

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

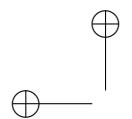
Relatori

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

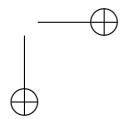
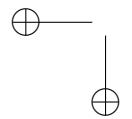
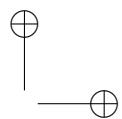
Candidato

Righelli Dario
Matr. 8800800010

ANNO ACCADEMICO 2017/2018



“thesis” — 2018/12/12 — 17:49 — page 2 — #2



*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*

⊕

—

“thesis” — 2018/12/12 — 17:49 — page 4 — #4

⊕

—

⊕

—

⊕

—

Add acknowledgements here

⊕

⊕

“thesis” — 2018/12/12 — 17:49 — page 6 — #6

⊕

⊕

⊕

⊕

⊕

⊕

Abstract

Massive parallel sequencing technologies are producing a vast amount of genome-wide data about cells, tissues and model organisms, useful to understand many of biological mechanisms, like protein-chromatin interactions (e.g. ChIP-Seq), DNA methylation (Methyl-Seq or BS-Seq), chromatin accessibility (e.g. Atac-Seq), global transcriptional and translational activities (e.g. RNA-Seq) and 3-D organisation of chromatin (e.g. Hi-C), giving the possibility to study same individual or experimental condition from many different points of view (transcriptomics, epigenomics, etc.) with a very high resolution. Each type of these “omic” data explains a different aspect of cellular behaviour. To give a comprehensive view of the cell regulatory mechanisms it is necessary not only to perform a single level analysis but also to provide novel statistical and computational models for integrating different omic types within a unified study.

This thesis is focused on the development of three main computational tools (*Ticorser*, *DEScan2* and *IntegrHO*), allowing data analysis and integration of multiple next-generation sequencing experiments. Additionally, a fourth tool (*easyReporting*) for reproducible computational research is presented.

Ticorser (time course RNA-seq data analyser) is a novel R package aimed to analyse time-course RNA-seq data. It offers multiple methods for differential expression data analysis and provides multiple plots useful to explore and visualize the results at each step of the analysis. Furthermore, it also provides methods for functional integration by annotating genes in pathways and GO-terms.

DEScan2 (Differential Enriched Scan 2) is a novel R package for ATAC-seq data analysis, one of the emerging techniques for investigating chromatin accessibility. It consists in the following three-step procedure: 1) It identifies candidate regions inside each sample with a peak caller; 2) It filters out potential artefacts by aligning the candidate regions between the samples and removing those candidate regions that were not reproducible between samples 3) It produces a count matrix of regions and samples, useful for differential enrichment between multiple conditions and also for integrating this data type with other omics data, such as RNA-Seq.

Contents

Acknowledgements	5
Abstract	7
1 Introduction	13
1.1 Biological Background	13
1.1.1 The cell	13
1.1.2 The DNA	14
1.2 Sequencing Techniques	17
1.2.1 RNA-seq	18
1.2.2 ChIP-seq	20
1.2.3 ATAC-seq	23
1.2.4 Multi-omics Integration	24
2 Time Course RNA-seq analyzer ticorser	27
2.1 Introduction	27
2.2 Methods	28
2.2.1 Filtering low counts	28
2.2.2 Data Normalization	30

2.2.3	Differential Expression	31
2.2.4	Data Visualization	33
2.3	Additional Features	42
2.4	Case Study	44
2.5	Future Works	62
3	Differential Enriched Scan 2	
	DEScan2	65
3.1	Introduction	66
3.2	Methods	66
3.2.1	Peak Caller	67
3.2.2	Peak Filtering and Alignment	68
3.2.3	Counting Peaks	69
3.2.4	Additional Features	70
3.3	Case Study	71
3.4	Future Works	83
4	A reproducible computational research R6 Class	
	easyReporting	85
4.1	Introduction	85
4.2	Methods	86
4.3	Usage	89
4.4	Future Works	93
5	Integration of High-Throughput Omics data	
	IntegrHO	95
5.1	Introduction	95
5.2	Methods	96
5.2.1	The interface	96
5.2.2	Available Methodologies	98
5.3	Reproducible Computational Research	100
5.4	Future Works	101

6 Conclusions	103
7 Bibliography	105
List of Figures	115
List of Tables	119

Chapter 1

Introduction

This chapter explains some basic information useful to understand the context where this thesis work has been developed. Explaining some biological basic aspects and how it is possible to study the cellular behaviour from multiple points of view using different sequencing techniques.

1.1 Biological Background

1.1.1 The cell

Cells are the fundamental units of every living being, which can be made up of one cell (unicellular) or more (multicellular). Independently on how big and complex an organism could be, each cell always maintains its individuality and its independence but maintaining common structural proprieties.

The internal volume is defined by the *cytoplasm*, which is a liquid solution where several insoluble particles stand, such as enzymes, *RNA* and metabolites. Moreover, it is possible to distinguish multiple organelles, such as *ribosomes*, the *endoplasmic reticulum*, the *golgi complex*, *lisosomes* and the *nucleus* (figure 1.1.1). In particular, this last one has the role of containing the genome,

represented by the *DNA*.

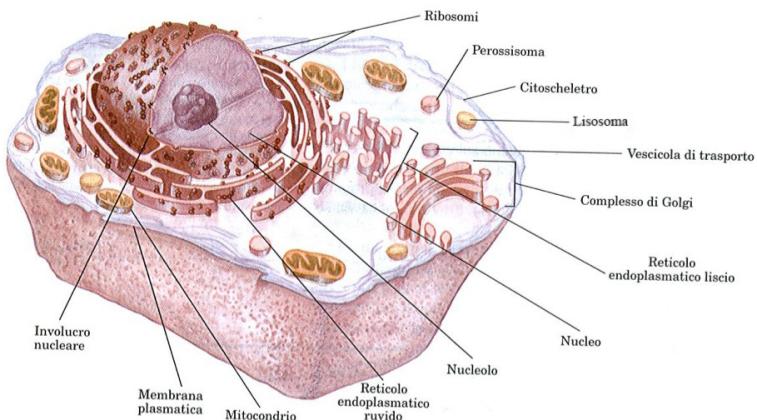


Figure 1.1.1: Schematic representation of a cell.

1.1.2 The DNA

The *DNA* was been isolated for the first time by the German doctor Friedrich Miescher in 1869, while in the same decade the English biologist Charles Darwin was publishing *On the Origin of the Species* and the Augustinian friar scientist was communicating his results on the pees to the Brunn Natural History Society.

Because the substance isolated by Miescher was white, lightly acid and present only into the cells nuclei, it was been termed *Nucleic Acid*. Name modified afterwards in DeoxyriboNucleic Acid (DNA), to distinguish is from another one, very similar, the RiboNucleic Acid (RNA).

These two molecules are constituted by *nucleotides*, constituted by a nitrogen base, deoxyribose sugar and a phosphate group. We distinguish two nitrogen bases, purines and pyrimidines. Inside the DNA, we have two *pyrimidines*, *adenine* (A) and *guanine* (G), and two *pyrimidines*, the *Cytosine* (C) and the *Thimine* (T). Inside RNA *Thymine* is substituted by the *Uracil* (U).

DNA structure (figure 1.1.2) was discovered, in the '50s, by the American scientist James Watson, the French physicist Francis Crick and the English

1.1. BIOLOGICAL BACKGROUND

15

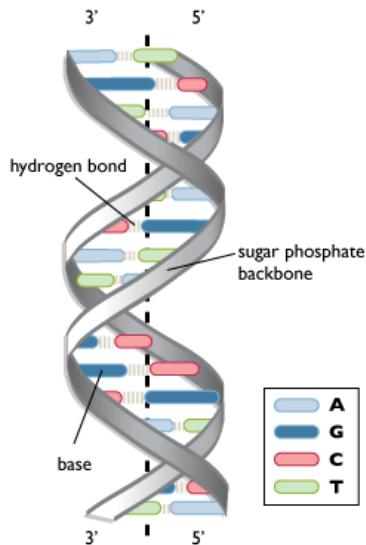


Figure 1.1.2: Schematic representation of double-stranded filament structure of DNA. The legend report the four nitrogen bases, Adenine, Guanine, Cytosine and Thymine.

chemist-physicist Rosalind Franklin. According to their model, the DNA is a double-stranded filament, where Adenines can pair only with Thymines and Guanines only with Cytosines. The four bases constitute the alphabet for the genetic message.

DNA is folded on itself (*DNA packaging*, thanks to specific "beads" called *nucleosomes*, which themselves consist of eight proteins with tails, called *histones*, that have the DNA wrapped on them. This mechanism enables to store around 2 meters of chromatin inside a nucleus of a 2-10 micron diameter when referring to Human species.

Moreover, the DNA contains the *genes*, particular sections containing relevant information for building proteins and other fundamental molecules for the cellular behaviour regulation. Each gene is localized on a precise position of a *Chromosome*, which are in a different number for each species. Each chromo-

some is constituted by DNA within thousands of genes.

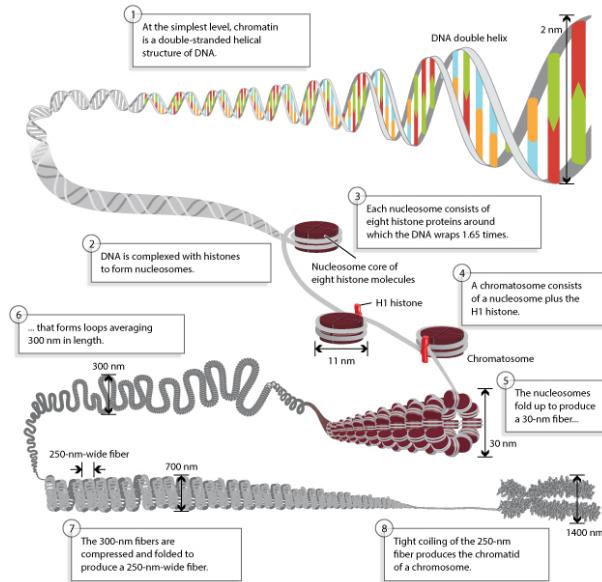


Figure 1.1.3: Representation of the relation between DNA and Chromosomes. Inside the cell nucleus, there are pairs of chromosome, constituted by chromatin, which fundamental unit is constituted by nucleosomes, on which the DNA is wrapped around, containing the genetic information in gene form. (image adapted from [1])

Figure 1.1.3 better helps to understand the relationship between chromosomes, chromatin, nucleosomes and genes.

It is important to underlying that since some decades ago the Central Dogma of Molecular Biology was founded on the transcription-translation principle, where DNA was transcribed in RNA, which subsequently it would have been translated into protein.

Nowadays, we know that the gene transcription is regulated by several mechanisms, and moreover, the translation is not the only process fated for RNA.

Indeed, for a transcription of a gene, there are some requisites to be respected, such as the accessible of that specific part of the chromatin, or the binding of specific proteins enabling the accession to the gene region, or the his-

1.2. SEQUENCING TECHNIQUES

17

tone modification processes, such as *Acetylation*, *Methylation*, *Phosphorylation*, and others, which modifies the state of specific histones, and influencing gene expression regulation.

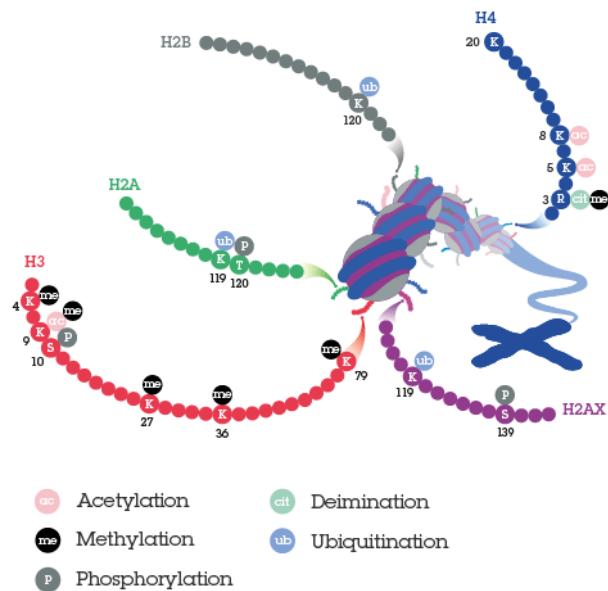


Figure 1.1.4: Representation of some processes involved in histone modification, influencing gene expression regulation.

1.2 Sequencing Techniques

To study multiple aspects of cellular behaviour related to different experimental conditions, due to drug treatments, pathologies, diseases, etc, several sequencing technologies have been developed during the last decades.

Starting with Sanger sequencing first and passing through Microarrays technologies then, nowadays with Next Generation Sequencing (NGS) we are able to understand many of biological mechanisms, like protein-chromatin interactions

(e.g. ChIP-seq), DNA methylation (Methyl-seq or BS-seq), chromatin accessibility (e.g. ATAC-seq), global transcriptional and translational activities (e.g. RNA-seq) and 3-D organisation of chromatin (e.g. Hi-C), giving the possibility to study same individual or experimental condition from many different points of view (transcriptomics, epigenomics, etc.) with a very high resolution. Each type of these “omics” data explains a different aspect of cellular behaviour. To give a comprehensive view of the cell regulatory mechanisms it is necessary not only to perform a single level analysis but also to provide novel statistical and computational models for integrating different omic types within a unified study.

