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 this data type, it helps the analysis of time course RNA-Seq data, offering a vast amount of hypothesis tests, setup for differential expression between two or more conditions. With the help of edgeR, DESeq2 and nextMASigPro 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, starting from the counting 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 also

graphics as KEGG-maps and heatmaps.

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.

Moreover, it's possible to compare the results of different analysis and to investigate the most influenced biological functions (i.e. Gene Ontology terms and Pathways).

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

It gives the possibility to analyse time course RNA-Seq data starting from *BAM* files. It enables RNA expression quantification with `featureCounts` method producing a count matrix useful for Differentially Expressed Genes (DEGs) detection.

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

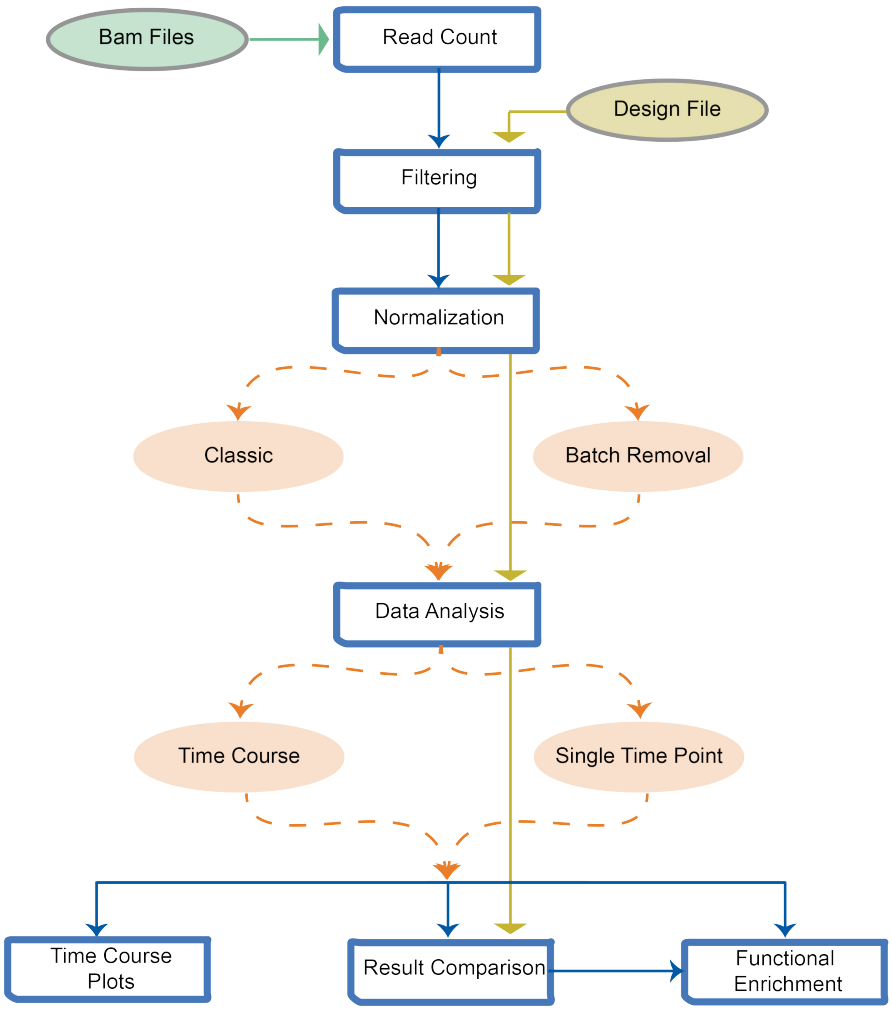

Figure 2.2.1: Main flow of ticorser R package.

A particular attention is given to the normalization phase, providing the possibility not only to use several traditional normalizations methods, but also the possiblity to remove batch effect.

In oder to inspect different approaches to analyze TC data, *ticorser* offers four different methodologies for analyzing time course RNA-Seq data, and three different ways 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 [1] influencing final results.

In order to addres this task, *ticorser* offers multiple statistical methods for low expressed genes filtering. The `filterLowCounts` method is partly based on the `filtered.data` function of *NOISeq* R/Bioconductor package [2]. We 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 less 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 the features with a relative expression equal to $cpm/10^6$.

While, 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 mean counts between all the samples, higher than the `quantile.threshold` argument (default is 99%).

2.2.2 Data Normalization

Also for the normalization process *ticorser* offers several ways to normalize the data.

Thanks to the `normalizeData` function we are able to offer five different normalization methods without taking care of additional information. The design

matrix based system allows to 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 [3]. In particular, *ticorser* offers to apply *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. 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.

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 four 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 [4] R/Bioconductor packages, *DESeq2*[5];*MASigPro*[6]; *edgeR*[7]; *NOISeq*[2].

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:

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

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 in 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:

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

Time-Course DE Method 4 - *nextMASigPro*

The fourth method takes advantage of the Generalized Linear Model (GLM) with Negative Binomial distribution as defined in the *maSigPro* R/Bioconductor package. This method, unlike the previous ones, allows to detect all the DEGs showing any kind of differences between the conditions across all the time points. Indeed, as suggested by the *maSigPro* authors is a good norm to cluster the genes to better understand which is their singular behaviour.

Single DE Methods

Our package offers a way to analyze the differences between the conditions at single time point, offering 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, applying a posterior probability in both cases.

2.2.4 Data Visualization

While analyzing RNA-Seq data, it is almost mandatory to explore data during each step of the analysis, in order to understand which is the best method to apply for the next step and to be sure that the analyzed data are trusty.

In order to help to analyze RNA-Seq data, we implemented in *ticorser* several useful graphics 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.

