particular for an emerging omic as *ATAC-seq*.

To address this need, we decided to create a useful instrument for analysing chromatine regions accessibility data (such as *ATAC-seq*, *Sono-seq*). Very often the biological questions, to be answered, as for *RNA-seq*, need the comparison of two or more different biological conditions. Starting from a set of already published [28] scripts, we designed Differential Enriched Scan 2 (*DEScan2*), a software for the analysis of chromatin accession sequencing data.

In this chapter we firstly illustrate the developed methodologies and then, with a case study, we will show the obtained results as an application.

3.1 Introduction

DEScan2 is an R [30] tool developed for detecting open chromatin regions signal in order to facilitate the differential enrichment of genomic regions between two or more biological conditions.

The package has been implemented using Bioconductor [13] data structures and methods, and it is available through the Bioconductor repository since version 3.7.

The tool is organized in three main steps. A peak caller, which is a standard moving scan window that compares the reads coverage signal within a sliding window to the signal in a larger region outside the window. It uses a Maximum Likelihood Estimator of a Poisson Distribution, providing a final score for each detected peak.

The filtering and alignment steps are aimed to determine if a peak is a "true peak" on the basis of its replicability in other samples. These steps are grouped in a single procedure and are based on a double user-defined threshold, one on the peaks's scores and one on the number of samples.

The third step produces a count matrix where each column represents a sample and each row a peak. The value of each cell represents the number of reads for the peak in the sample.

The produced count matrix, as illustrated in figure 3.2.1, is useful both for doing differential enrichment between multiple conditions and for integrating the epigenomic data with other -omic data types.

3.2 Methods

The package is organized in three main steps, the peak caller in section 3.2.1, the filtering and alignment of the peaks in section 3.2.2 and the peak counting described in section 3.2.3.

Furthermore, it offers some additional features as described in 3.2.4.

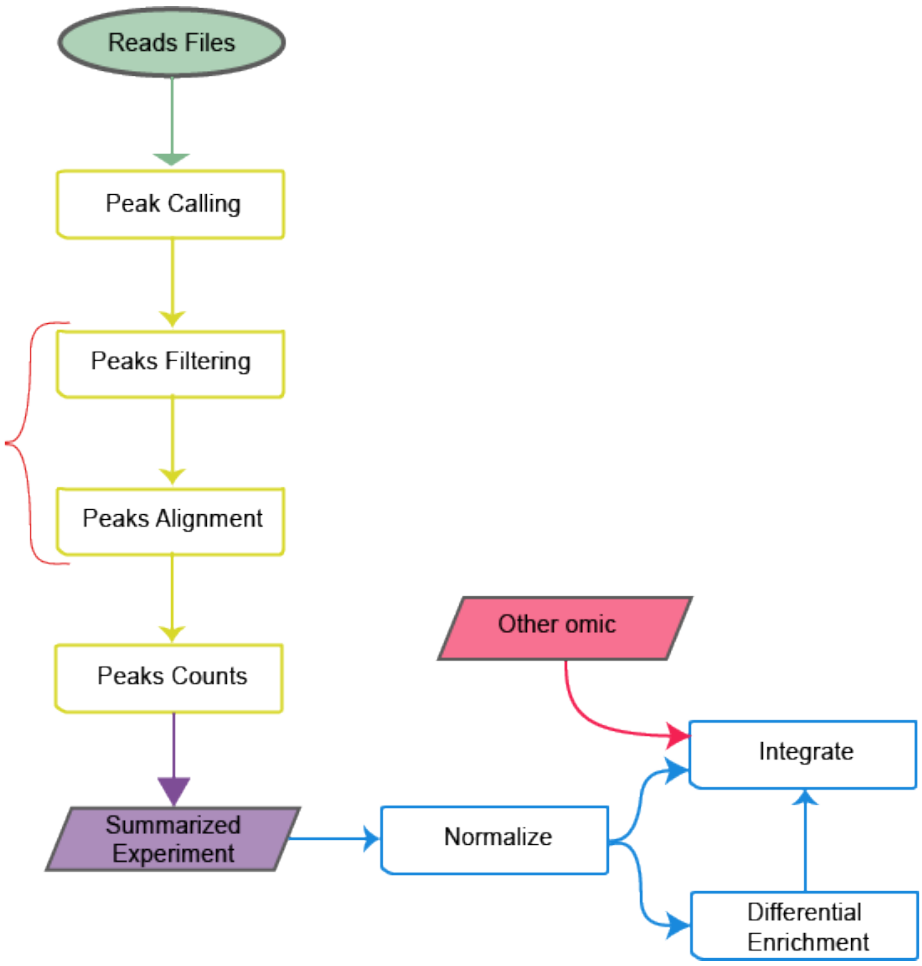


Figure 3.2.1: A differential enrichement flow representation. *DEScan2* steps are highlighed in yellow.

3.2.1 Peak Caller

The Peak Caller (defined by the `findPeaks` function) takes as input a set of alignment files (BAM [31] or BED format) with the code identifier of the refer-

ence genome (i.e. *mm10* for Mus Musculus version 10) and several additional parameters, useful for the peak detection setup.

The alignment data are stored as and object of class *GenomicRangesList* [32], where each element represents a file. In order to facilitate the parallelization of the computations over the chromosomes, the list is re-arranged as a chromosome list of *GenomicRangesList*, where each element represents the file containing just the *GenomicRanges* of the specific chromosome (see section 3.2.4 for a detailed description of this procedure).

For each element of this data structure the algorithm firstly divides each chromosome in bins of `binSize` parameter length (the default value is 50bp) and then computes the reads coverage of the bins with moving scan windows, spanning from `minWin` to `maxWin` parameters of `binSize` interval.

In order to be able to catch narrow and broad peaks the algorithm computes the coverage also using windows of two different lengths, that can be defined with `minCompWinWidth` and `maxCompWinWidth` (defaults values are 5000bp and 10000bp) parameters, computing a matrix of n bins and p windows.

The coverage matrix is useful to merge contigous regions and to compute a score for each of them, applying a Maximum Likelihood Extimator (MLE), assuming a Poisson distribution of the coverage across the windows.

Formalizing: assuming that each window is distributed as a Poisson random variable, we assume to observe the n coverages as an IID sequence X_n. Thus, the probability mass function is described as:

$$p(x_i) = \frac{\lambda^{x_i}}{x_i!} exp(-\lambda)$$

Where the integer nature of the data support the Poisson distribution as the set of non-negative integer number and where λ is the Poisson parameter to estimate with a MLE, described as the estimator:

$$\hat{\lambda}_n = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Which corresponds to the sample mean of the n observations in the sample.

Additionally, on user request, the function provides as output, for each alignment file, a Tab Separated Value (tsv) file within the regions coordinates and the score of the detected peaks.

3.2.2 Peak Filtering and Alignment

In order to filter out false positives peaks, we designed a method (defined in the `finalRegions` function) which firstly filters out low score regions and then aligns the resulting regions between samples, using two different thresholds. One on the peaks's score and one on the number of samples.

The filtering step is designed to take as input a list of peaks as *GenomicRangesList*, where each element represents a file. This is the data structure produced by the peak caller, but, we also developed a method to load peaks produced by other software like MACS [33], as described in section 3.2.4.

Firstly, using the threshold on the peaks's score (defined by the `zThreshold` parameter), the method filters out the peaks with a score lower than the user-defined threshold value.

Then, for aligning the peaks between the samples, it extends a 200bp window in both directions of remaining regions, computing the overlaps using the `findOverlapsOfPeaks` method (using the `connectedPeaks` parameter set as `merge`), as defined in the *ChIPpeakAnno* [34] R/Bioconductor package.

Based on this idea, the filtering step is developed to filter out those peaks not present in at least a user-defined number of samples, defined by the `minCarriers` parameter. In the light of this, the user can decide the minimum number of samples where each peak has to be detected. In our experience, we suggest to set the samples threshold as a mutiple of the number of replicates of the conditions.