Here we present some basics of sequencing techniques that will be addressed during the next chapters.

1.2.1 RNA-seq

Since the beginning of modern biology, gene expression is part of the central dogma of molecular biology. Of course, during the decades some aspects have changed, but the RNA transcripts still have very high relevance. Nowadays, *RNA-seq* [2–5] 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 [6].

The 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* experiment can be viewed as the conversion of long messengers RNA segments in Complementary DNA (cDNA) fragments with RNA or 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).

Afterwards, the so-obtained reads need to be analyzed with several tools, depending on the particular question the researcher is interested in [7, 8].

1.2. SEQUENCING TECHNIQUES

19

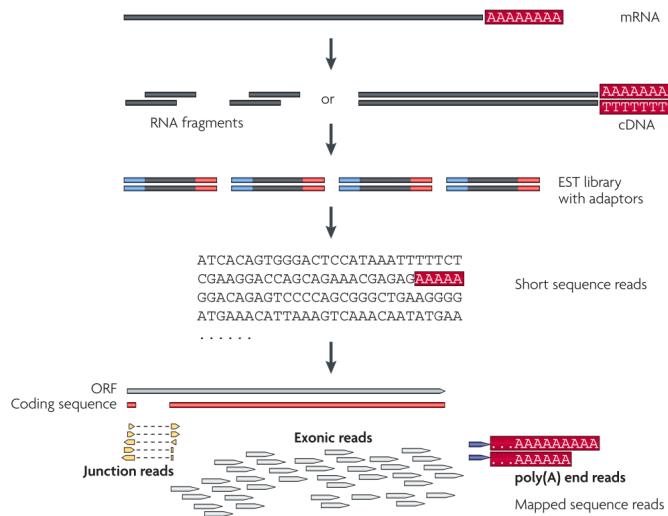


Figure 1.2.1: Representation of an RNA-seq experiment. [3]

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 species’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 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.

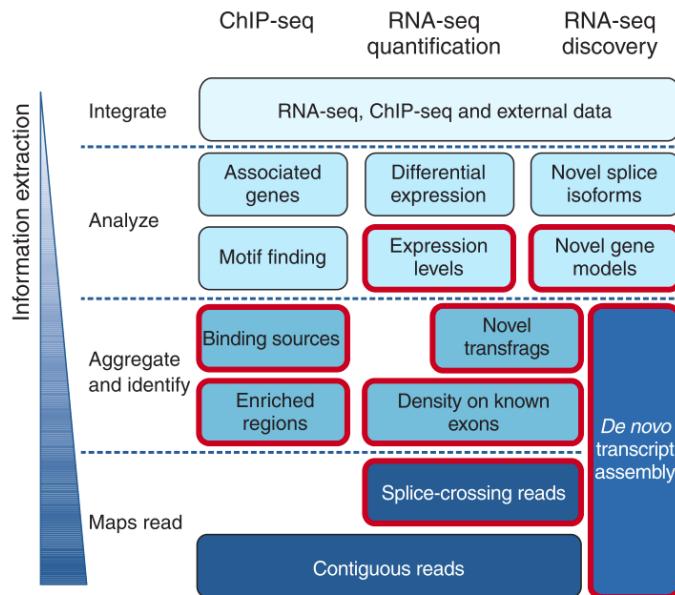


Figure 1.2.2: Representation of RNA-seq analysis complexity. [7]

1.2.2 ChIP-seq

As previously mentioned, epigenetic marks are fundamental aspects of the cellular biological processes. Indeed, their state influences not only gene accession but also the regulation of gene expression.

Some relevant aspects are Histone Modifications (HMs) and Transcription Factors (TFs), the first ones are related to transcriptional activation/inactivation, chromosome packaging, and DNA damage/repair. While TFs, also named as sequence-specific DNA-binding factor, are proteins that control the transcription of genetic information by binding specific sequences of DNA.

To investigate these epigenetic aspects nowadays is mainly used the Chromatin ImmunoPrecipitation sequencing (*ChIP-seq*) technique [9]. This technique allows to use antibodies for selected proteins or nucleosomes, which enriches for specific DNA sequences bounded to these proteins/nucleosomes.

1.2. SEQUENCING TECHNIQUES

21

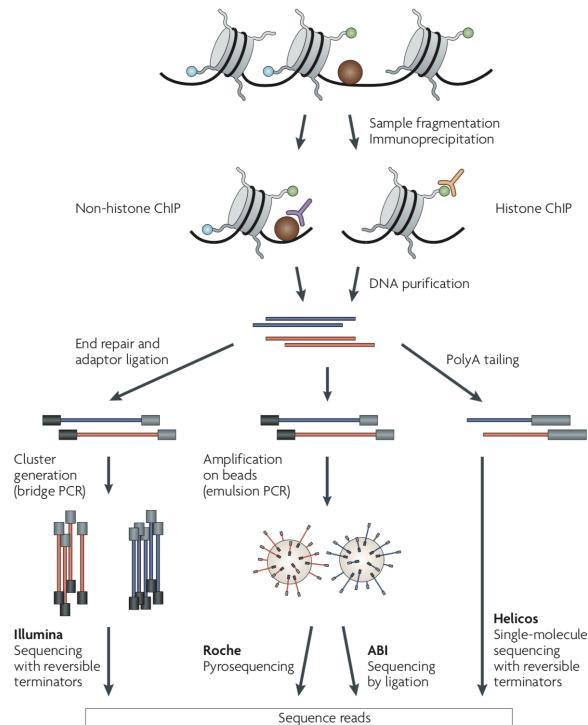


Figure 1.2.3: Representation of a ChIP-seq experiment. [9]

The library preparation consists into lock biological processes with formaldehyde and then cutting the chromatin into small fragments with sonication. Afterwards, a specific antibody for the interested protein is used to immunoprecipitate the DNA-protein complex, in order to be purified and, after amplification, be sequenced.

It is relevant to distinguish between HM and non-HM ChIP because, even if the library preparation it's the same (it differs only for the antibody used, just because the proteins are different), the data analysis pipeline is different in methodologies used because of the different signal produced by them. Indeed, after read mapping on a reference genome, reads need to be processed with methodologies for the protein-binding regions detection, that are typically called

Peak callers. After the peak calling process it is possible to highlight that the TF signals (peaks) are more narrowed respect to the ones detected for HM, leading to develop different methodologies for investigating further aspects for each one of these *ChIP-seq* signals.

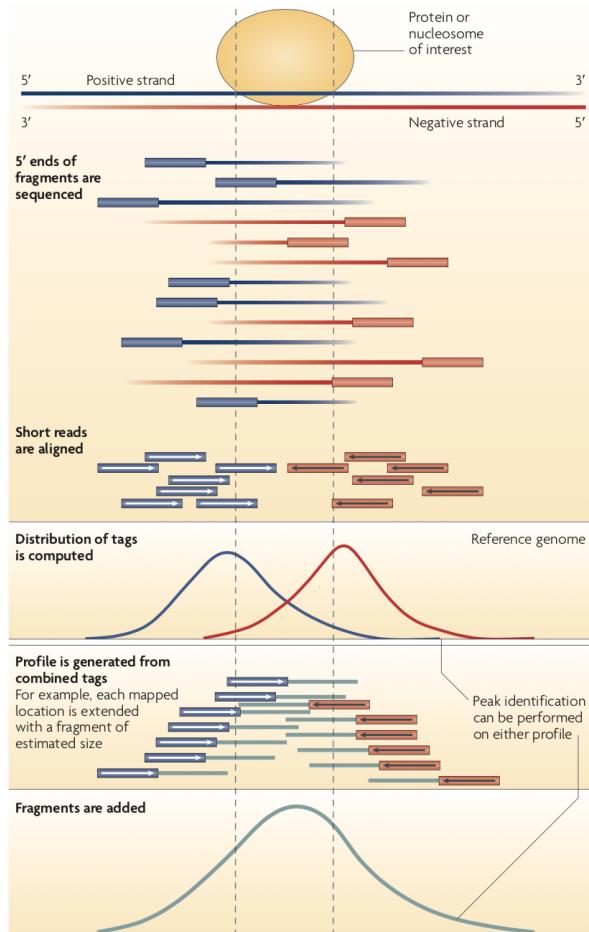


Figure 1.2.4: Representation of a ChIP-seq peak calling process. [9]

1.2. SEQUENCING TECHNIQUES

23

Subsequently to the peak detection, several aspects can be investigated about the signal (figure 1.2.2) such as the identification of motifs related to the peaks, or the genes associated with the regions detected, and, when other omics are available (e.g. RNA-seq), it is interesting to associate the expressed features (e.g. genes) and, consequently, doing functional enrichment analysis.

1.2.3 ATAC-seq

The chromatin packaging of the genome plays a fundamental role in the gene regulation of eukaryotic individuals. To study this aspect of the DNA several technologies have been developed, such as *FAIRE-seq* [10], *DNase-seq* [11] and *ATAC-seq* [12], 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.

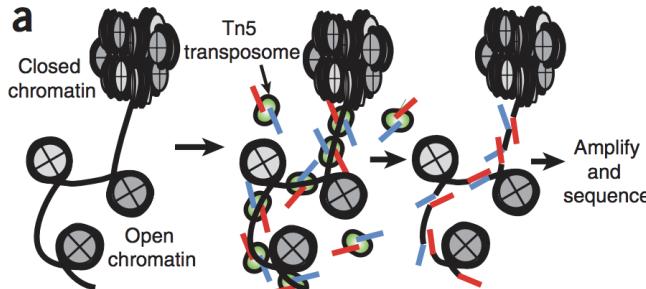


Figure 1.2.5: Representation of ATAC-seq library preparation. [12]

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 exploits 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 stand-

ard 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 have been detected, etc.

To obtain the 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 the second level of integration.

1.2.4 Multi-omics Integration

All those sequencing techniques are aimed to investigate only one cellular compartment at a time, but in order to obtain a more comprehensive view of the cellular behaviour, it is necessary to look at more than one -omic at the same time.

Indeed, we can imagine each omic as a camera in a multi-view camera system pointed on a building from different points of view. Each device takes snapshots of the building from different angles, but the information is still fragmented. In order to reconstruct a 3D model of the building, we need to put the snapshots took from each device together.

The same idea can be adapted to the sequencing techniques, we need to integrate information coming from different omics in order reconstruct (and understand) how multiple mechanisms orchestrate the cellular behaviour. As figure 1.2.8 outline, multi-omics data integration can be made at different levels, by graphical exploration, by functional annotation, by network fusion and by dimensionality reduction.

With graphical exploration, we can visualize data coming from different

1.2. SEQUENCING TECHNIQUES

25

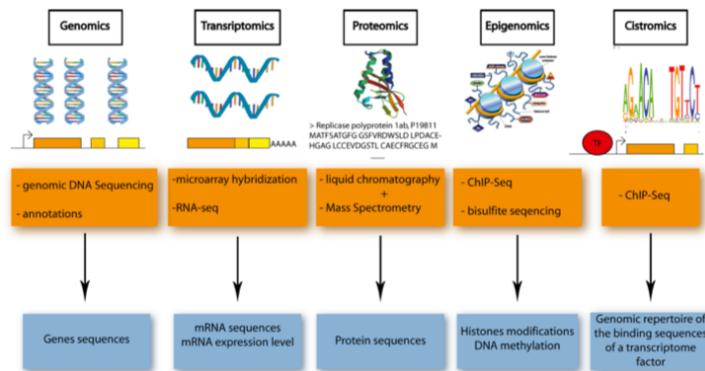


Figure 1.2.6: A scheme of different omics with their related sequencing techniques.



Figure 1.2.7: An illustrative example of a multi-view camera system pointed on a building.

sources (e.g. *RNA-seq* and *ChIP-seq*) using specific tools designed at this scope, such as *Genome Browser* [13] or Integrative Genomics Viewer (IGV) [14, 15] and looking to the overlapping regions, or where expressed epigenomic markers have correspondence with gene expression sites.

We refer to functional annotation integration when using methods combining analysis results (such as relevant lists of genes) with publicly available databases, like the Gene Ontology¹ and pathway (*KEGG*² or *Reactome*³) based ones, to

¹<http://www.geneontology.org/>

²<https://www.genome.jp/kegg/>

³<https://reactome.org/>

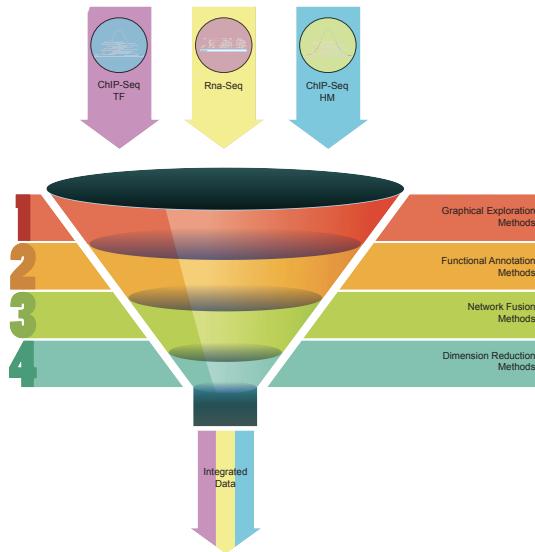


Figure 1.2.8: A schematic representation of multi-omic data integration levels.

detect functional responses highly related to the experimental condition under investigation.