Exploration Plots on Counts

For exploring counts data we implemented two very common 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 according to the time-point they belonging to. It is a box divided in two parts, with an outgoing segment from each side. The upper and lower part of the box represent the first and third quartile, while in the middle is represented the median. The upper and lower part of the segments represent the minimum and maximum values of the sample distribution.

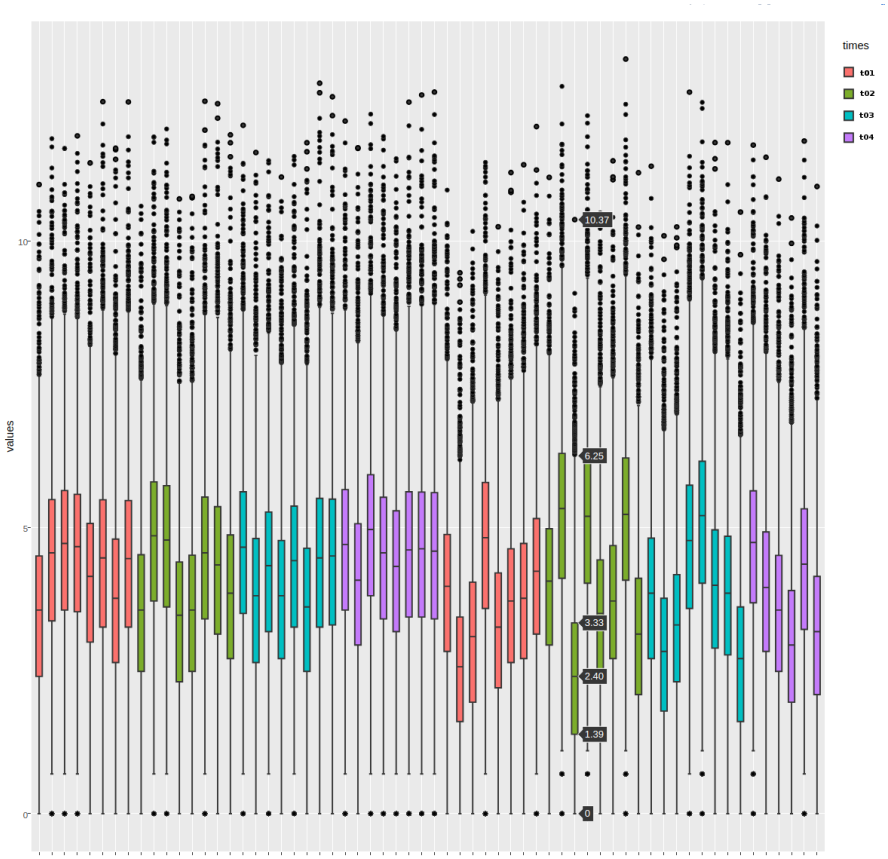

Figure 2.2.2: An example of interactive boxplot made with *ticorser* package. Each boxplot represents a specific sample, while each colour represents a specific time point. When passing the mouse over a boxplot it shows additional information about its quartiles.

Due to the very high dimensionality of RNA-Seq data, it is widely considered common sense to apply the PCA dimensionality reduction technique, which allows 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 `plotPCA` function, by using the `prcomp` from the *stats* package. The user needs just 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, colouring the samples according to the groups identified in the chosen column. Moreover, by setting to `TRUE` the `ellipses` argument, the function will plot also the ellipses surrounding each samples group.

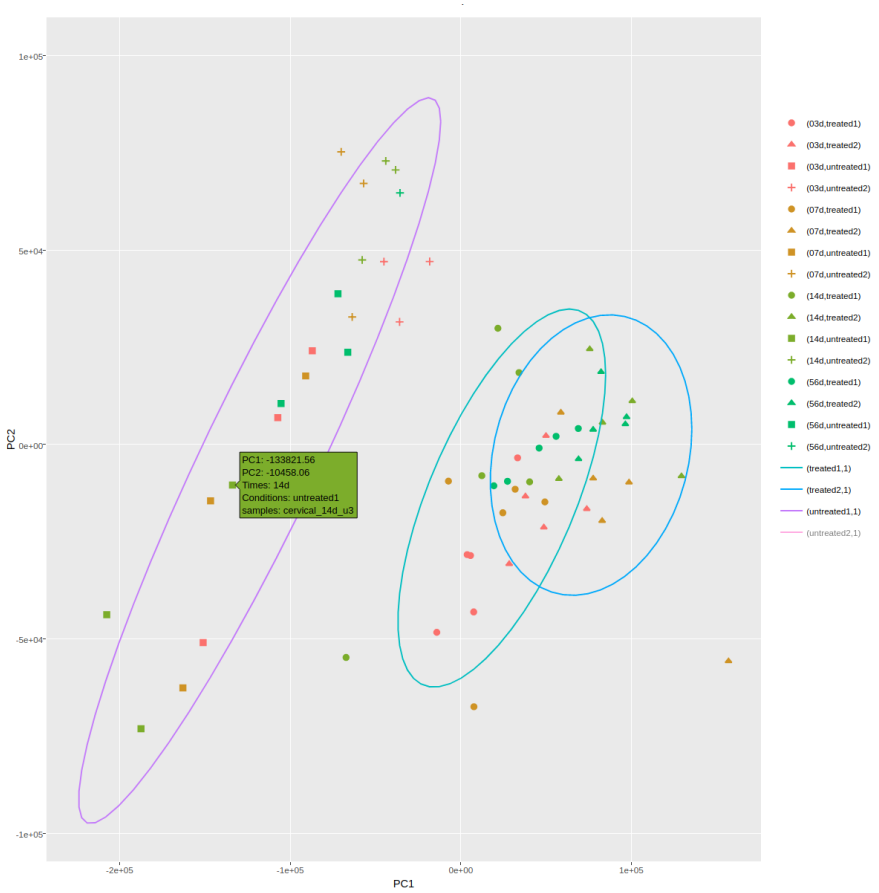

Figure 2.2.3: An example of interactive PCA made with *ticorser* package. Each dot represents a specific sample, while each colour represents a time point, each simbol represents a biological condition group. When passing the mouse over a dot it shows additional information about the selected sample, 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 implementations takes 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.