3.2.3 Counting Peaks

The counting step (`countFinalRegions` method) is designed to take a *GenomicRanges* data structure as input, where for each peak additional attributes are saved, as well as the score and the number of samples. Moreover, to quantify

the peaks given as input, it requires also the path of the alignment files where the reads are stored.

For each region the method counts the number of reads present in each sample. In so doing, it produces a matrix of counts, where the rows and the columns, respectively, represent the regions and the samples.

In order to keep track of all information associated to the regions, it produces a *SummarizedExperiment* [35] data structure, giving the possibility to retrieve the *GenomicRanges* of associated peaks and the count matrix, respectively, using the `rowRanges` and `assays` methods.

The choice to produce a count matrix is guided by the versatility of this data structure, useful not only for the differential enrichment of the regions between multiple conditions, but also for integrating the epigenomic data with other -omics data types, such as RNA-Seq.

3.2.4 Additional Features

The package offers some additional features for loading data (i.e. peaks) resulting from other sources, and for manipulating *GenomicRanges* data structure.

To give the possibility to use our pipeline with external peak callers, the function `readFilesAsGRangesList` takes as input a directory containing BAM or BED data, to load in *GenomicRangesList* format. This data structure is useful to store genomic information, as peaks or mapped reads, produced by other software like *MACS2* or *STAR* and, in case of peaks, it is necessary during the *DEScan2* filtering/aligning step. Additionally to `fileType` (BAM, BED, BED.zip) parameter specification it requires the genome code to use during the file processing. Moreover, when the input files represent peaks the `arePeaks` flag needs to be set to `TRUE`.

Furthermore, *DEScan2* provides several functionalities for GenomicRanges data structure handling. One example is `fromSamplesToChrsGRangesList`, which gives the possibility to split a GenomicRangesList by chromosome. This procedure could be useful for parallelizing computations on the chromosomes, when common operations on them, between multiple samples, are needed. Assigning a single chromosome to a single computing unit. Taken as input a *Genomic-*

RangesList organized by samples, this method returns a list of chromosomes, where each element has a *GenomicRangesList* of samples, containing only the regions associated to the single chromosome.

Other useful utilities are `keepRelevantChrs`, that takes a *GenomicRangesList* and a list of chromosomes and return only the interested chromosomes with a cleaned *genomeInfo* assigned; the `saveGRangesAsTsv` function that saves a tab separated value file starting from a GenomicRanges; the `saveGRangesAsBed` that save a standard BED file format starting from a GenomicRanges data structure; and the `setGRangesGenomeInfo` which, starting from a genome code, sets a specific *genomeInfo* to a *GenomicRanges* object.

3.3 Case Study

Data Description and Preprocessing

ATAC-seq is an emerging evoluted technique which enables to investigate the open chromatin regions at whole genome level. The capability of this technology has been demonstrated in the regulation of mouse brain activity under different conditions [36].

To illustrate the performances of *DEScan2* we chose a dataset [36] that describes in vivo adult mouse dentate granule neurons before and after synchronous neuronal activation using *ATAC-seq* and *RNA-seq* technologies (see sections 1.2.2 and 1.2.1 for a description of these sequencing techniques).

This dataset is organized in 62 samples of *ATAC-seq* and *RNA-seq*, extracted at four different time points (0, 1h, 4h, 24h), with four replicates at each time point. We chose to compare the differences between the first two stages, time 0 (E0) and 1 hour after neuronal induction (E1), in order to show a potential *ATAC-seq* workflow for Differential Enrichment, and how to integrate this data type with *RNA-seq*. A general illustration of this dataset is represented in figure 3.3.1.

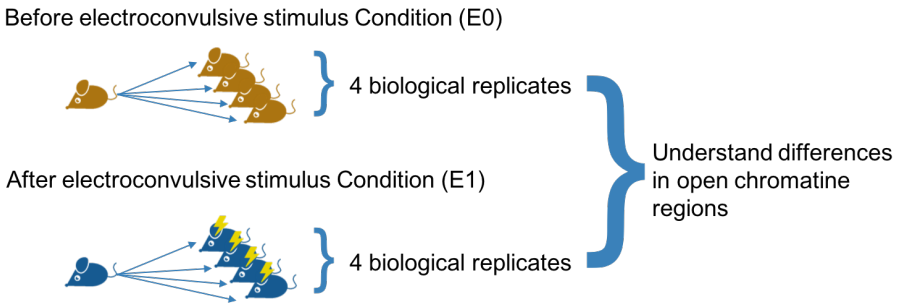


Figure 3.3.1: An illustration of our extraction of the GSE82015[36] dataset.

We downloaded the data from Gene Expression Omnibus (GEO) database [37, 38] with accession number GSE82015[1] and mapped the raw data using *STAR* [39] with default parameter on Mus Musculus Genome ver.10 (mm10).

Peaks Detection

In order to detect open chromatin regions we run our peak caller, cutting the genome in bins of 50bp and using running windows of minimum 50bp and maximum 1000bp. In this a way we are able to detect not just broad peaks, but also smaller peaks.

To be confident with our results we run *DEScan2* and *MACS2* [33] on the same samples, and (as shown in figure 3.3.2) looking to the numbers *DEScan2* always find more peaks than *MACS2*. This can be due to a major accuracy given by *MACS2* on the reliability of the detected peaks, while our method finds more peaks, but, at this stage, still preserving false positive regions.

[1] `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE82015`

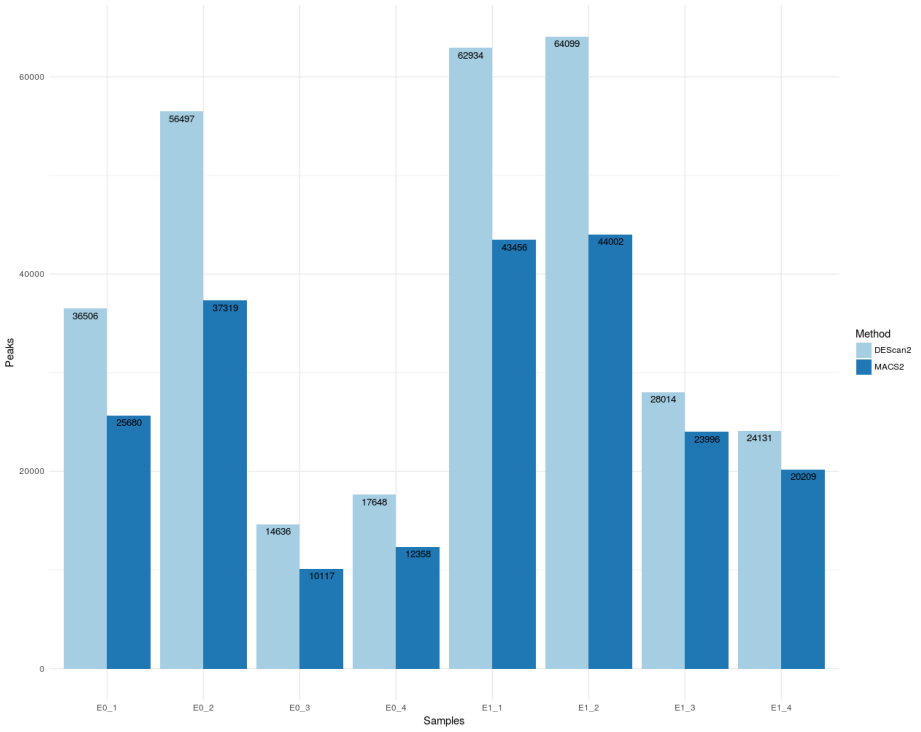


Figure 3.3.2: A comparison of *DEScan2* and *MACS2* detected peaks for each sample in the dataset.