It is possible to integrate multiple omics data types by constructing multiple networks, one for each omic analyzed, and then combine these networks with fusion techniques. This integration aspect is used when a high number of data samples is available or when having multiple-omics data types coming from the same patients [16].

A deeper level of integration is achievable with dimensionality reduction techniques. Also in this case, several samples are needed to be able to obtain relevant and reliable results. Generally speaking, these methods are able to start from multiple samples coming from different omics and to reduce their dimensionality, enabling to identify common cellular behavioural aspects [17, 18].

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 set up for differential expression between two or more conditions. With aid of *edgeR*, *DESeq2* the tool offers the possibility 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 possibility of several interactive plots and graphics, such as KEGG-maps and hierarchical GO terms trees.

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 the differential expression of genes at the 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 [19, 20] terms and Pathways).

Overall, *ticorser* offers the possibility to analyze data using different R-/Bioconductor [21] 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 [22] 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.

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 [23] influencing final results.

In order to address this task, *ticorser* offers multiple statistical methods for

2.2. METHODS

29

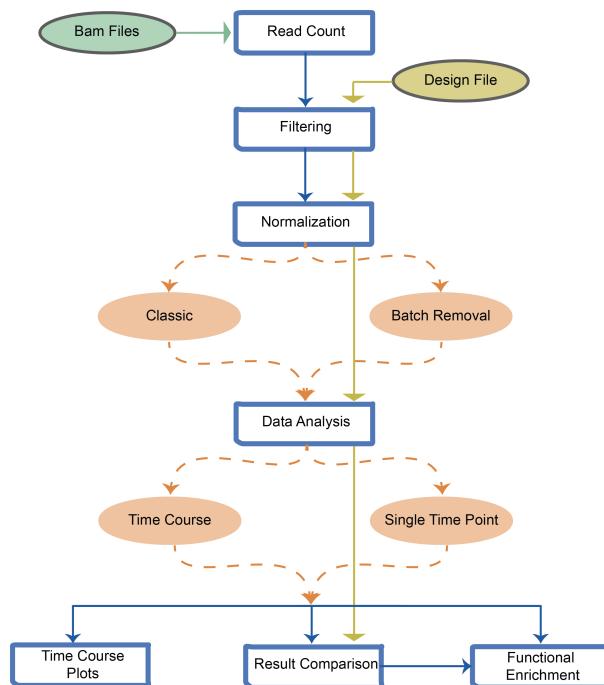


Figure 2.2.1: Main flow of ticorser R package.

low expressed genes filtering. The `filterLowCounts` method is partially based on the `filtered.data` function of *NOISEq* R/Bioconductor package [24]. 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 than 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, where the user doesn't have to take care of any 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 [25], focusing 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 negative controls, giving better results when their estimation is approximated. The method takes 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 three different ways of analyzing time course RNA-Seq data.

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

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

To do so, we took advantage of some of the mostly used and well-performing [26] R/Bioconductor packages, *DESeq2* [27]; *MASigPro* [28]; *edgeR* [29]; *NOISeq* [24].

All the selected methods model the *RNA-seq* data counts for each gene as independent Negative Binomial distributions, which has been demonstrated [30] to be better suited for this data type. At the same time, each of them differs for the statistical test implemented, while approaching the biological question under investigation.

In the following sections, we first 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 between the conditions across all the time points.

Exploiting the LRT, as implemented in *DESeq2* R/Bioconductor 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.

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

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:

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

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:

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

Single DE Methods

To account for fixed time point experiment we implemented functionalities for helping also the exploration 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 the authors suggest.

The `NOISeq` package offers the possibility to discriminate between *biological* and *technical* replicates, computing a posterior probability in both cases, but applying different hypothesis tests.

2.2.4 Data Visualization

During *RNA-seq* data analysis, it is almost mandatory to explore the 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 to be used at each analysis step.

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

Exploration Plots on Counts

As already mentioned (section 2.2.2), the normalization phase critically affects the final results in terms of DEGs detected. That’s why it is a good norm to always understand how the applied transformations are influencing the counts. To address this need we implemented two plots, the *Boxplot* and the Principal Component Analysis (PCA).

The *boxplot* is a graphical representation of the distribution of the samples. Our *boxplot* is organized in group colours accordingly to the time-point each sample belongs to (figure 2.2.2).

2. TIME COURSE RNA-SEQ ANALYZER

TICORSER

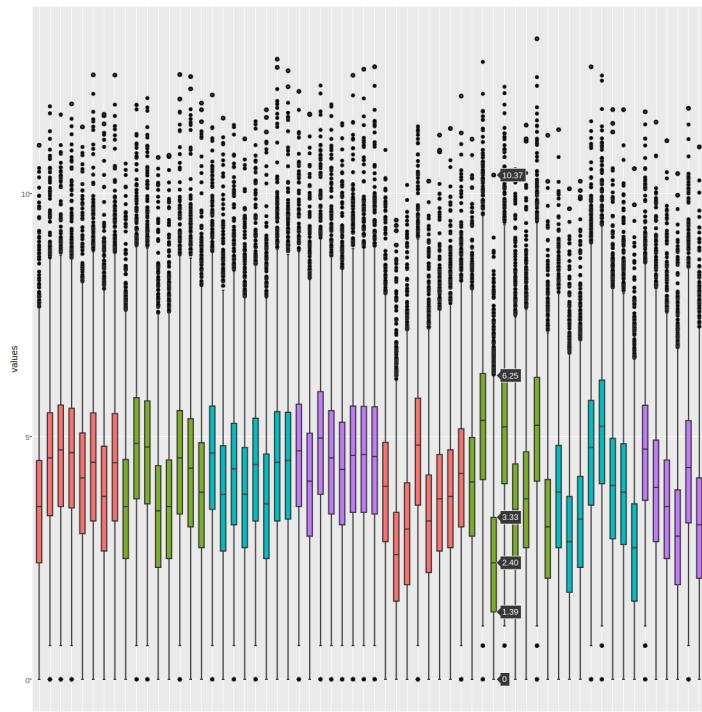


Figure 2.2.2: An example of interactive boxplot made with *ticorser* package. Each boxplot represents a specific sample, while each colour represents a specific time point. When passing the mouse over a boxplot it shows additional information about its quartiles.

Each box is divided into two main parts, with an outgoing segment from each side. The horizontal upper and lower lines 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.

Our `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.

2.2. METHODS

35

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.

2. TIME COURSE RNA-SEQ ANALYZER

TICORSER

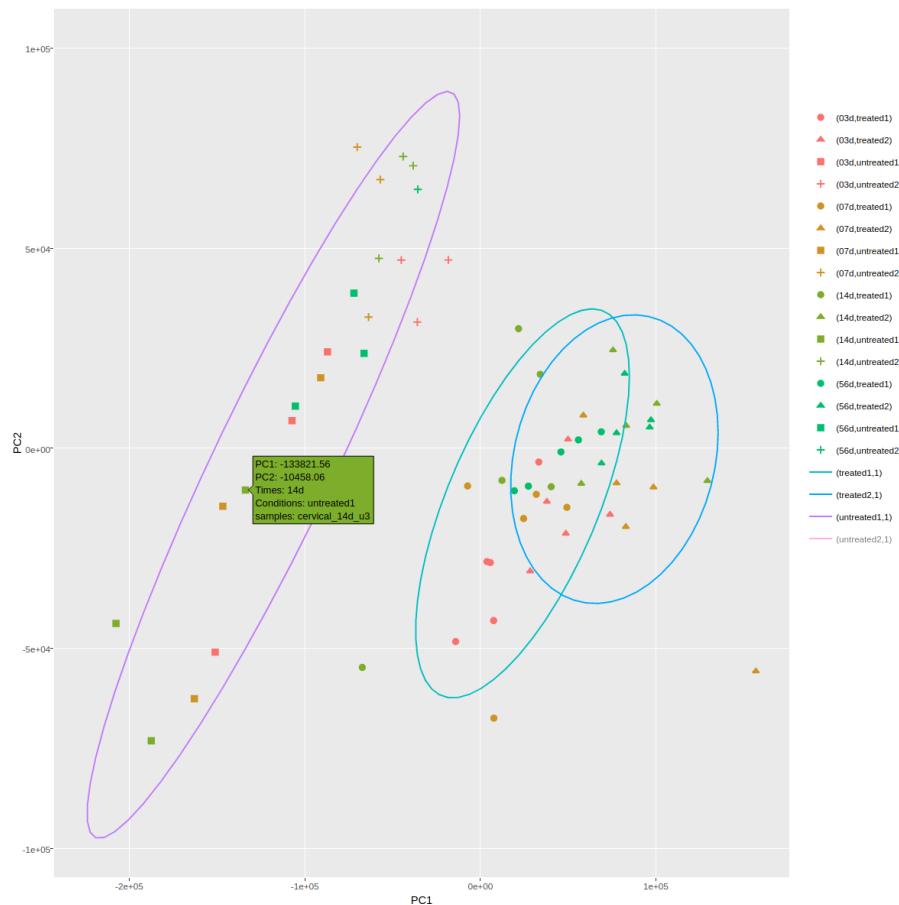


Figure 2.2.3: An example of interactive PCA made with *ticorser* package. Each dot represents a specific sample, while each colour represents a time point, each symbol represents a biological condition group. When passing the mouse over a dot it shows additional information about the selected sample,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

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, colouring the samples according

2.2. METHODS

37

to the groups identified by the chosen column. It is also possible to specify the *Principal Components* (PCs) to plot with `xPCA` and `yPCA`, where x and y indicate respectively the x-axis and y-axis. Moreover, by setting the `ellipse.flag` argument to `TRUE` the function will show the ellipses surrounding each sample group, each one describing the variance of the groups.

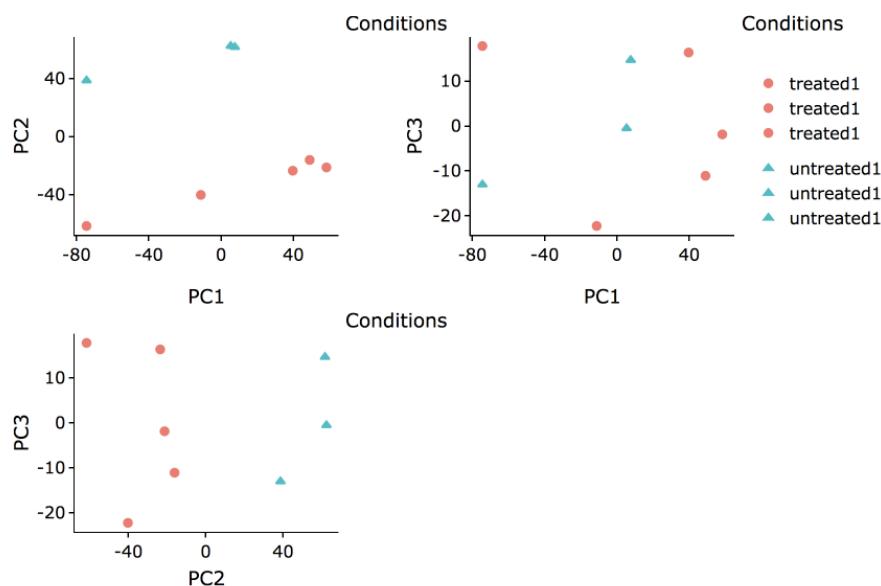


Figure 2.2.4: An example of interactive matrix PCA made with *ticorser* package. Each plot puts in relation two different Principal Components. While each dot represents a specific sample, while each colour represents a time point, each symbol represents a biological condition group. When passing the 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.

In order to plot more than one PCs, *ticorser* allows to plot more than one comparison between them at the same time (figure 2.2.4). By using the `plotPCAMatrixPlotly` function it's possible to compare more than two PCs at the same time, simply by using the `pca.list` argument.

DE Results Plots

In order to inspect the results produced by so many different *DE* methods, we implemented two kind of plots, the *VolcanoPlot* and the *MAPlot*.

Both our implemented methods take as input a *DE* results data frame, automatically recognizing which method produced it, lightning, in such a way, the user load during the analysis. Moreover, it gives the possibility to add a list of positive control genes, in order to annotate them on the volcano plot with a third colour (figure 2.2.5).

The volcano plot shows the relation of $\log_2(FC)$ with $\log_{10}(p\text{-value})$ of each gene, in order to highlight the significant changes inside the data experiment.