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.

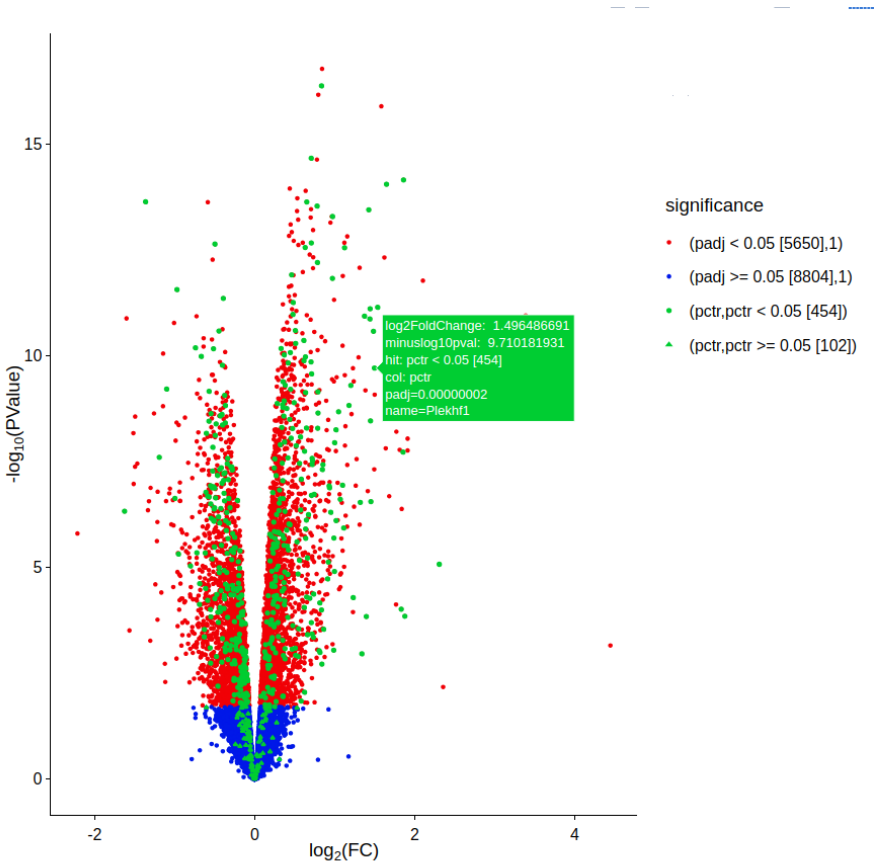

Figure 2.2.4: An example of interactive volcano plot made with *ticorser* package. Each dot represents a gene, while blue and red colours highlights the significance of the genes. In green there are those genes coming from the positive control list. When passing the mouse over a dot it shows additional 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 $log_2(T/C)$ where T is the treatement condition and C its con-

trol. It is not mandatory to have a DE results dataframe to plot an MA-plot, but it's pretty useful to have it in order to understand the distribution of the significant genes.

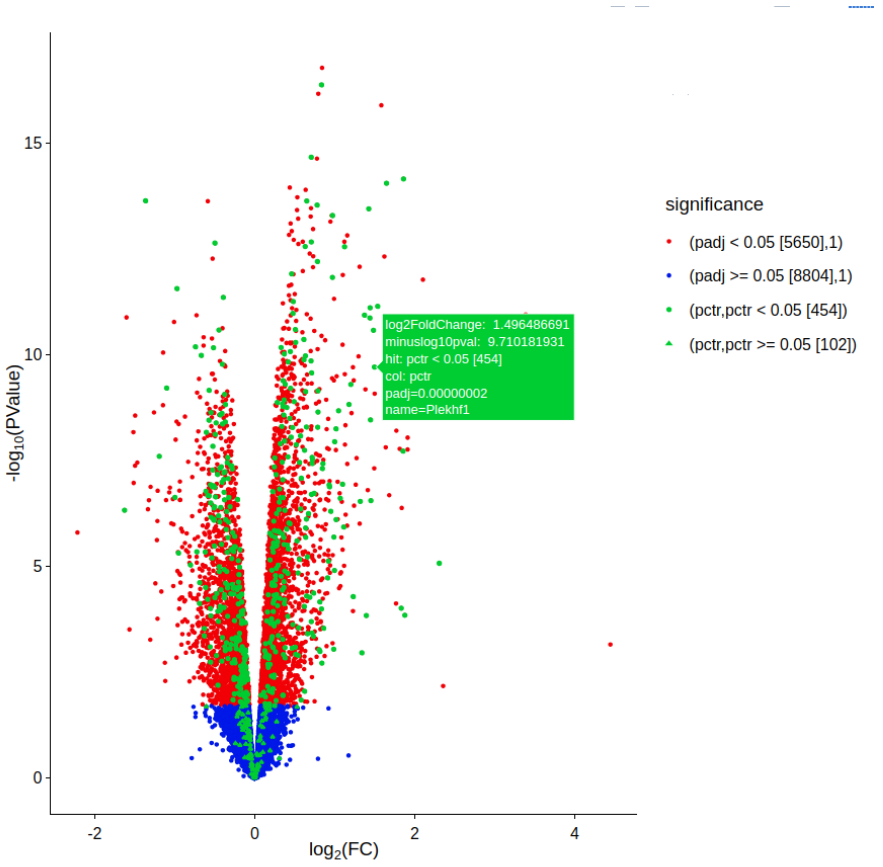

Figure 2.2.5: An example of interactive MA-plot made with *ticorser* package. Each dot represents a gene, while blue and red colours highlights the significance of the genes. When passing the mouse over a dot it shows additional information about the selected gene.