To be more robust, we compared *DEScan2* detected peaks with the same validated regions (*Arc*[2] and *Gabrr1*[3]) of the original work [36]. The lower part of figure 3.3.3 shows the detected and validated regions (in blue and red) resulting differentially enriched between the E0 (in pink) and E1 (in green) conditions, while the upper part shows *DEScan2* filtered and aligned peaks (in blue) between the samples, highlighting a capability to catch not only the same regions of the published ones, but also (gold circles) to be more accurate in the smaller peaks detection.

[2]https://www.genecards.org/cgi-bin/carddisp.pl?gene=ARC
[3]https://www.genecards.org/cgi-bin/carddisp.pl?gene=GABRR1

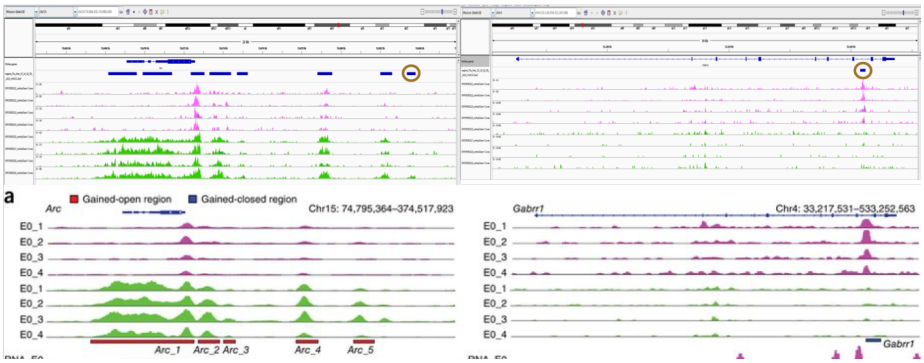


Figure 3.3.3: A comparison of *DEScan2* detected peaks with validated peaks in article [36].

Removing Unwanted Peaks

While it is very important to detect good peaks with a peak caller, it seems to be more relevant to detect reproducible regions. Indeed, during the filtering/aligning step, the number of peaks depends not only by the peak score, but also by the number of replicates designed in the experiment. The figure 3.3.4 puts in relation these two relevant information for both *MACS2* and *DEScan2*. On the x-axis is represented the number of replicates, while on the y-axis is traced the number of peaks, and each curve represents a different threshold on the peaks score, showing that the higher are the thresholds on the scores and the number of replicates, the lower is the number of the detected peaks. Highlighting a inverse relationship between the number of the peaks and the combination of the number of samples and the detected regions score.

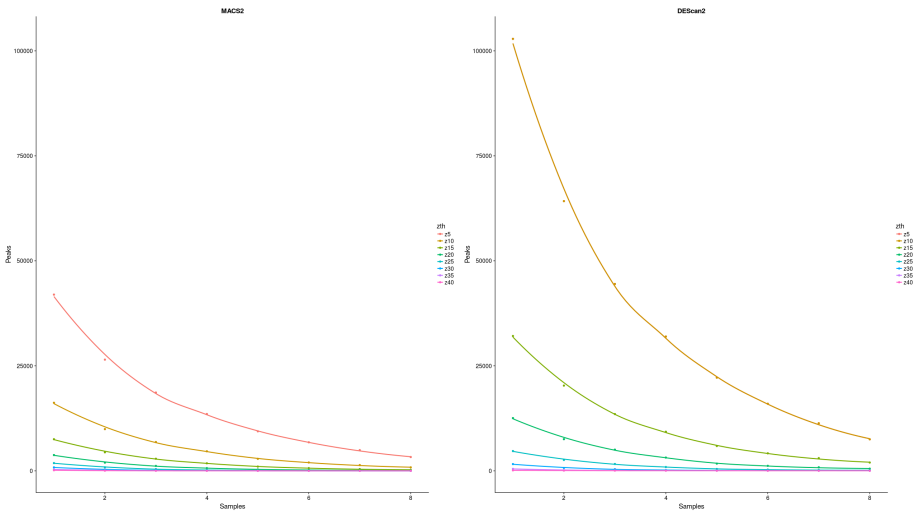


Figure 3.3.4: Filtering the detected regions with different thresholds on peak scores between *MACS2* and *DEScan2*.

Moreover, comparing left and right panels, we notice the high difference in pooling the samples-peaks together with the *DEScan2* filtering/aligning step when using *MACS2* and *DEScan2* peaks. Using *MACS2* peaks the pooling highly reduces the number of detected peaks, even using threshold as low as 5 on the score, showing that there are many peaks with a score lower than 5. While in the *DEScan2* case the curves representing the threshold equal to 5 and the threshold equal to 10 totally overlap, highlighting that the *DEScan2* peak caller produces scores higher than 10.

Quantifying Peaks

Afterwards, the filtered-in regions can be processed by *DEScan2* in order to obtain a count matrix with samples on the columns and peaks on the rows. This type of data structure is very versatile, because it enables to perform several operations, like the Differentially Enriched genomic Regions (DERs) and the integration with other kind of omics, as RNA-Seq.

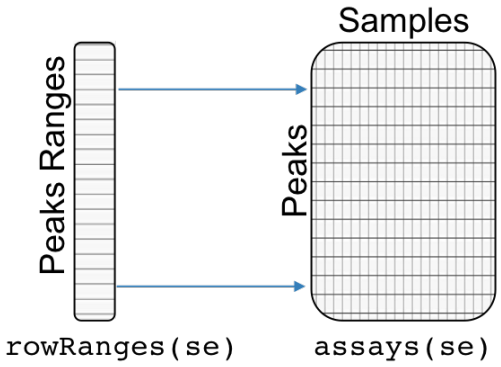


Figure 3.3.5: An illustration of the *SummarizedExperiment* data structure produced by *DEScan2*.

In order to preserve the information associated to the peaks, *DEScan2* produces as output a *SummarizedExperiment* (figure 3.3.5) data structure, which enables to retrieve the count matrix with `assays` method, and to access the peaks information in *GenomicRanges* format with the `rowRanges` method.

Peaks Normalization

Before detecting DERs, it is a good practice to normalize the data. This is especially needed when working with neuroscience data, where many possible sources of technilcal and biological noise can confound the analysis [25]. The nature of the data, in count format, makes it possible to apply several well known RNA-Seq normalizations techniques, such as *upper-quartile*, *full-quantile*, *RUV-Seq*, etc [17, 40]. To filter out false positives, but preserving enough signal at the same time, we fixed the peaks's score threshold to 20.

In order to compare the effects of normalizations we had to do differential enrichment of the conditions using *edgeR* (see next section for further details).

While the *upper-quartile* normalization affects the data in a way that makes it impossible to detect DERs, other kind of normalizations and combinantions of them give good results.

Figure 3.3.7 summarizes this concept very well, highlighting a relation between the number of DERs (y-axis) and the minumum number of samples (x-axis) used for aligning the peaks during the *DEScan2* filtering/aligning step.

To better compare the normalization effects, we created a *null dataset* of 8 samples, by shuffling the original dataset samples as combinations of conditions took in pairs. The detection of DERs has been performed on each combination (18) of shuffled samples, and then taking the median of the results.