2.2. METHODS

39

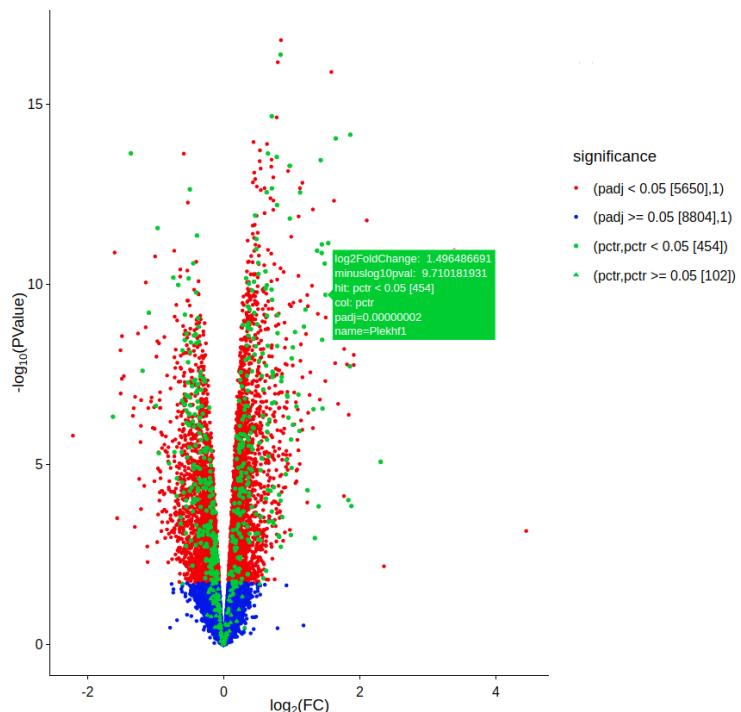


Figure 2.2.5: An example of interactive volcano plot made with *ticorser* package. Each dot represents a gene, while blue and red colours highlight the significance of the genes. In green, there are those genes coming from the positive control list. When passing the mouse over a dot it shows additional information about the selected gene.

The MA-Plot shows the relation between the two quantities useful to understand the differences between the measurements in two conditions. On the x-axis, there is represented the $\log_2(\text{FC})$, 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 dataframe to plot an MA-plot, but it's pretty useful to have it in order to highlight the distribution of the significant genes (figure 2.2.6).

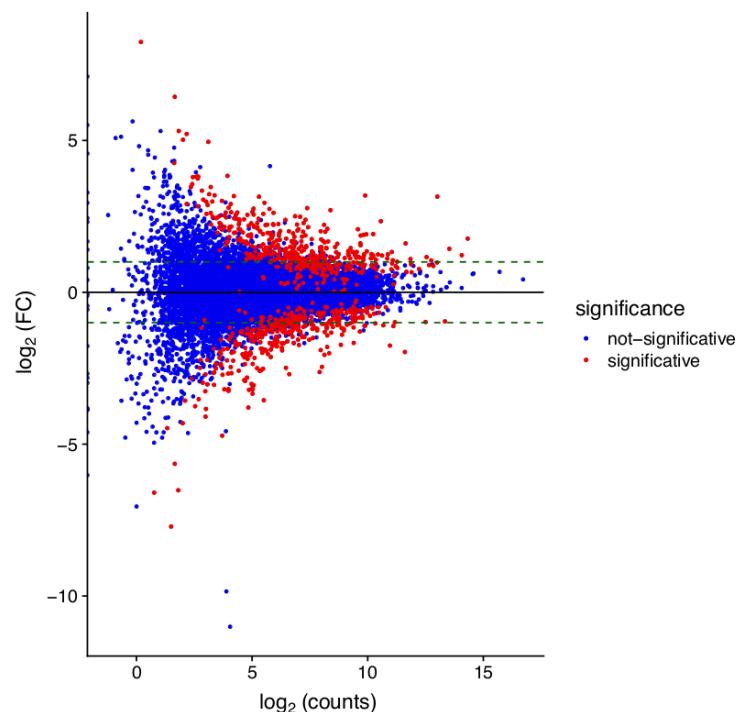


Figure 2.2.6: An example of interactive MA-plot made with *ticorser* package. Each dot represents a gene, while blue and red colours highlights the significance of the genes. When passing the mouse over a dot it shows additional information about the selected gene.

Gene Profiles plot

When working with time course data experiment, it is really useful to understand the trend of one or more genes across all the time points and of one condition in relation to the other.

In order to highlight the gene expression of a gene across multiple time points and different conditions, we implemented the `plotGeneProfile` function, which takes as input a count matrix, its linked design matrix and a gene name (figure

2.2. METHODS

41

2.4.7).

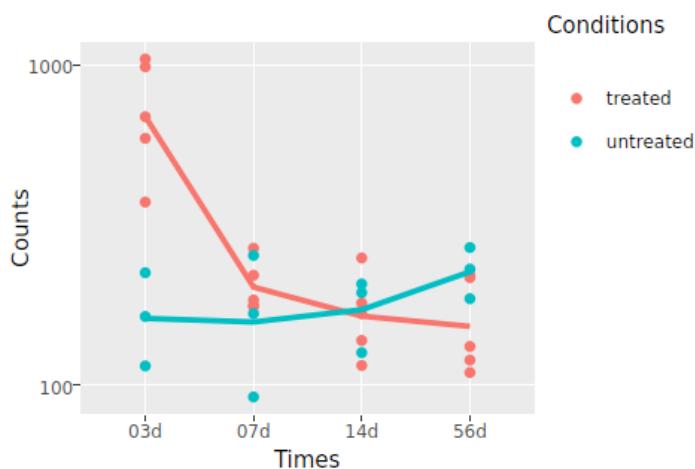


Figure 2.2.7: An example of interactive gene profile made with *ticorser* package. Each dot represents the counts value of the gene in a sample. Colours identifies the conditions of the samples. The lines represent the gene trend over all time points.

keggmap

Once detected DEGs it is really important to look for functional mechanisms regulated by up-regulated and down-regulated genes.

ticorser offers the possibility to plot *keggmaps*[31] taking into account the $\log_2(FC)$ of the genes involved in the graphical representation, through all the time points. Indeed, using the `plotKeggmap` function enables to plot a *keggmap*, using as input the counts and the design matrix, computing the $\log_2(FC)$ at each time point, and showing it in the gene box inside the *KEGG* map.

2. TIME COURSE RNA-SEQ ANALYZER

42

TICORSER

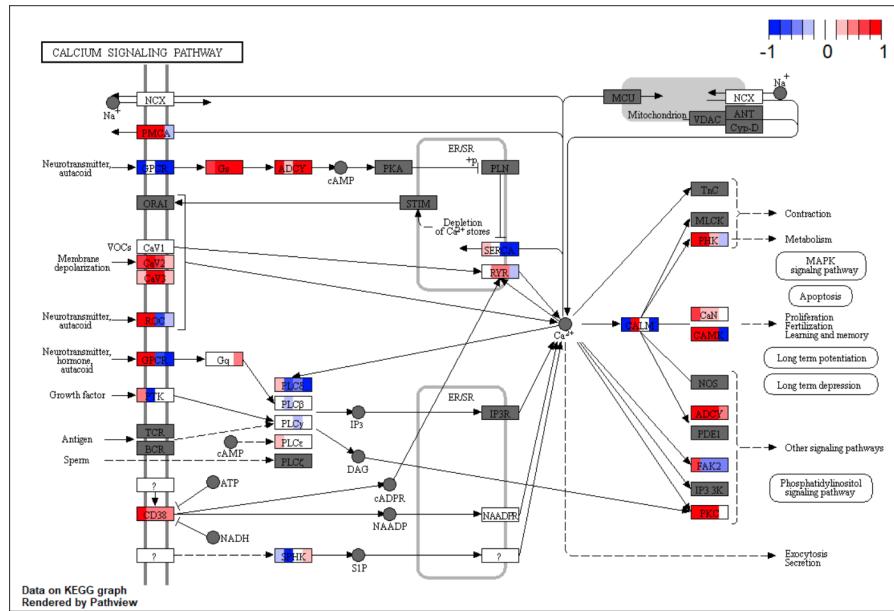


Figure 2.2.8: add description

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 [32], 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, `Venn2de`, `Venn3de` and `Venn4de`, respectively for two, three and four lists. 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 store output file lists 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` [33] for *Mouse* and using the `biomaRt` R/Bioconductor package 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 needs 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 composed of 62 samples of bulk *RNA-seq*, divided into groups of two different tissues at four different time points, with treatments and controls at each time point.

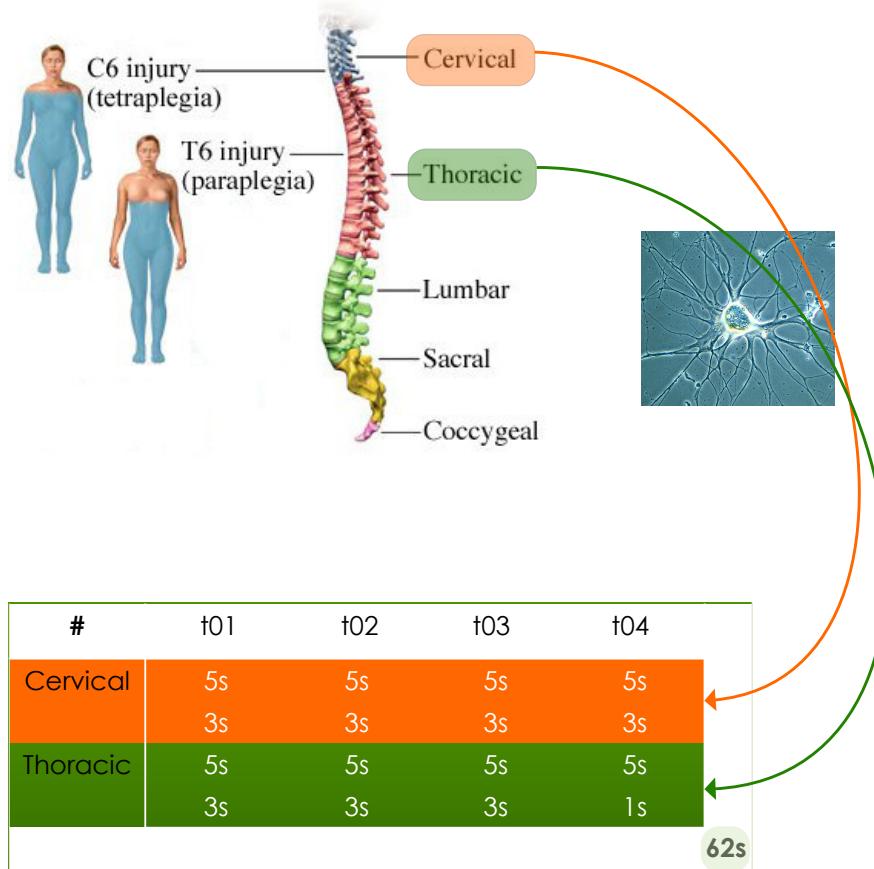
2.4. CASE STUDY**45**

Figure 2.4.1: An illustrative example of the used dataset. Neurons of Cervical and Thoracic spinal cord injury, have been extracted and sequenced at 4 different time points.

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 an illustrative example (figure 2.4.1 shows the dataset with most relevant details masked).

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 `countBamFilesFeatureCounts` method with the path of the *BAM* files and a GTF¹ 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. For this reason, we suggest to always use the latest version of the GTF of the genome for the under investigation species 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 a discrete value indicating the number of reads quantified for the feature on the row in the column of the sample.

The design matrix

From this point afterwards, *ticorser* requires a design file illustrating the descriptive 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 or the rownames 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.

¹<https://www.ensembl.org/info/website/upload/gff.html>

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

2.4. CASE STUDY

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, it is a good practice to normalize data, a step which affects the DEGs detection[34–36]. But before normalizing, it might also be useful to filter out low expressed features, since they are not relevant and may lead to biased results introducing unwanted noise

2. TIME COURSE RNA-SEQ ANALYZER

TICORSER

in the analysis.

Figure 2.4.2 shows the effects of filtering step on row counts (in upper left corner). What emerges from this comparison is that low expressed features are in a high amount in raw data, while it is quite un-important the method used for filtering them.

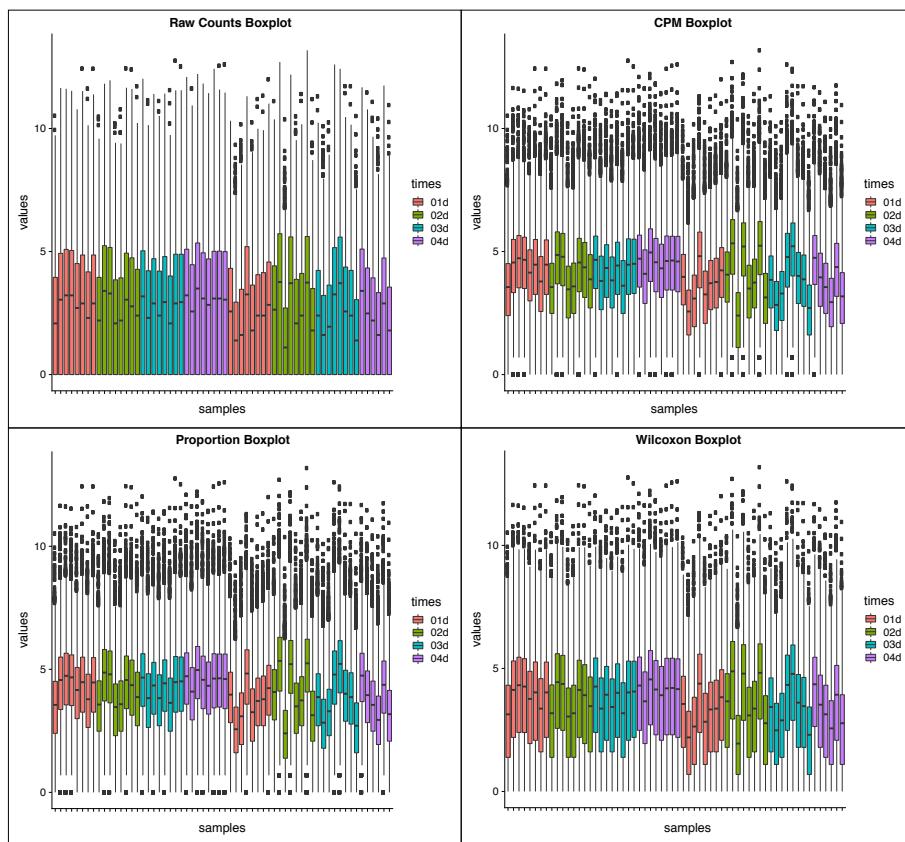


Figure 2.4.2: A comparison of how the filtering methods available in *ticorser* affect the data. The first panel shows the row counts with low expressed genes, while the other boxes show the filtering effect on the samples.