Gene Profiles plot

For highlighting the profile of a gene, we implemented the function `plotGeneProfile` , where takes as input the count matrix, a gene name, and the design matrix. 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.

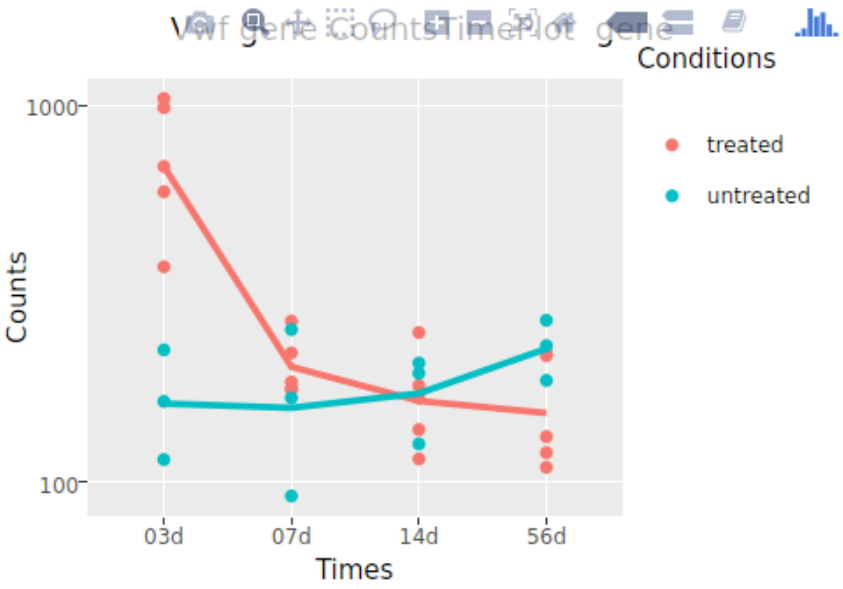

Figure 2.2.6: An example of interactive gene profile made with *ticorser* package. Each dot represents the counts value of the gene in a sample. Colours identifies the conditions of the samples. The lines represent the gene trend over all time points.

keggmap

ticorser offers the possibility to plot *keggmaps*[8] 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 plot.

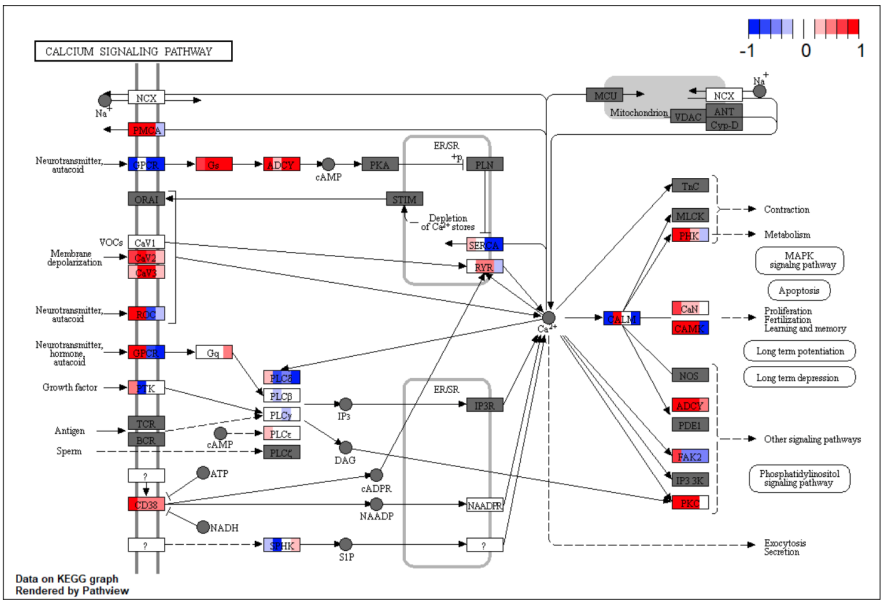

Figure 2.2.7: add description

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

One feature is aimed to give the possibility to quantify the gene expression starting from samples mapping files. To provide this functionality we builded up the `countBamFilesFeatureCounts` function with the aim to guide and facilitate this operation to the user. It is based on the `featureCounts` method of the *Rsubread*

R/Bioconductor package [9] 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* [10] 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

2.5 Conclusions and Future Works

Chapter 3

Differential Enriched Scan 2 DEScan2

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

Some studies [11, 12] demonstrated the importance of genomewide 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 lack, 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 the RNA-Seq, need the comparison of two or more different biological conditions. Starting from a set of already published [11] scripts, we designed Differential Enriched Scan 2 (*DEScan2*), a software for helping the analysis of chromatin accession sequencing data.

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

3.1 Introduction

The *DEScan2* is an R [13] 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 [14] data structures and methods, and it is available through 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 on 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 counts 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 so produced counts matrix, as illustrated in the 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.