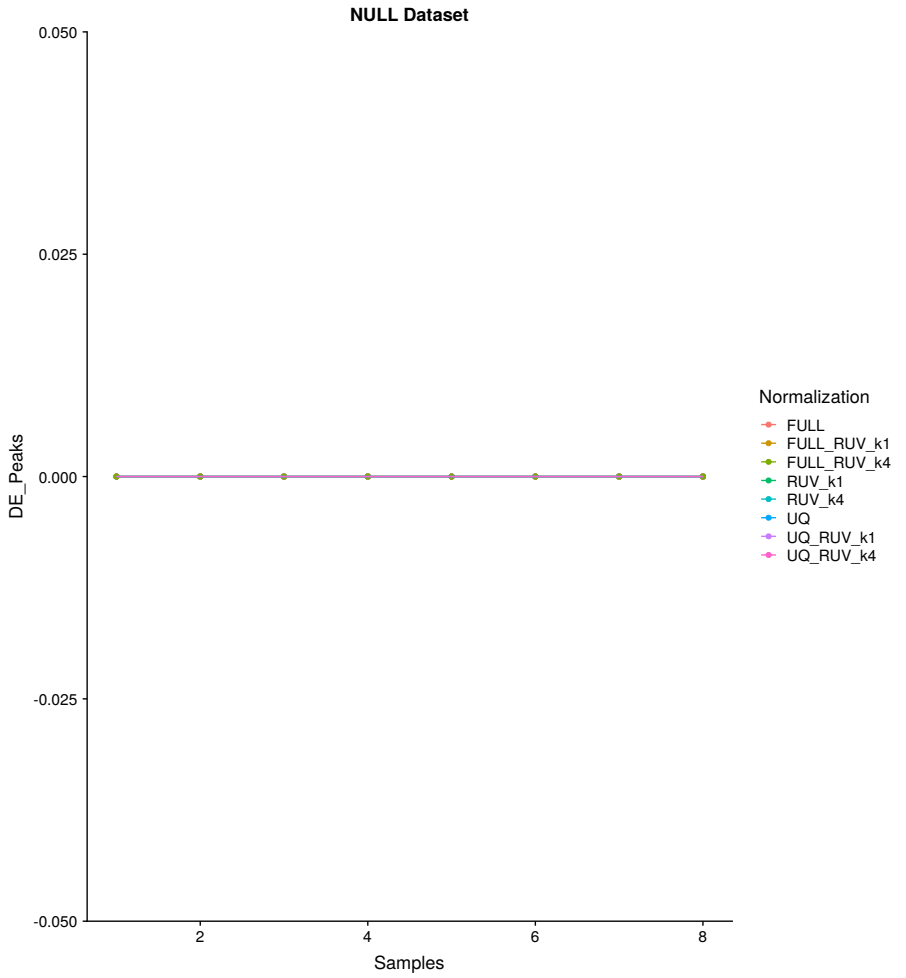


Figure 3.3.6: The figure shows the effects of different normalizations on a null dataset of epigenetic regions, putting in relation the DERs with the threshold on the samples used to align peaks.

Figure 3.3.6 shows, as expected, no DER detection. Highlighting that even if a normalization is applied, once reduced the false positives number, we are

confident with our results.

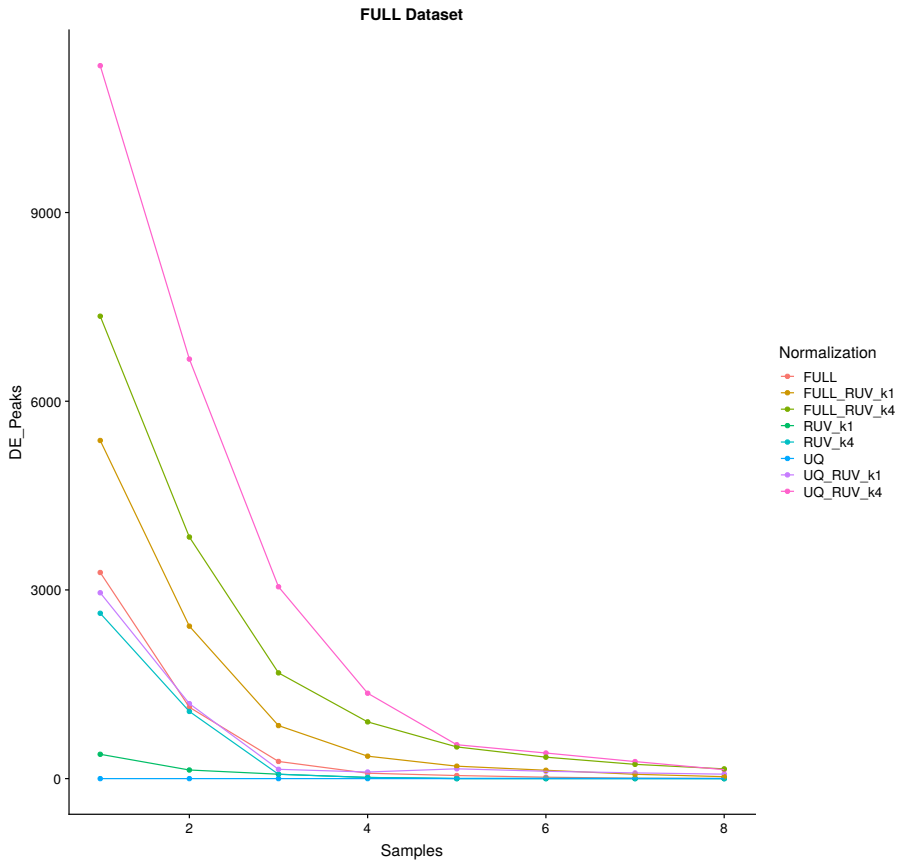


Figure 3.3.7: The figure shows the effects of different normalizations on a dataset of epigenetic regions, putting in relation the DERs with the threshold on the samples used to align peaks.

While figure 3.3.7, representing the "full dataset", shows that *upper-quantile*, by itself is not able to linearly detect any amount of DERs. When using *RUV-Seq*, the DERs detection depends on the parameter used (in figure we choose k equal to 1 and 4). While, the *full-quantile*, even when used alone is able to

detect a good amount of DERs. But each normalization, when combined with *RUV-Seq*, seems to affect the data in a way that overdetect the number of DERs. In particular, even the *Upper quartile*, which doesn't detect any signal by itself, is able to detect the highest amount of DERs when combined with *RUV-Seq*.

Even if these normalization methods show good performances with this type of epignomic data, our investigations suggest that more testing is required, and maybe an ad-hoc normalization method for these data has to be developed.

The left panel represent the "null dataset" highlighting that portion of DERs due to randomness/bias. Indeed, any kid of normalization produces almost the same trend, underlying that *full quantile*, even if combined with *RUV-Seq* still not reduces the bias. While *upper quartile* preserves oscillations when using 7/8 samples. The one which seems to well interpret the data, producing a good compromise between bias and signal, is RUV-Seq. Indeed, it preserves a gradually downhill of the DER without totally flatten the signal.

Differential Enrichment of Peaks

To estimate the DERs, any of the RNA-Seq methods can be applied, such as *DESeq2*, *edgeR*, *NOISeq*, etc [16, 21, 41].

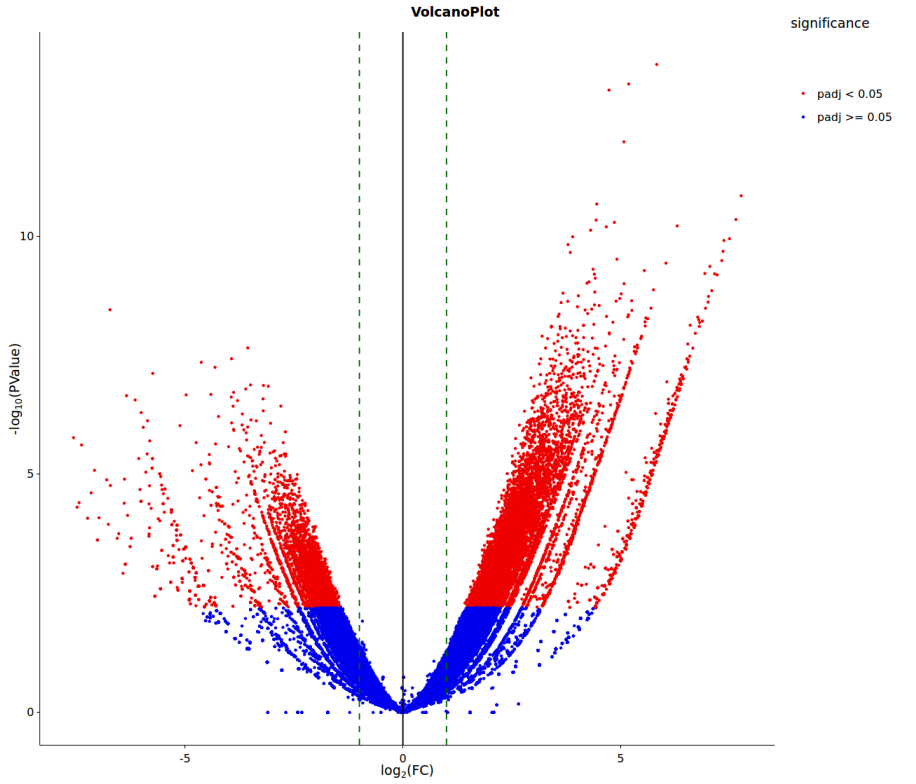


Figure 3.3.8: A volcano plot of Differential Enriched Regions. Blue dots represent the not significant DERs, while the red ones represent the significant DERs.