Moreover, looking at the *Wilcoxon* test, it is more conservative, because it

2.4. CASE STUDY

49

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 2.4.3, but also the PCA representation, enabling us to understand which normalization better remove biases from the samples by clustering the samples of the same group (figure 2.4.4).

2. TIME COURSE RNA-SEQ ANALYZER

TICORSER

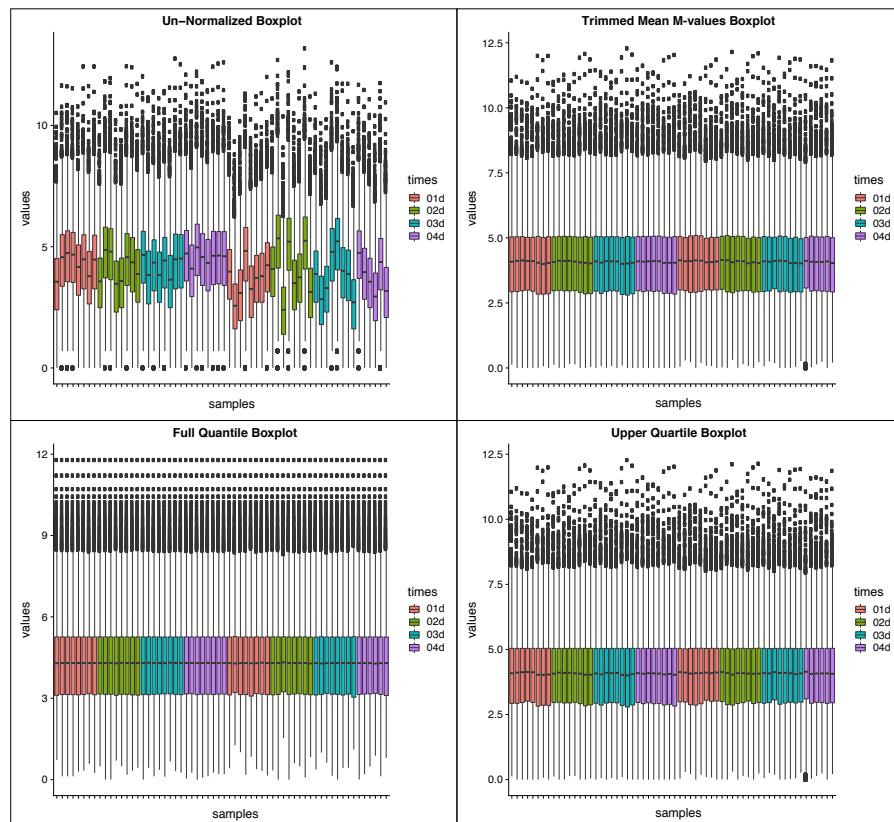


Figure 2.4.3: A comparison of how the filtering methods available in *ticorser* affect the data. The first panel shows the row counts with low expressed genes, while the other boxes show the filtering effect on the samples.

Figure 2.4.3 perfectly shows different effects of each normalization type on count data. In fact, we can see that while *Upper quartile* aligns all the samples on the third percentile leaving the medians not aligned, the *Full quantile* applies a stronger effect, which 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

2.4. CASE STUDY

51

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

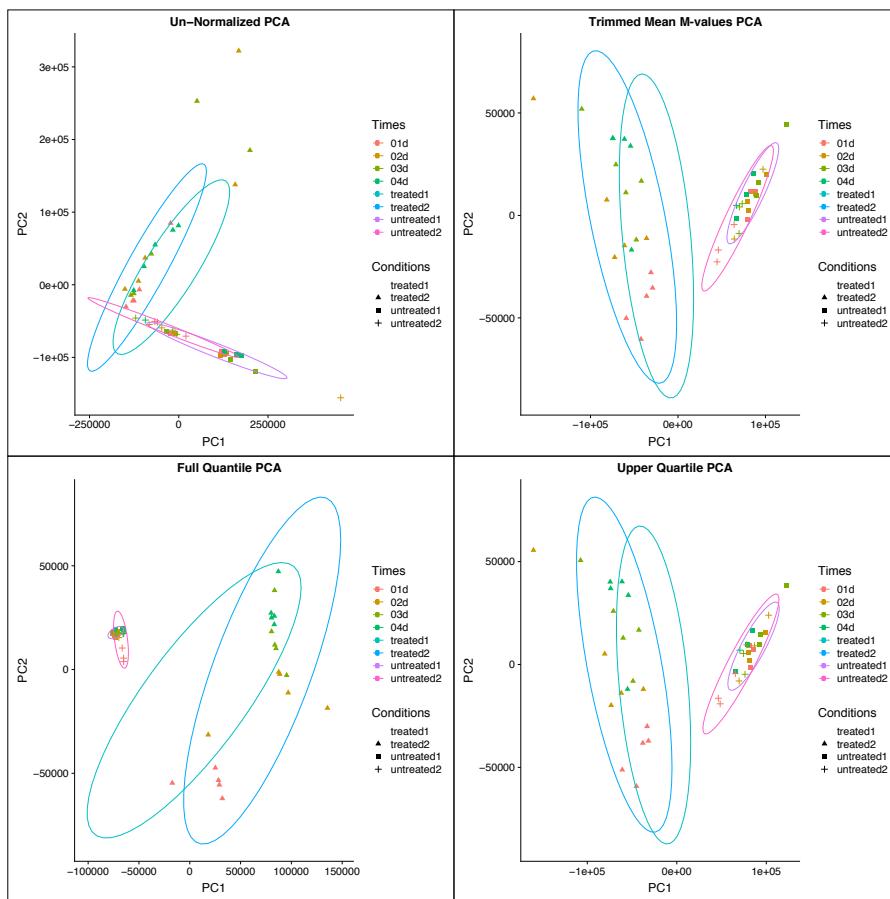


Figure 2.4.4: A comparison of how the normalization methods available in *ticorser* affect the data. The first panel shows the clustering of row samples, while the other boxes show the normalization effect on the samples.

As already mentioned a boxplot data inspection could not be enough to determine which normalization method is better working on the data. Indeed,

when looking at *PCAs* for normalized counts (figure 2.4.4 the *TMM* and the *Upper quartile* normalizations are able to well discriminate treated and untreated samples, but still preserving some variability inside each group. While *Full quantile*, even if it is able to help to discriminate the treated and untreated groups, it smashes too much the variability inside 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 interesting 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 change 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 previously 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.

2.4. CASE STUDY

53

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 2.4.5 shows.

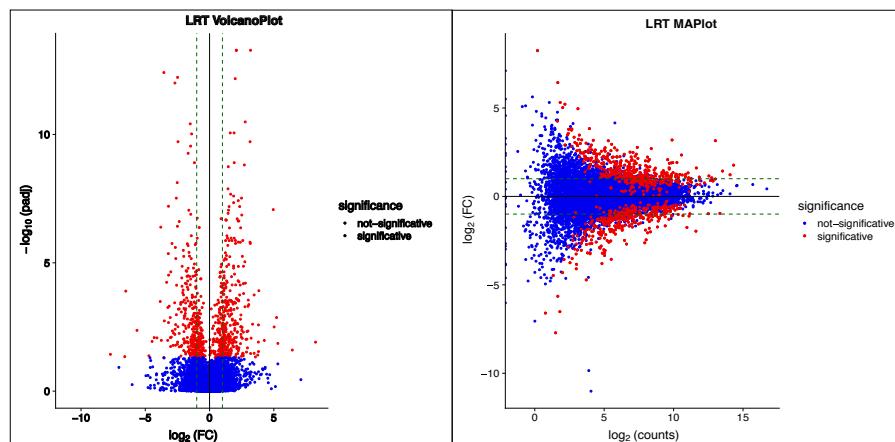


Figure 2.4.5

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).

Venn Diagram

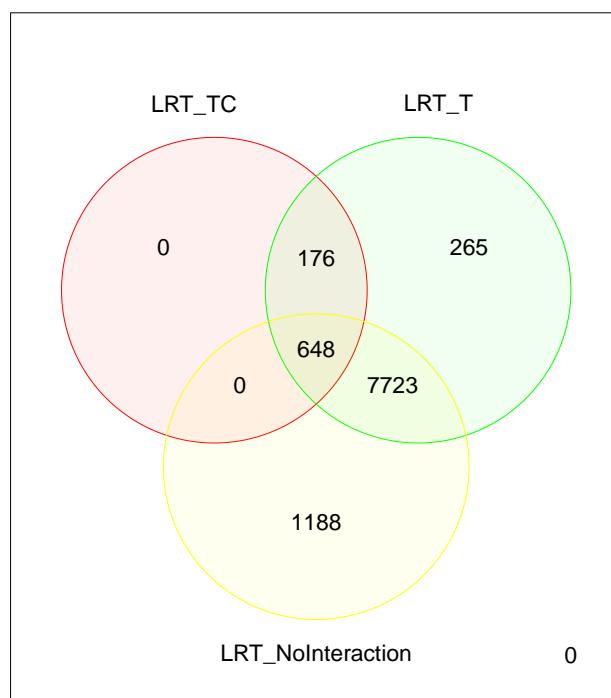


Figure 2.4.6

2.4. CASE STUDY

55

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 expression 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.

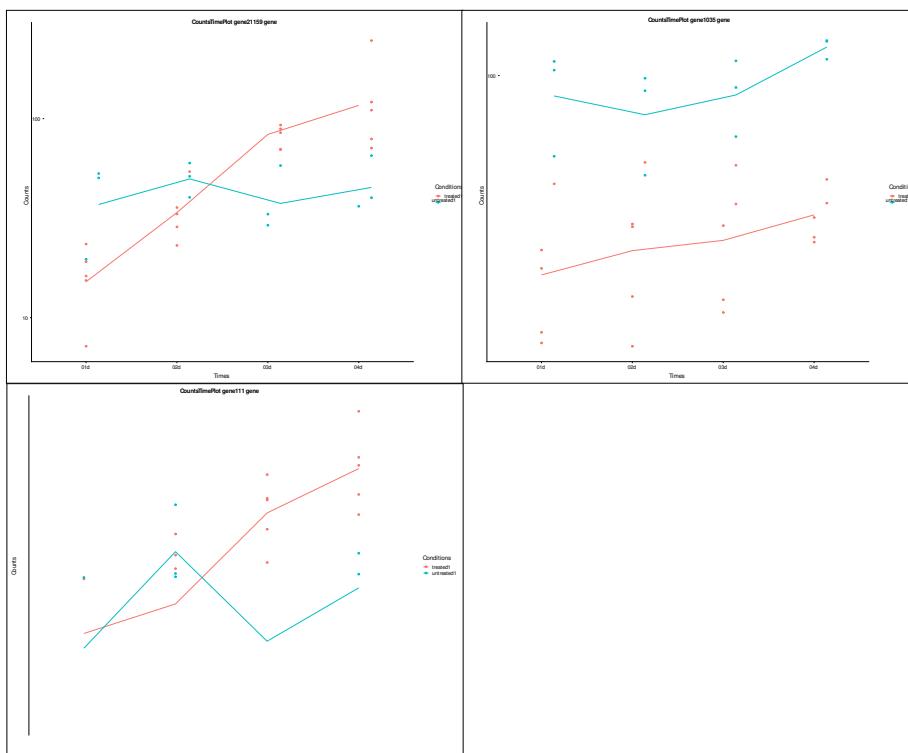


Figure 2.4.7: Three different trends examples of genes detected from different TC methods.

For example looking at figure 2.4.7 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, otherwise, the second gene, detected with `DeSeqTime_T` method, in the right upper corner, shows a difference between the conditions that remain unchanged across the experiment. Finally, using the `DeSeqTime_NoInteraction` method, we detected a gene with a strange behaviour, 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 `ApplyNoisep` 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 about 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 2.4.8 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.