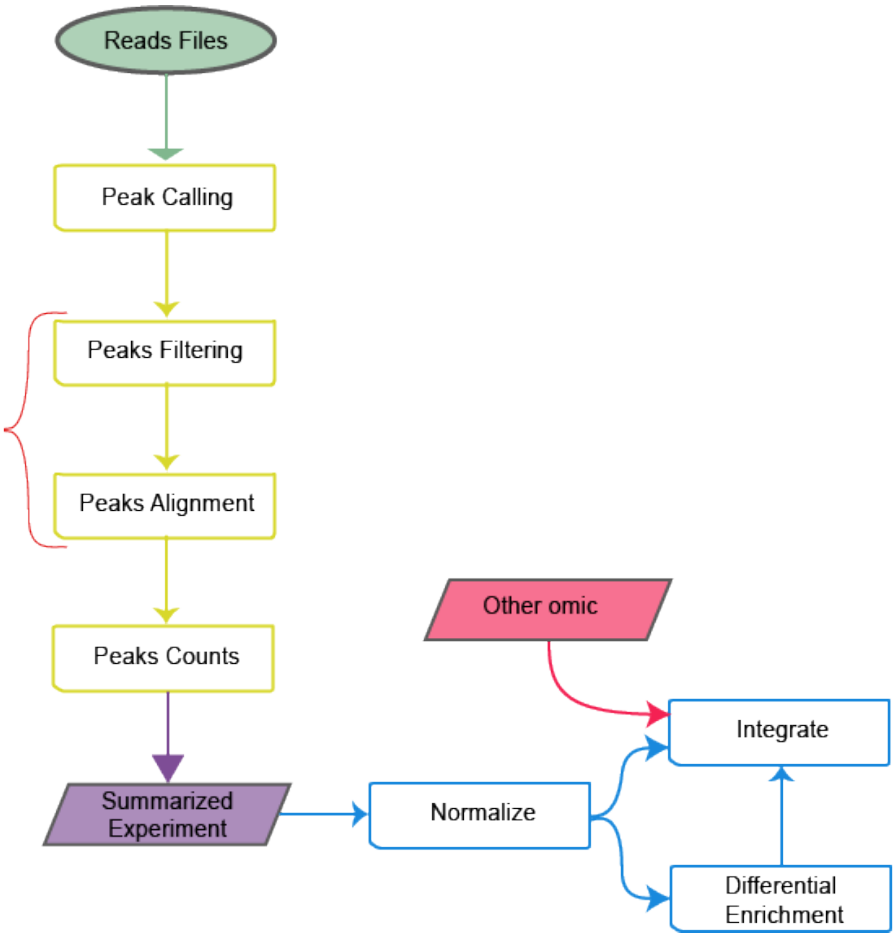

Figure 3.2.1: A differential enrichement flow representation. *DEScan2* steps are highlighed in yellow.

3.2.1 Peak Caller

The Peak Caller (defined by the `findPeaks` function) takes as input a set of alignment files (BAM [15] 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 *GenomicRangesList* [16], 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 as bins of `binSize` parameter length (the default value is 50bp) and then computes the reads coverage on the bins with moving scan windows, spanning from `minWin` to `maxWin` parameters of `binSize` interval.

In order to be able to catch small and spread 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 coverages 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 coverages 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 derive with a MLE, described as the estimator:

$$\lambda_n = \frac{1}{n} \sum_{i=1}^{n} x_i$$

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 the 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 [17], 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 *ChIPpeakAnno* [18] 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. On 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 the counts, where the rows and the columns, respectively, represent the regions and the samples.

In order to keep trace of all information associated to the regions, it produces a *SummarizedExperiment* [19] 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 omic data types, 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 peaks, 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 over the others is `fromSamplesToChrsGRangesList`, which gives the possibility to split a GenomicRangesList by the chromosomes. This procedure could be useful for parallelizing the 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 *GenomicRangesList* 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. It has been demonstrated the adequacy of this technology in the regulation of mouse brain activity under different conditions.

To illustrate the performances of *DEScan2* we chose a dataset [20] 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 btween the first two stages, time 0 (E0) and 1 hour after neuronal induction (E1), in order to show a potential Atac-Sec 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.

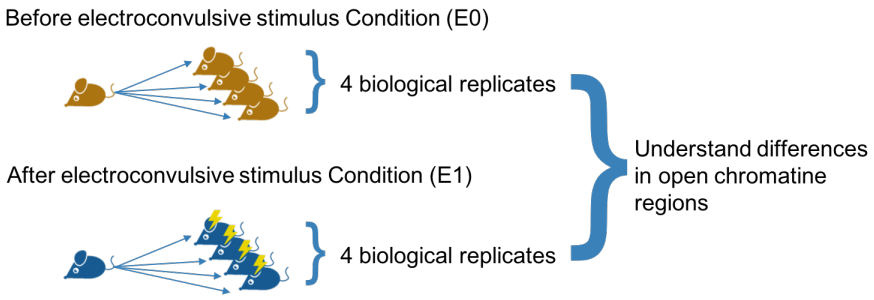

Figure 3.3.1: An illustration of our extraction of the GSE82015[20] dataset.

We downloaded the data from Gene Expression Omnibus (GEO) database [21, 22] with accession number GSE82015[1] and mapped raw data using *STAR* [23] 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 such 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* on the same samples, and (as shown in figure 3.3.2) looking to the numbers *DEScan2* always outperforms *MACS2* peaks.