In this case, we decided to use *edgeR* package, because of its wide range of available statistical approaches and the possibility to better tune the design of the experiment. Indeed, because we used the RUV-Seq normalized counts with `k` parameter set to 4, we modeled the experimental design with the `model.matrix` function, adding to our model not only the experimental conditions, but also the *RUV-Seq* estimated factors. Then we used the resulted design to estimate the dispersion and fit a Quasi-Likelihood test, as defined in edgeR[21].

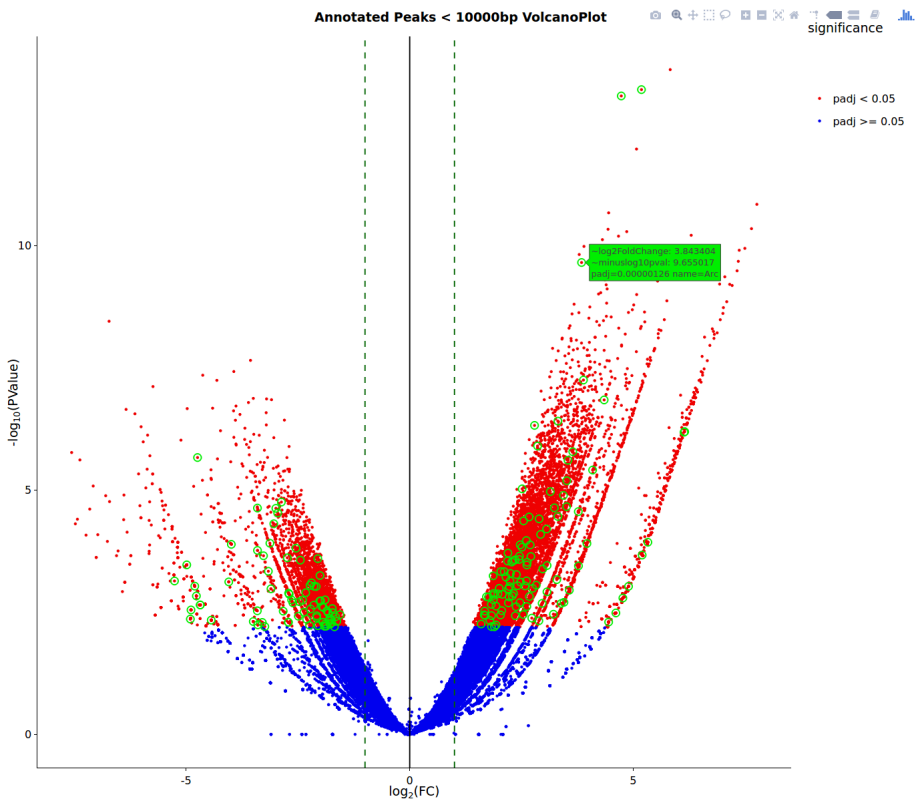


Figure 3.3.9: A volcano plot of DERs. Blue dots represent the not significant DERs, while the red ones represent the significant DERs. Green circles highlights the peaks with a DEG annotated.

Figure 3.3.8 shows a volcano plot of DERs between E0 and E1 conditions. Red dots highlight the regions with a False Discovery Rate (FDR)[42] lower than 0.05, while blue dots highlight non significant regions.

Peaks Integration

The next task is to integrate the obtained results with other omic data types, as RNA-Seq. Because of the low number of the samples, the easiest way to integrate

the data is to annotate the DERs with DEGs resulting from the analysis of RNA-Seq.

For the differential expression of the RNA-Seq data we firstly quantified the signal with the `featureCounts` methods available in the *Rsubread* [23] R/Bioconductor package. Then we filtered lowly expressed genes with the *proportion* test as implemented in *NOISeq* package, and applied the `noiseq` method for differential expression.

We used the resulting significant DEGs (with posterior probability higher than 0.95) to annotate the peaks with `annotatePeakInBatch` method of *ChIPpeakAnno*. Figure 3.3.9 illustrates with green circles the peaks with an annotated gene with distance lower than 10000bp from the gene TSS, producing a total of 430 annotated peaks. Realizing the plot with *ggplot2* combined with *plotly* library it is possible to enhance the names of the genes with a tooltip.

Then we used the annotated genes to do functional annotation on Gene Ontology (GO) [11, 12] and Reactome pathways, which showed several interesting results for the neuronal regulation.

3.4 Conclusions and Future Works

In the lack of methodologies for open chromatin region detection and analysis, we developed a novel approach which, compared with very well known tools as *MACS2*, seems to be competitive in the detection of the signal.

We demonstrated to be able to catch not only wide signal, but also narrow regions across the samples. And with our filtering/aligning step we demonstrated to be able to keep relevant signl producing data structures as *SummarizedExperiment* which are candidates to become standards in the biological data analysis. With our 3-steps analysis we puts our tool at the top of a pipeline for open chromatin regions data analysis, proposing also a possible candidate for a standard analysis of this data type.

In the next future we plan to check if other distributions, as *Negative Binomial*, fit better this kind of data and to improve our filtering/aligning step with additional probabilistic methodology.

Chapter 4

IntegrHO - Integration of High-Throughtput Omics data

4.1 Implementation Aspects

Integration of High-Throghput Omics data (*IntegrHO*) is a web based Graphical User Interface (GUI) for the analysis and the integration of multiple Next Generation Sequencing (NGS) data with the aid of several already published packages designed for this aim.

For the GUI implementation we used the R *Shiny* libraries because of its power to render R code in web format.

The tool presents itself with a main upper menu of main topics organized by scope. For each of this topic, a sub-menu with specific functionalities is available. Moreover, depending on the selected functionality, a side menu is presented with additional functionalities or with input parameters to setup the specific tool. After the parameter setup the results in graphical or table format are presented in the main part of the interface. (figure 4.1.1)

Before to proceed to its data analysis, it is mandatory for the user to setup

the project with a dedicated interface. The user has to upload a design matrix which describes the information related to its samples, some of them are mandatory as the filename (with path) of the BAM files and the condition of each sample, while others are optional as the tissue or the run id. It is also possible to edit the design matrix by hand directly from the interface. In such a way *IntegrHO* creates in the working directory (returned by the `getwd()` function) a dedicated folder with all the required subfolders and stores all the basic information of the project into an ad-hoc designed `R6ProjectClass` which will be re-used during the whole session to speed up the configuration of each step of the analysis.