2.4. CASE STUDY

57

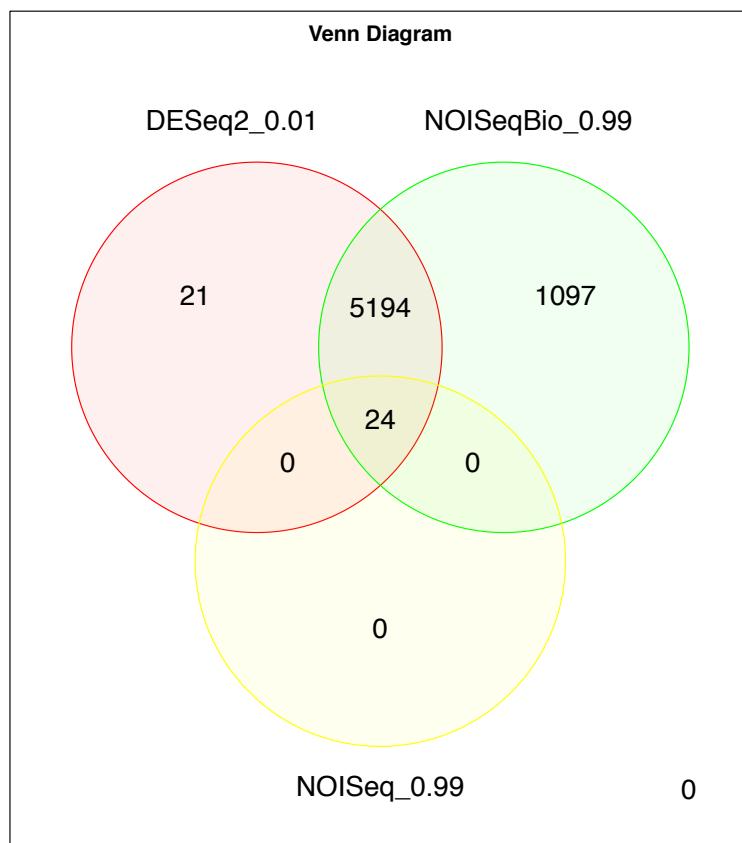


Figure 2.4.8: A venn diagram for comparing results coming from different methods used for DEG detection.

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

2.4. CASE STUDY

59

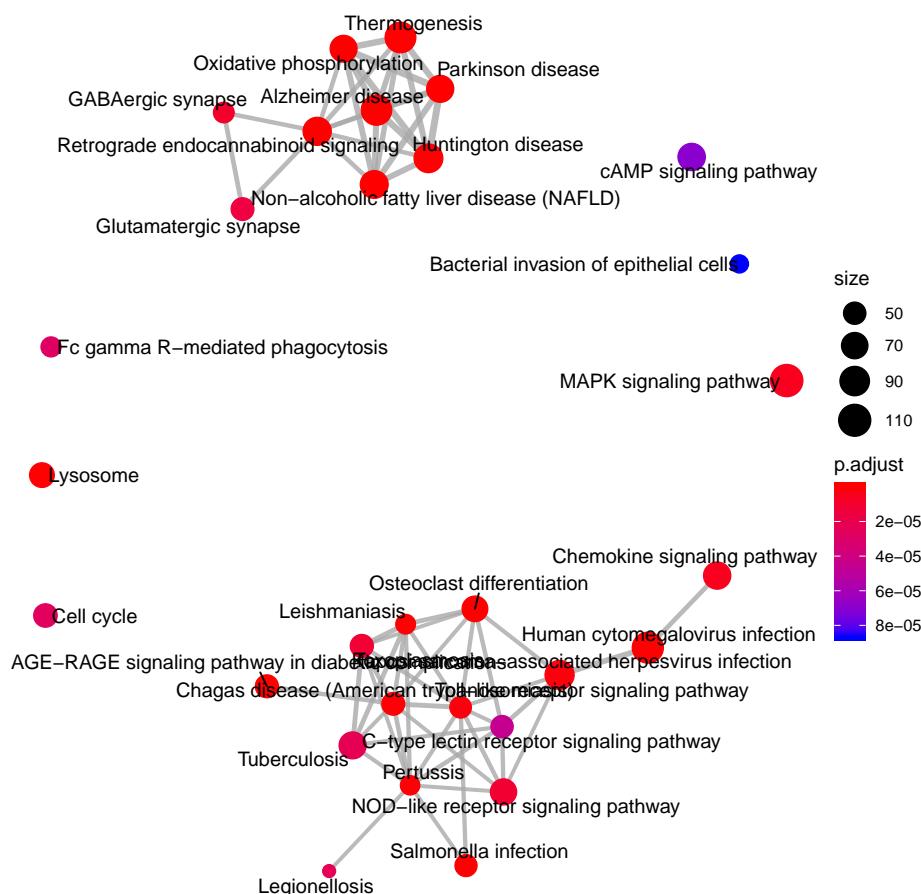
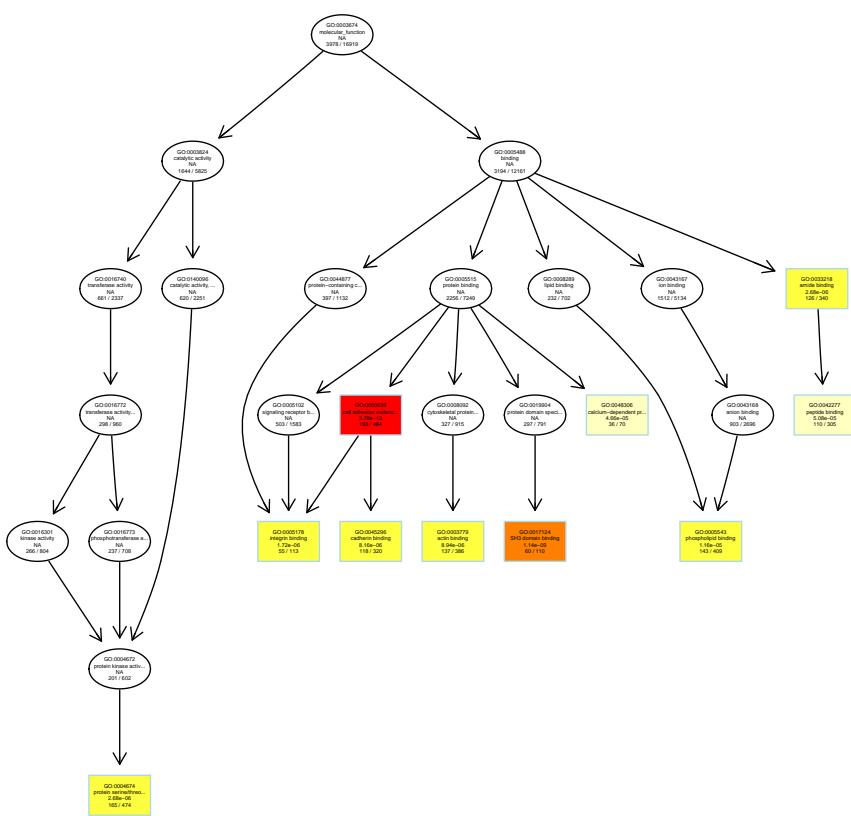


Figure 2.4.9: A network representation of the kegg results obtained with *clusterProfiler* package, by using *ticorser*

On the other hand, to perform a Gene Ontology functional enrichment analysis we use `enrichGOProfiler` 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 `enrichGOF` 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 2.4.10).



Finally, when we identify one or more interesting *KEGG* pathways, we can

2.4. CASE STUDY

61

plot a *KEGG map* representation of it, by using `PlotKeggMapTimeCoursePathview`, which takes as input a data frame of relevant genes expression values for the interested pathway, and the `keggid` identifying the pathway. In such a way a *KEGG map* with the expression values at each time point for those genes will be visualized. Figure 2.4.12 and 2.4.11 graphically shows the *KEGG-maps* of the *Parkinson disease* and *Alzheimer disease* pathways, which both resulted highly significant from our functional analysis, and which are known to be associated to SCI [37, 38].

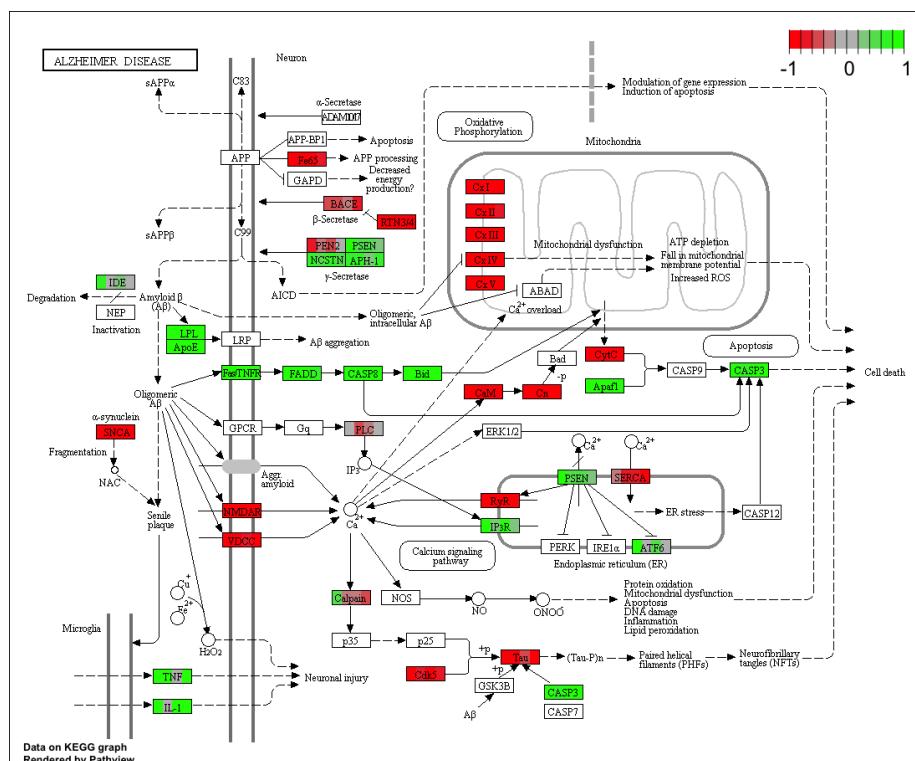


Figure 2.4.11: A keggmap for Alzheimer disease. Each DEG is coloured depending on the log2(fc) of its expression. Moreover, for each highlighted gene, the box is divided in 4 "mini-boxes", one for each time point.

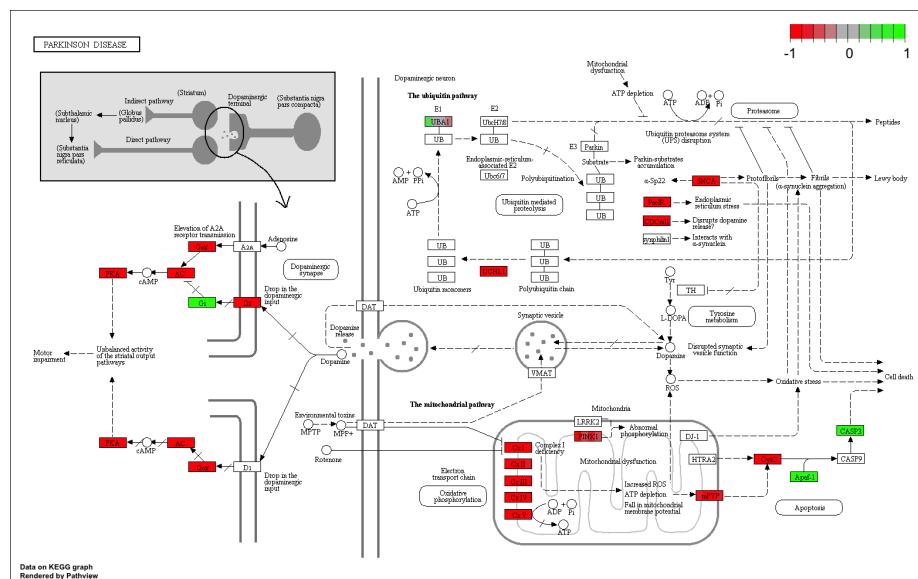


Figure 2.4.12: A keggmap for Parkinson disease. Each DEG is coloured depending on the $\log_2(\text{fc})$ of its expression. Moreover, for each highlighted gene, the box is divided in 4 "mini-boxes", one for each time point.

2.5 Future Works

We presented *ticorser*, an R command line tool for easy and fast analysis of time course *RNA-seq* data, presenting a wide range of methods for differential expression data analysis and their visualization.

While the package allows comparing different methodologies to well discriminate between multiple conditions at a single time point, it'd be a good practice also to compare results coming from multiple methodologies when working with TC data. A good candidate for this aim is the *nextMASigPro* R/Bioconductor package which takes advantage of the Generalized Linear Model (GLM) with Negative Binomial distribution. This method, unlike our implemented ones, allows detecting all the DEGs showing any kind of differences between the conditions across all the time points. And, as suggested by authors is a good norm

2.5. FUTURE WORKS

63

to cluster the genes to better understand which is their singular behaviour.

Additionally, to provide a useful instrument for gene expression inspection, we plan to insert features for time-oriented heatmaps creation and manipulation.

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 [39, 40] 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 a lack of them, in particular for emerging omics as *ATAC-seq*.