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

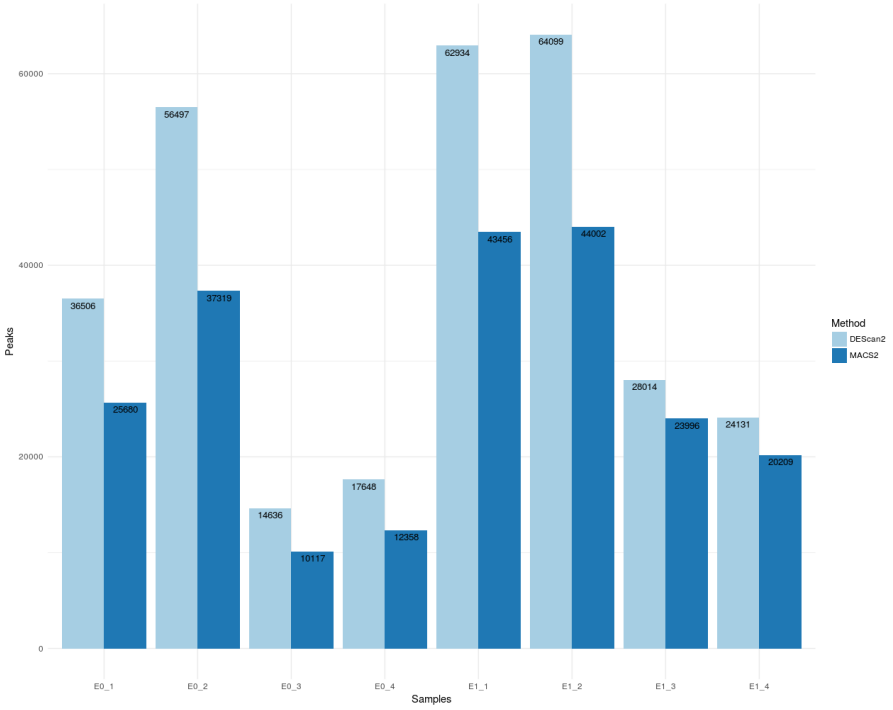

Figure 3.3.2: A comparison of *DEScan2* and *MACS2* detected peaks for each sample in the dataset.

To be more robust, we compared *DEScan2* detected peaks with the same validated regions (*Arc*[2] and *Gabrr1*[3]) of the original work [20]. 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 careful 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

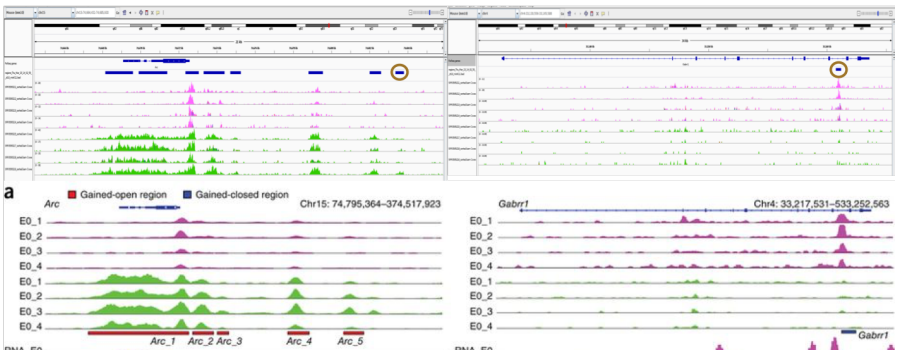

Figure 3.3.3: A comparison of *DEScan2* detected peaks with validated peaks in article [20].

Removing Unwanted Peaks

While it is very important to detect good peaks with a peak caller, it seems to be more relevant to detect reliable 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 higher are the thresholds on the scores and the number of replicates, lower is the number of the detected peaks. Highlighting a proportional inversion between the number of the peaks and the combination of the number of samples and the detected regions score.

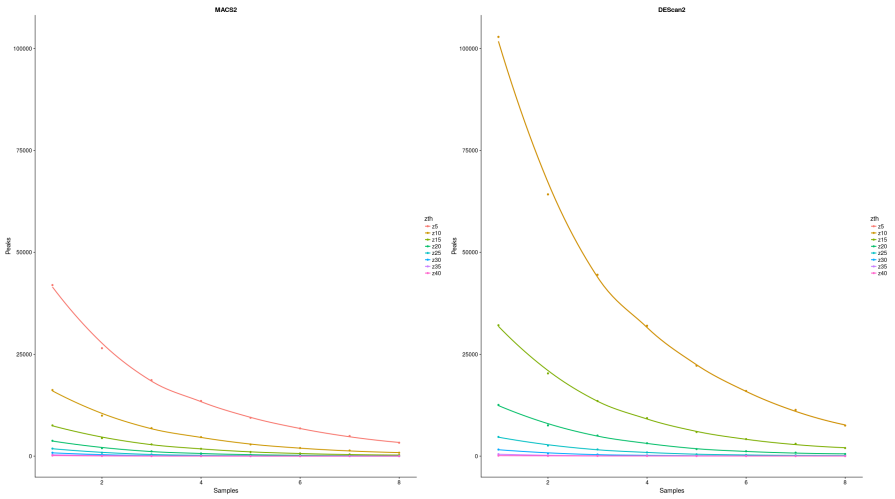

Figure 3.3.4: Filtering the detected regions with different thresholds on peak scores between *MACS2* and *DEScan2*.

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 the *MACS2* and the *DEScan2* peaks. Using the *MACS2* peaks the pooling highly reduce the number of detected peaks, even using a low threshold 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.

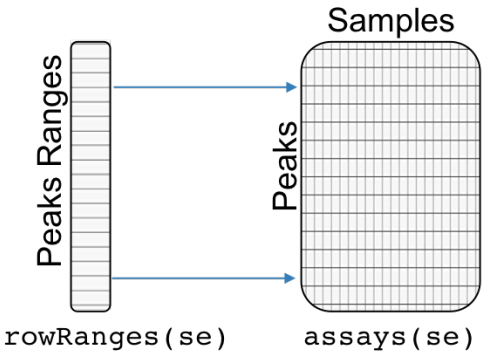

Figure 3.3.5: An illustration of the *SummarizedExperiment* data structure produced by *DEScan2*.