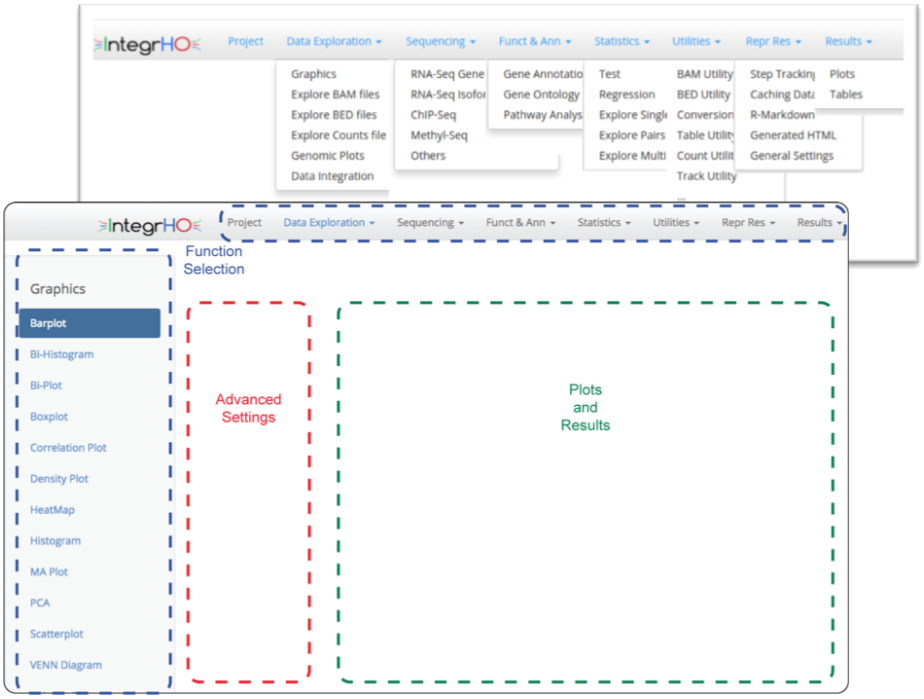


Figure 4.1.1

IntegrHO implements several functionalities and methodologies for *RNA-*

Seq, *ChIP-Seq* and *ATAC-Seq* data analysis, complementing these aspects by providing methodologies for their integration at different levels, as functional enrichment with Gene Ontology and Pathways, peaks and genes annotation, and more statistical methods, as mixOmics, working with high-dimensional datasets.

For each -omics, *IntegrHO* takes as input the BAM files, previously defined in the design matrix of the main project definition interface.

For each step of the analysis of each -omics, we tried to select more than one method to perform that task. In so doing we give the possibility to try different approaches in order to compare the final results, tuning the methods on the basis of the dataset under investigation.

For *RNA-Seq* we constructed a dedicated interface for each step of a standard RNA-Seq data analysis pipeline, such as to build a count matrix, to filter out low counts with multiple tests, to normalize them and to account for batch effect. Moreover, we selected multiple methods for DEG, such as *edgeR*, *DESeq2*, *NOISeq*.

For *ChIP-Seq* we constructed specific interfaces for peak calling, annotation and DERs detection. For the peak calling, because of the lack of specific methods starting from BAM files, we implemented interfaces for the DEScan2 and the *csaw*[43] peak callers. The first one born for broad peaks identification and the second one for both broad and narrow peaks quantification. For the annotation we chose the *ChIPpeakAnno* and the *ChIPseeker* R/Bioconductor, which produces similar output formats starting from peaks. While for the detection of DERs we used the same methods as for *RNA-Seq*.

For *ATAC-Seq* we used mostly the same methods implemented for the *ChIP-Seq*, but designing specific interfaces for the filtering/alignment and the counting matrix as implemented in the DEScan2 package (see chapter 3 for further details).

To provide an integration of these -omics, we dedicated an entire section to this aspect, with functionalities for the annotation of DERs with DEGs using *ChIPpeakAnno* and to use this information to investigate the functional response by enriching for Gene Ontology or for Pathways. These last two aspects implemented with aid of *g:Profiler* [44] and *graphite* [45] R/Bioconductor

packages.

Moreover, to work with high-dimentional data sets, we are working on the implementation of methods like mixOmics, which gives a graphical response of the samples or the features, using two different methods diablo and mint.

4.2 Reproducible Computational Research

The most difficult part when using a GUI is to trace the executed functionalities during the analysis. To face this need we equipped *IntegrHO* of a Reproducible Research (RR) hidden layer able to trace all the code executed by the user.

In combination with a system of caching database files (CDF), it stores the code chunks and the input/output data of each analysis step into an R Markdown (RMD) file.

Because of the need of adding personal comments to each analysis step or to delete portions of the analysis, we built a specific interface enabling the user to edit the automatically produced RMD file and to compile it on the fly.

The enriched output report can be produced in Hypertext Markdown Language (HTML) or Portable Document Format (PDF) formats, in order to be easily attached as supplementary material of a published article, facilitating the reproducibility of the analysis to a third party user.

Chapter 5

Easy Reporting: a reproducible computational research R6 Class

In the -omics data field, the complexity of the analysis, due to the high dimensionality of the data and to the wide range of methdologies to use, has revived an interest in RR, because of the difficulties in reproducing third party scientific findings.

As previous chapters **??** clearly show, the huge amount of computational tools, statistical methods and data visualization approaches used for High-Throughput (HTP) data analysis and their integration ...

During last years several approaches (add reference) in different programming languages (such as HTML, *Python*, *R*, etc.) have been proposed for helping to trace the analysis steps.

The common underlying idea of these instruments is to provide a mixture of natural language sentences along with computational language (*Code Chunks* (CCs)) and visual outputs, in order to produce a unique final product where the CCs and their outputs are explained to the reader, enhancing comprensibility and reproducibility of the work.

To address this scope the R community proposed several solutions, like

sweave before, *knitr* and *rmarkdown*[1] later. Due to it's easy interactive usage, *rmarkdown* became one of the most used instruments in *R* community, but its usability when developing automated instruments like GUI or packages becomes more difficult, leading developers to give up using it.

Here we present *easyReporting*, an *R6* [2][3] class which helps developers to integrate *rmarkdown* into their computational products.

The project is accessible at the following link:
https://github.com/drighelli/easyreporting

5.1 Introduction

5.2 Methods

EasyReporting is a class[4] builded with the intention of helping developers to integrating a reproducible research layer inside their software products.

In such a way, thanks to a minimal additional effort of the developer, the end user has available an *rmarkdown* file within all the source code generated during the analysis, divided into CCs ready for the compilation.

Once manually edited with comments and descriptions the file can be compiled to produce an enriched document within input data, source code and output results.

A so final document can be attached to the publication of the analysis as supplementary material, helping the interested community to entirely reproduce the computational part of work.

General Description and Initialization

The class has to be seen as a representation of the *rmarkdown* file (*report*), indeed it has been provided of a list of attributes describing the characteristics

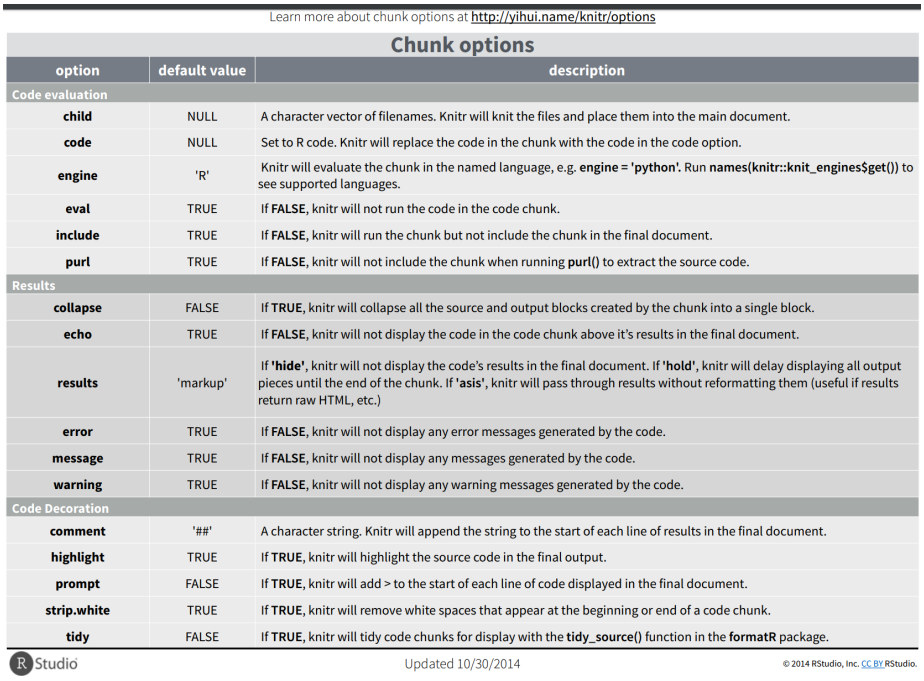
[1] `https://rmarkdown.rstudio.com/`
[2] `https://adv-r.hadley.nz/r6.html`
[3] `https://cran.r-project.org/web/packages/R6/index.html`
[4] `https://en.wikipedia.org/wiki/Class_(computer_programming)`

of the *report*, which will be inserted in the header of the file.