To address this need, we decided to create a useful instrument for analysing chromatin 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 [39] 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 [41] 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 [21] data structures and methods, and it is available through the Bioconductor repository since version 3.7.

The tool is organized into 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 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.

3.2. METHODS

67

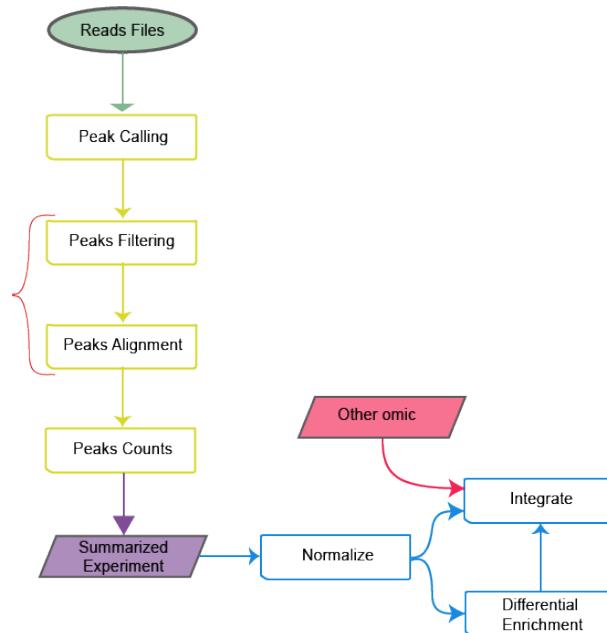


Figure 3.2.1: A differential enrichment flow representation. *DEScan2* steps are highlighted in yellow.

3.2.1 Peak Caller

The Peak Caller (defined by the `findPeaks` function) takes as input a set of alignment files (BAM [42] or BED format) with the code identifier of the reference 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 an object of class *GenomicRangesList* [43], 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 contiguous 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 score and one on the number of samples.

3.2. METHODS

69

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 [44], as described in section 3.2.4.

Firstly, using the threshold on the peaks 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* [45] 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 setting the samples threshold as a multiple 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* [46] 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 *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 evolved 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 [47].

To illustrate the performances of *DEScan2* we chose a dataset [47] that describes *in vivo* adult mouse dentate granule neurons before and after synchronous neuronal activation using *ATAC-seq* and *RNA-seq* technologies (see sections 1.2.3 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.

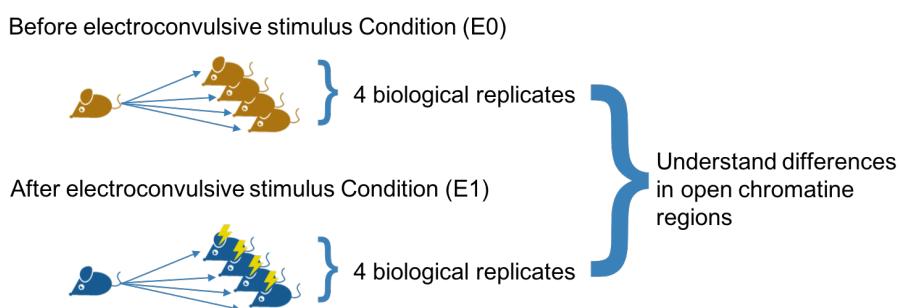


Figure 3.3.1: An illustration of our extraction of the GSE82015[47] dataset.

We downloaded the data from Gene Expression Omnibus (GEO) database [48, 49] with accession number GSE82015¹ and mapped the raw data using *STAR* [50] 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 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* [44] 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 gave 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>

3.3. CASE STUDY

73

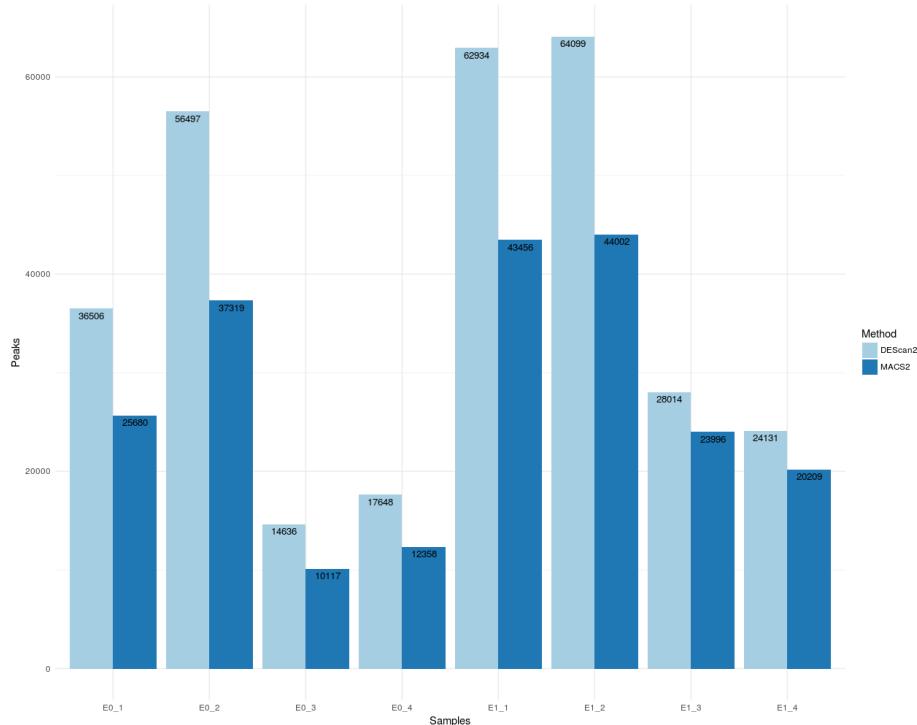


Figure 3.3.2: A comparison of *DEScan2* and *MACS2* detected peaks for each sample in the dataset.

To be more robust, we compared *DEScan2* detected peaks with the same validated regions (*Arc²* and *Gabrr1³*) of the original work [47]. 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>

3. DIFFERENTIAL ENRICHED SCAN 2

DESCAN2

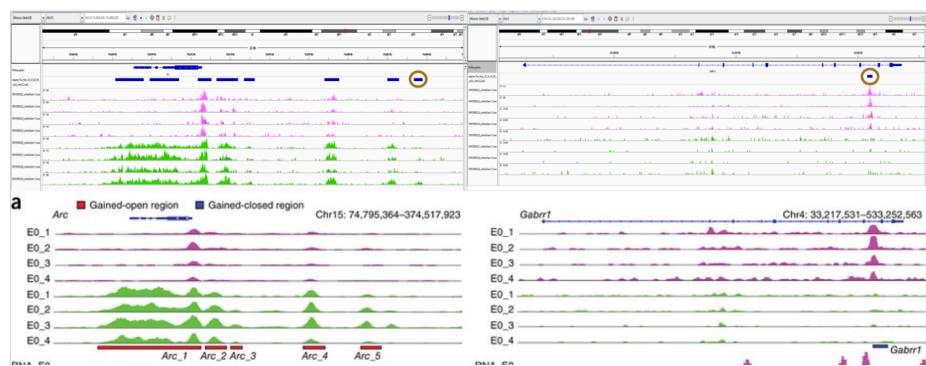


Figure 3.3.3: A comparison of *DEScan2* detected peaks with validated peaks in article [47].

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 is 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.

3.3. CASE STUDY

75

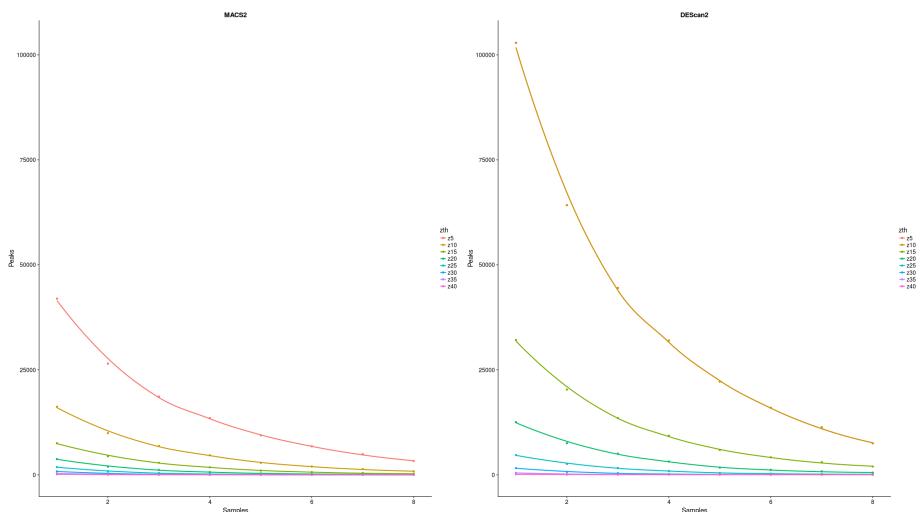


Figure 3.3.4: Filtering the detected regions with different thresholds on peak scores between *MACS2* and *DEScan2*.

Moreover, comparing the 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 a 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 kinds of omics, as *RNA-seq*.

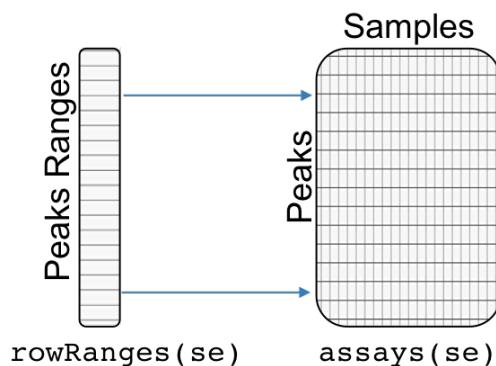


Figure 3.3.5: An illustration of the *SummarizedExperiment* data structure produced by *DEScan2*.

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 technical and biological noise can confound the analysis [34]. 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 [25, 51]. To filter out false positives, but preserving enough signal at the same time, we fixed the peaks 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 kinds of normalizations and combinations of them give good results.

3.3. CASE STUDY

77

Figure 3.3.7 summarizes this concept very well, highlighting a relation between the number of DERs (y-axis) and the minimum 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.

3. DIFFERENTIAL ENRICHED SCAN 2

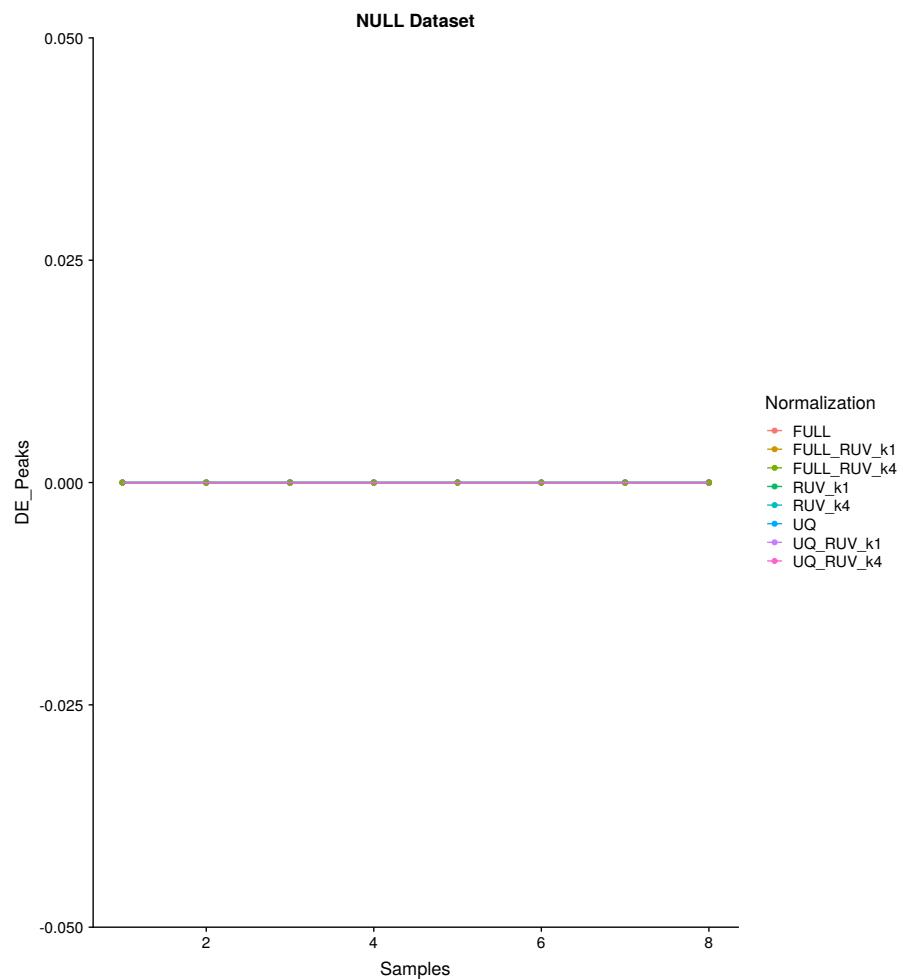
DESCAN2

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

3.3. CASE STUDY

79

confident with our results.