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 to proceed to detect DERs, it is a good standard to normalize the data, also because without any kind of normalization we are not able to detect any DER. The nature of the data, in count format, makes it possible to apply several well known RNA-Seq normalizations techniques, such as *TMM*, *upper-quartile*, *full-quantile*, *RUV-Seq*, etc [3, 24, 25]. In this case, we fixed the peaks's score threshold to 10, in order to have as much signal as possible.

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

The figure 3.3.6 sintetizes this concept very well, highlighting a relation between the number of DERs (y-axis) and the minumum number of samples (x-axis) used for filtering the data during the *DEScan2* filtering step.

To better compare the normalization effect, we created a *null dataset* of 8 samples, simply doubling the E0 samples and differentially express the signal between E0 (4 samples) and E0-fake (4 samples).

The right panel of the plot, representing the "full dataset", shows that *upper-quantile*, even if combined with *RUV-Seq* normalization, is not able to linearly detect a good amount of DERs, while *full-quantile*, when combined with *RUV-Seq* seems to affect the data in a way that overdetect the number of DERs. When looking at the *full-quantile* and *RUV-Seq* by themself seem to perform better than the other normalizations. The first one has a downhill almost linear, while the second one has a very fast downhill with a regrowth when the number of samples is higher.

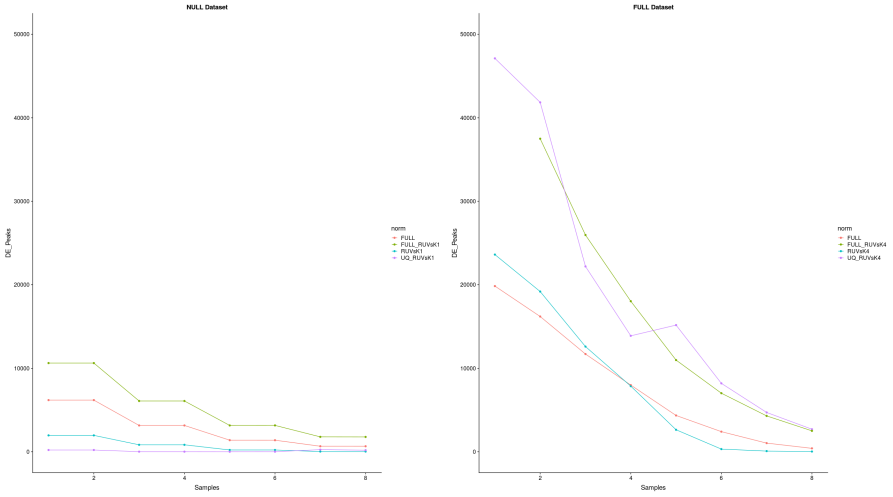

Figure 3.3.6: The figure shows the effects of different normalizations on the epigenomic differentially enriched regions.

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.

A note to be accounted is that in the case of the "null dataset" we had to set the RUV-Seq `k` parameter to 1, otherwhise we were no able to obtain any result.

Differential Enrichment of Peaks

To estimate the DERs, any of the RNA-Seq methods can be applied, such as *DESeq2*, *edgeR*, *NOISeq*, etc [2, 7, 26].

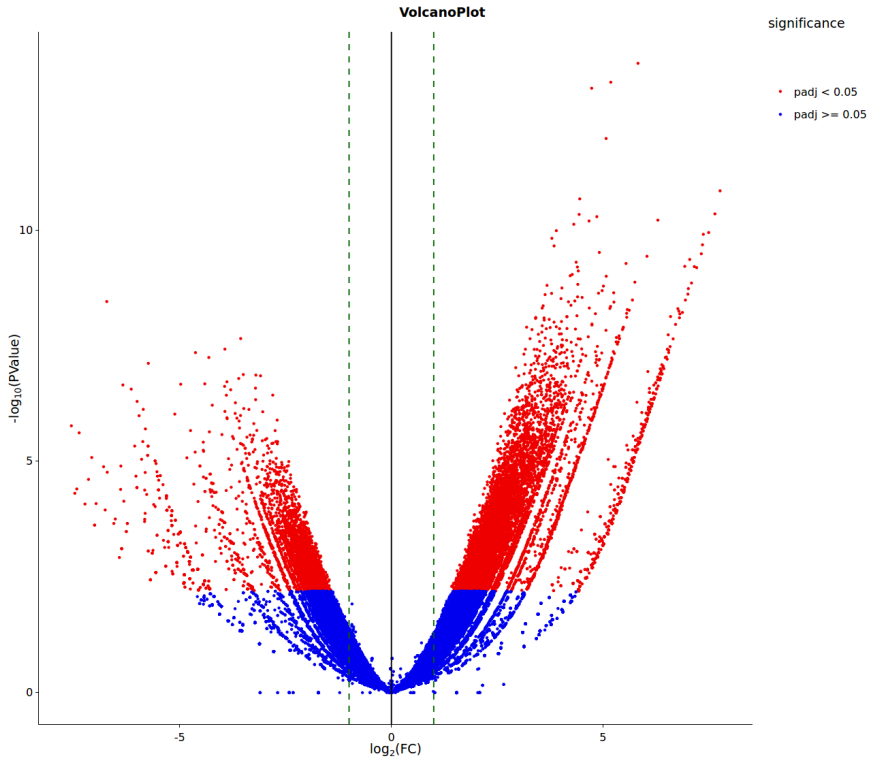

Figure 3.3.7: A volcano plot of Differential Enriched Regions. Blue dots represent the not significant DERs, while the red ones represent the significant DERs.