The class methods are not only for the attributes manipulations, but also for adding CCs to the *report*.

Before of using it, *easyReporting* requires to be initializated with the `new` command, passing as mandatory arguments the *path* and the name of the file as `filenamepath` and a title as `mainTitle`. Additionally, an `author` and the `documentType` can be specified.

When initializing, the class automatically creates the *report* with the entire specified folder tree, setting up the header of it and declaring the general options for the *rmarkdown* file. If *rmarkdown* personal options (see figure 5.2.1) are required, before creating an instance of the class, it is possible to use the `makeOptionsList` function, and then assigning the output to the `optionsList` argument of the class `constructor`.



Learn more about chunk options at http://yihui.name/knitr/options

Chunk options

option	default value	description
Code evaluation		
child	NULL	A character vector of filenames. Knitr will knit the files and place them into the main document.
code	NULL	Set to R code. Knitr will replace the code in the chunk with the code in the code option.
engine	'R'	Knitr will evaluate the chunk in the named language, e.g. **engine = 'python'**. Run **names(knitr::knit_engines$get())** to see supported languages.
eval	TRUE	If **FALSE**, knitr will not run the code in the code chunk.
include	TRUE	If **FALSE**, knitr will run the chunk but not include the chunk in the final document.
purl	TRUE	If **FALSE**, knitr will not include the chunk when running **purl()** to extract the source code.
Results		
collapse	FALSE	If **TRUE**, knitr will collapse all the source and output blocks created by the chunk into a single block.
echo	TRUE	If **FALSE**, knitr will not display the code in the code chunk above it's results in the final document.
results	'markup'	If **'hide'**, knitr will not display the code's results in the final document. If **'hold'**, knitr will delay displaying all output pieces until the end of the chunk. If **'asis'**, knitr will pass through results without reformatting them (useful if results return raw HTML, etc.)
error	TRUE	If **FALSE**, knitr will not display any error messages generated by the code.
message	TRUE	If **FALSE**, knitr will not display any messages generated by the code.
warning	TRUE	If **FALSE**, knitr will not display any warning messages generated by the code.
Code Decoration		
comment	'##'	A character string. Knitr will append the string to the start of each line of results in the final document.
highlight	TRUE	If **TRUE**, knitr will highlight the source code in the final output.
prompt	FALSE	If **TRUE**, knitr will add > to the start of each line of code displayed in the final document.
strip.white	TRUE	If **TRUE**, knitr will remove white spaces that appear at the beginning or end of a code chunk.
tidy	FALSE	If **TRUE**, knitr will tidy code chunks for display with the **tidy_source()** function in the **formatR** package.

RStudio Updated 10/30/2014 © 2014 RStudio, Inc. CC BY RStudio.

Figure 5.2.1

Class Methods

The class is provided of several methods for *rmarkdown* CC construction.

Once an *easyReporting* instance is available, with its `mkdTitle` is possible to insert six levels of titles, by setting the parameters `title` and `level`. It is also possible to add natural language comments with `mkdGeneralMsg`.

When working with CCs, two main possibilities are available. The first one gives the possibility to construct a CC as additional steps, by using first the `mkdCodeChunkSt`, then adding variable assignment and/or function calling with `mkdVariableAssignment` or `mkdGeneralMsg`, and finally closing the CC with `mkdCodeChunkEnd`.

In particular, when starting a CC with `mkdCodeChunkSt`, it is possible to assign a specific `optionList` and/or a `source.files.list` to be added to that

CC.

The second way gives the possibility to create an entire CC just with `mkdCodeChunkComplete` and assigning the entire function call as a `message`. This way is really useful with function calls, where inside a function a simple recursive call with parameters assignment can be done as single `message`.

Usage Example

Here we report a script where a fast illustration of the package is reported.

```
1  ## creating report file with default options on global
       document
2  rd <- easyreporting$new(filenamepath="./project_report",
       title="example_report", author=c("Dario␣Righelli"))
3
4  rd$mkdTitle("First␣Level␣Title")
5
6  rd$mkdGeneralMsg("Here␣I'm␣writing␣a␣simple␣paragraph␣useful
       ␣to␣describe␣my␣code␣chunk")
7
8  ## leaving the default options to the code chunk
9  rd$mkdCodeChunkSt()
10 ## adding a variable assignement
11 variable <- 1
12 rd$mkdVariableAssignment("variable", "variable", show=TRUE)
13 rd$mkdCodeChunkEnd()
14
15 ## or i can create my own options for the chunk
16 optList <- maketOptionsList(includeFlag=TRUE)
17 rd$mkdCodeChunkSt(optionsList=optList)
18 rd$mkdCodeChunkEnd()
19
20 ## moreover I can add a list of files to source in che code
       chunk
21 rd$mkdCodeChunkSt(optionsList=optList, source.files.list=c("
       R/cachingFunctions.R", "R/cachingFunctions.R"))
```

```
22 rd$mkdCodeChunkEnd()
23
24
25 rd$mkdCodeChunkComplete(message="a␣<-␣1\nb␣<-␣2\nc␣<-␣a+b\n␣
       print(c)")
26
27
28 ## otherwhise i can make a direct call with all the code
       chunk and the code as message
29
30
31 rd$compile()
```

5.3 Future Works

Appendices

.1 R Language

.2 R Markdown Language

Chapter 6

Bibliography

1. Thermes, C. *Ten years of next-generation sequencing technology* 2014. doi:10.1016/j.tig.2014.07.001. arXiv: arXiv:1312.0570v2.
2. Wang, Z., Gerstein, M. & Snyder, M. *RNA-Seq: A revolutionary tool for transcriptomics* 2009. doi:10.1038/nrg2484. arXiv: NIHMS150003.
3. Costa, V., Angelini, C., De Feis, I. & Ciccodicola, A. *Uncovering the complexity of transcriptomes with RNA-Seq* 2010. doi:10.1155/2010/853916.
4. Ozsolak, F. & Milos, P. M. *RNA sequencing: Advances, challenges and opportunities* 2011. doi:10.1038/nrg2934. arXiv: NIHMS150003.
5. Costa, V., Aprile, M., Esposito, R. & Ciccodicola, A. *RNA-Seq and human complex diseases: Recent accomplishments and future perspectives* 2013. doi:10.1038/ejhg.2012.129.
6. Pepke, S., Wold, B. & Mortazavi, A. Computation for chip-seq and rna-seq studies. *Nature Methods* **6,** S22. ISSN: 15487105 (2009).
7. Oshlack, A., Robinson, M. D. & Young, M. D. *From RNA-seq reads to differential expression results* 2010. doi:10.1186/gb-2010-11-12-220.

8. Giresi, P. G., Kim, J., McDaniell, R. M., Iyer, V. R. & Lieb, J. D. FAIRE (Formaldehyde-Assisted Isolation of Regulatory Elements) isolates active regulatory elements from human chromatin. *Genome Res.* **17,** 877–885. ISSN: 09254773 (2007).

9. Winter, D. R., Song, L., Mukherjee, S., Furey, T. S. & Crawford, G. E. DNase-seq predicts regions of rotational nucleosome stability across diverse human cell types. *Genome Research* **23,** 1118–1129. ISSN: 10889051 (2013).

10. Buenrostro, J. D., Giresi, P. G., Zaba, L. C., Chang, H. Y. & Greenleaf, W. J. Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. *Nature Methods* **10,** 1213–1218. ISSN: 15487091 (2013).

11. Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research.* ISSN: 1362-4962. doi:10.1093/nar/gkh036 (2004).

12. Gene Ontology Consortium. Gene Ontology Consortium: going forward. *Nucleic acids research.* ISSN: 1362-4962. doi:10.1093/nar/gku1179 (2015).

13. Gentleman, R. *et al.* Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology.* ISSN: 1465-6914. doi:10.1186/gb-2004-5-10-r80 (2004).

14. Liao, Y., Smyth, G. K. & Shi, W. FeatureCounts: An efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics.* ISSN: 14602059. doi:10.1093/bioinformatics/btt656. arXiv: 1305.3347 (2014).

15. Sha, Y., Phan, J. H. & Wang, M. D. *Effect of low-expression gene filtering on detection of differentially expressed genes in RNA-seq data* in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* (2015). ISBN: 9781424492718. doi:10.1109/EMBC.2015.7319872. arXiv: 15334406.

16. Tarazona, S., García, F., Ferrer, A., Dopazo, J. & Conesa, A. NOIseq: a RNA-seq differential expression method robust for sequencing depth biases. *EMBnet.journal* **17,** 18. ISSN: 2226-6089 (2012).

17. Risso, D., Ngai, J., Speed, T. P. & Dudoit, S. Normalization of RNA-seq data using factor analysis of control genes or samples (RUVSeq). *Nature Biotechnology* **32,** 896–902. ISSN: 1087-0156 (2014).

18. Costa-Silva, J., Domingues, D. & Lopes, F. M. *RNA-Seq differential expression analysis: An extended review and a software tool* 2017. doi:10.1371/journal.pone.0190152. arXiv: arXiv:1804.06050v3.

19. Love, M. I., Huber, W. & Anders, S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology.* ISSN: 1474760X. doi:10.1186/s13059-014-0550-8. arXiv: arXiv:1303.3997v2 (2014).

20. Nueda, M. J., Tarazona, S. & Conesa, A. Next maSigPro: Updating maSigPro bioconductor package for RNA-seq time series. *Bioinformatics.* ISSN: 14602059. doi:10.1093/bioinformatics/btu333 (2014).

21. Robinson, M. D., McCarthy, D. J. & Smyth, G. K. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics (Oxford, England)* **26,** 139–140. ISSN: <null> (2009).

22. Kanehisa, M., Sato, Y., Kawashima, M., Furumichi, M. & Tanabe, M. KEGG as a reference resource for gene and protein annotation. *Nucleic Acids Research.* ISSN: 13624962. doi:10.1093/nar/gkv1070 (2016).

23. Liao, Y., Smyth, G. K. & Shi, W. The Subread aligner: Fast, accurate and scalable read mapping by seed-and-vote. *Nucleic Acids Research.* ISSN: 03051048. doi:10.1093/nar/gkt214 (2013).

24. Carlson, M. *org.Mm.eg.db: Genome wide annotation for Mouse.* 2018. doi:https://doi.org/doi:10.18129/B9.bioc.org.Mm.eg.db.

25. Peixoto, L. *et al.* Survey and Summary: How data analysis affects power, reproducibility and biological insight of RNA-seq studies in complex datasets. *Nucleic Acids Research.* ISSN: 13624962. doi:10.1093/nar/gkv736 (2015).

26. Soneson, C. & Delorenzi, M. A comparison of methods for differential expression analysis of RNA-seq data. *BMC bioinformatics* **14,** 91. ISSN: 1471-2105 (2013).

27. Bullard, J. H., Purdom, E., Hansen, K. D. & Dudoit, S. Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. *BMC Bioinformatics* **11.** ISSN: 14712105. doi:10.1186/1471-2105-11-94. arXiv: NIHMS150003 (2010).

28. Koberstein, J. N. *et al.* Learning-dependent chromatin remodeling highlights noncoding regulatory regions linked to autism. *Science Signaling.* ISSN: 19379145. doi:10.1126/scisignal.aan6500 (2018).

29. Auerbach, R. K. *et al.* Mapping accessible chromatin regions using Sono-Seq. *Proceedings of the National Academy of Sciences.* ISSN: 0027-8424. doi:10.1073/pnas.0905443106 (2009).

30. Ihaka, R. & Gentleman, R. R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics.* ISSN: 15372715. doi:10.1080/10618600.1996.10474713. arXiv: arXiv:1011.1669v3 (1996).

31. Li, H. *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics.* ISSN: 13674803. doi:10.1093/bioinformatics/btp352. arXiv: 1006.1266v2 (2009).

32. Lawrence, M. *et al.* Software for Computing and Annotating Genomic Ranges. *PLoS Computational Biology* **9** (ed Prlic, A.) e1003118. ISSN: 1553-7358 (2013).

33. Zhang, Y. *et al.* Model-based analysis of ChIP-Seq (MACS). *Genome Biology.* ISSN: 14747596. doi:10.1186/gb-2008-9-9-r137 (2008).

34. Zhu, L. J. *et al.* ChIPpeakAnno: A Bioconductor package to annotate ChIP-seq and ChIP-chip data. *BMC Bioinformatics.* ISSN: 14712105. doi:10.1186/1471-2105-11-237 (2010).

35. Morgan M, Obenchain V, Hester J, P. H. SummarizedExperiment: SummarizedExperiment container. doi:https://doi.org/doi:10.18129/B9.bioc.SummarizedExperiment (2018).

36. Su, Y. *et al.* Neuronal activity modifies the chromatin accessibility landscape in the adult brain. *Nature Neuroscience.* ISSN: 15461726. doi:10.1038/nn.4494 (2017).

37. Edgar, R. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Research.* ISSN: 13624962. doi:10.1093/nar/30.1.207 (2002).

38. Barrett, T. *et al.* NCBI GEO: Archive for functional genomics data sets - Update. *Nucleic Acids Research.* ISSN: 03051048. doi:10.1093/nar/gks1193 (2013).

39. Dobin, A. *et al.* STAR: Ultrafast universal RNA-seq aligner. *Bioinformatics.* ISSN: 13674803. doi:10.1093/bioinformatics/bts635. arXiv: 1201.0052 (2013).

40. Dillies, M. A. *et al.* A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. *Briefings in Bioinformatics.* ISSN: 14675463. doi:10.1093/bib/bbs046 (2013).

41. McCarthy, D. J., Chen, Y. & Smyth, G. K. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Research* **40,** 4288–4297. ISSN: 03051048 (2012).

42. Benjamini, Y. & Hochberg, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological).* ISSN: 00359246. doi:10.2307/2346101. arXiv: 95/57289 [0035-9246] (1995).

43. Lun, A. T. & Smyth, G. K. Csaw: A Bioconductor package for differential binding analysis of ChIP-seq data using sliding windows. *Nucleic Acids Research* **44.** ISSN: 13624962. doi:10.1093/nar/gkv1191. arXiv: arXiv:1011.1669v3 (2015).

44. Reimand, J. *et al.* g:Profiler-a web server for functional interpretation of gene lists (2016 update). *Nucleic acids research* **44,** W83–W89. ISSN: 13624962 (2016).

45. Sales, G., Calura, E., Cavalieri, D. & Romualdi, C. Graphite - a Bioconductor package to convert pathway topology to gene network. *BMC Bioinformatics* **13.** ISSN: 14712105. doi:10.1186/1471-2105-13-20 (2012).

List of Figures

List of Tables