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 weights. Then we used the resulted design to estimate the dispersion and fit a Quasi-Likelihood test, as defined in edgeR[7].

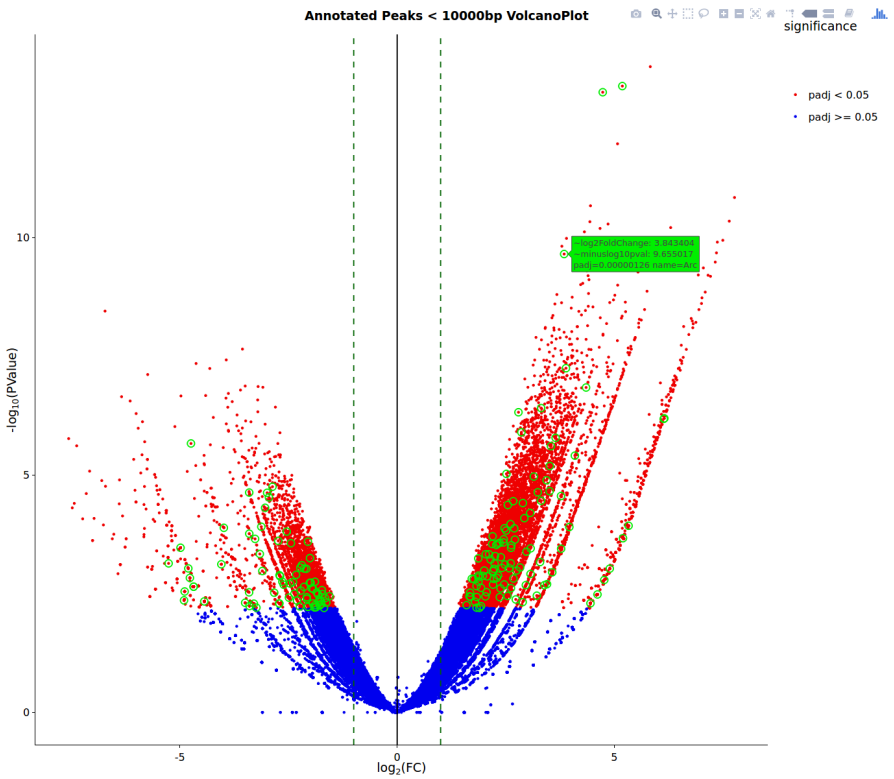

Figure 3.3.8: A volcano plot of DERs. Blue dots represent the not significant DERs, while the red ones represent the significant DERs. Green circles highlights the peaks with a DEG annotated.

The figure 3.3.7 shows a volcano plot of DERs between E0 and E1 conditions. Red dots highlights the regions with a False Discovery Rate (FDR)[27] lower than 0.05, while blue dots highlight not significant regions.

Peaks Integration

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 `featureCounts` methods available in the *Rsubread* [9] 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 3.3.8 illustrates with green circles the peaks with an annotated gene with distance lower than 10000bp from the gene Transcription Starting Site (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) [28, 29] and Reactome pathways, which showed several interesting results for the neuronal regulation.

Insert tables for the functional results, to discuss with Davide/Lucia

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, and, because it's newly born, aims to be more poweful and attractive in this field.

We demonstrated to be able to catch not only spread signal, but also small 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*, fits better this data kind and to improve our filtering/aligning step with additional probabilistic methodology.

3.5 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 3.5.1)

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.

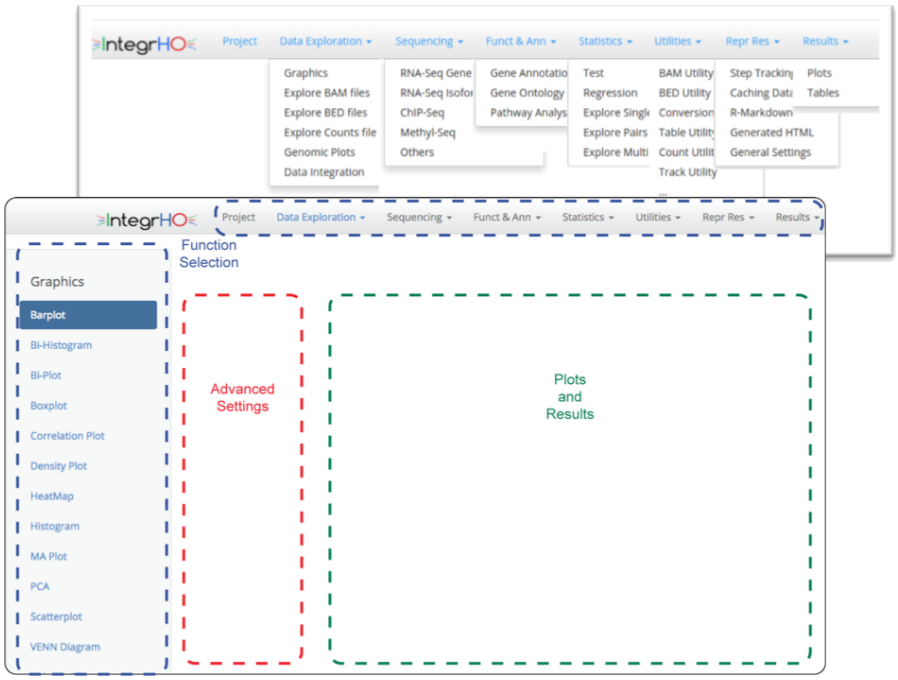

Figure 3.5.1

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*[30] 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* [31] and *graphite* [32] 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.

3.6 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.