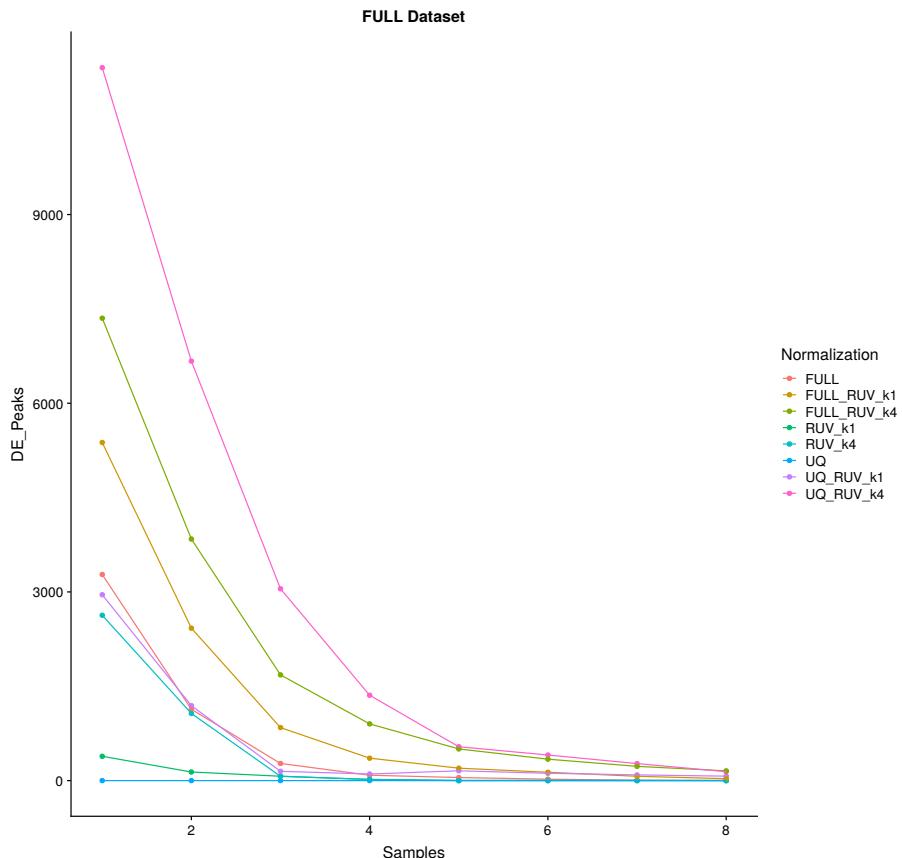


Figure 3.3.7: The figure shows the effects of different normalizations on a dataset of epigenetic regions, putting in relation the DERs with the threshold on the samples used to align peaks.

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 *RUVSeq*, the DERs detection depends on the parameter used (figures 3.3.6, 3.3.7 are made using k equal to 1 and 4, for *RUVSeq* normalization). 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 overestimates 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 epigenomics 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 represents the "null dataset" highlighting that portion of DERs due to randomness/bias. Indeed, any kind 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 [24, 29, 52].

3.3. CASE STUDY

81

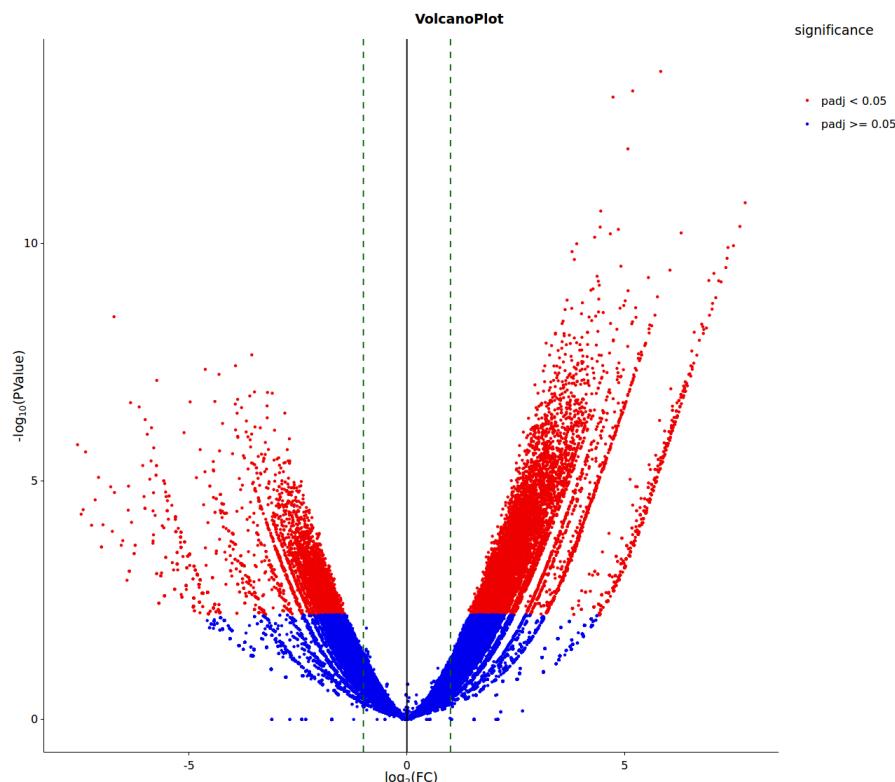


Figure 3.3.8: A volcano plot of Differential Enriched Regions. Blue dots represent the not significant DERs, while the red ones represent the significant DERs.

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 *RUVSeq* normalized counts with *k* parameter set to 4, we modelled 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*[29].

3. DIFFERENTIAL ENRICHED SCAN 2

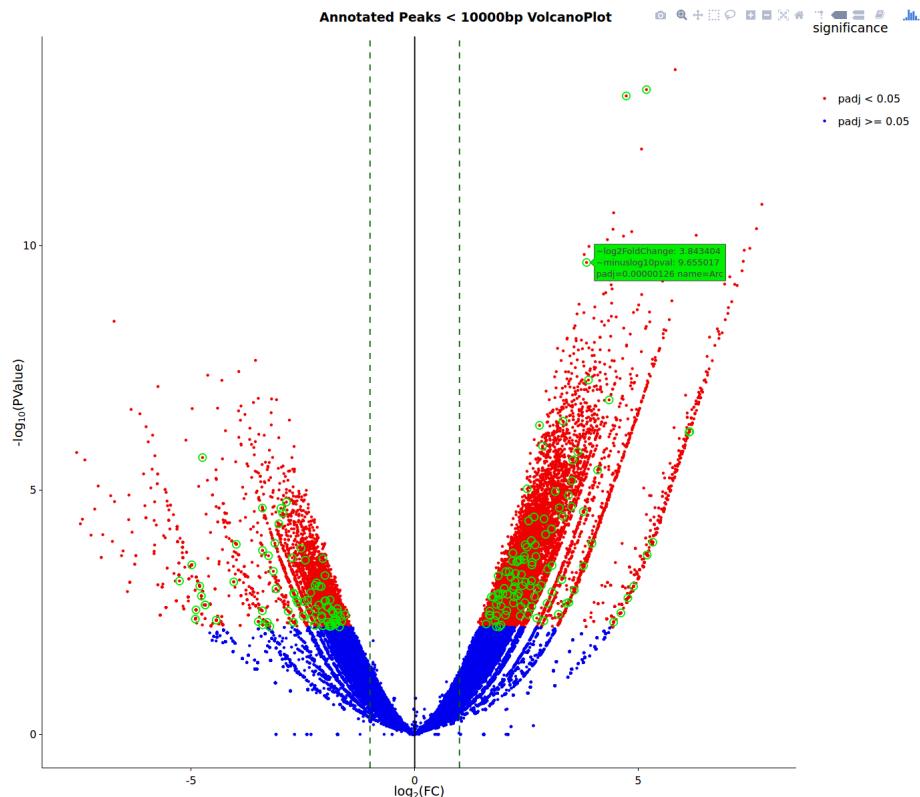
DESCAN2

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)[53] 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

3.4. FUTURE WORKS

83

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 first quantified the signal with the `featureCounts` methods available in the *Rsubread* [32] R/Bioconductor package. Then we filtered lowly expressed genes with the *proportion* test as implemented in *NOISEq* package, and applied the `noisep` 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 *ChIP-peakAnno*. 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, to obtain a seconf level of data integration, we used the annotated genes to do functional annotation on Gene Ontology (GO) [19, 20] and Reactome pathways, which showed several interesting results for the neuronal regulation.

3.4 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 signal producing data structures as *Summarized-Experiment* which are candidates to become standards in the biological data analysis. With our 3-steps analysis, we put 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 an additional probabilistic methodology.

Chapter 4

A reproducible computational research R6 Class *easyReporting*

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

In this Chapter we illustrate *easyReporting* a novel R package for speeding up the RR when analyzing data or when constructing other packages.

4.1 Introduction

During last years several approaches [54] have been proposed for helping to trace the analysis steps, using different programming languages; such as *Jupyter*

4. A REPRODUCIBLE COMPUTATIONAL RESEARCH R6 CLASS 86 EASYREPORTING

¹ [55] in *Python* or *Rmarkdown*² in *R*. Or by building a web environment to encapsulate several tools made with different programming languages, such as *Galaxy*³ [56–58].

The common underlying idea of each one 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 comprehensibility and reproducibility of the work, in a unique final resulting file.

To address this scope the *R* community proposed several solutions, like *sweave* before, *knitr* and *rmarkdown* later. Due to its easy interactive usage, *rmarkdown* became one of the most used instruments in the *R* community, but its usability when developing automated instruments like Graphical User Interface (GUI) or packages becomes more difficult, leading developers to give up using it.

4.2 Methods

Here we present *easyReporting*, an *R6*⁴⁵ class⁶ developers to integrate a reproducible research layer inside their software products, as well as lazy analysts to speed up their report production without learning the *rmarkdown* language.

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

¹<https://jupyter.org/>

²<https://rmarkdown.rstudio.com/>

³<https://usegalaxy.org/>

⁴<https://adv-r.hadley.nz/r6.html>

⁵<https://cran.r-project.org/web/packages/R6/index.html>

⁶[https://en.wikipedia.org/wiki/Class_\(computer_programming\)](https://en.wikipedia.org/wiki/Class_(computer_programming))

4.2. METHODS

87

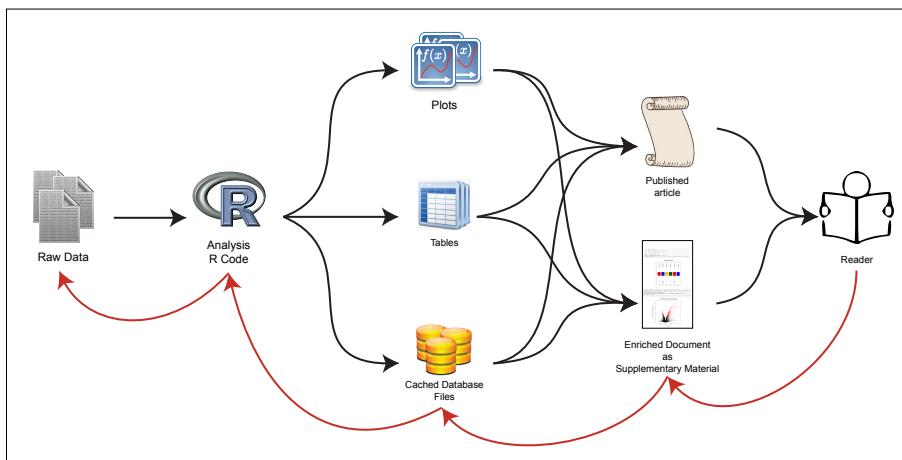


Figure 4.2.1: A general illustration of Reproducible Computational Research concept. Raw data can be analysed with R code, producing plots, tables and additional caching database files. All together, R code, data and results can be inserted in an enriched document, which can be added as supplementary material to a valuable published article.

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 (figure 4.2.1).

The package is accessible at the following link:
<https://github.com/drighelli/easyreporting>

General Description and Initialization

The class can be imagined as a schematic representation of the *rmarkdown* file (*report*), indeed its attributes represent the *report* characteristics, which are typically inserted in the header of the file. But our class methods are not only for the attributes manipulations but also for insertion of CCs, comments and

4. A REPRODUCIBLE COMPUTATIONAL RESEARCH R6 CLASS 88 EASYREPORTING

section titles inside the *report*.

As any typical class, before of using it, *easyReporting* requires to be initialized 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 4.2.2 for a list of available options) 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`.

Chunk options		
option	default value	description
Code evaluation		
<code>child</code>	NULL	A character vector of filenames. Knitr will knit the files and place them into the main document.
<code>code</code>	NULL	Set to R code. Knitr will replace the code in the chunk with the code in the code option.
<code>engine</code>	'R'	Knitr will evaluate the chunk in the named language, e.g. <code>engine = 'python'</code> . Run <code>names(knitr::knit_engines\$get())</code> to see supported languages.
<code>eval</code>	TRUE	If FALSE, knitr will not run the code in the code chunk.
<code>include</code>	TRUE	If FALSE, knitr will run the chunk but not include the chunk in the final document.
<code>purl</code>	TRUE	If FALSE, knitr will not include the chunk when running <code>purl()</code> to extract the source code.
Results		
<code>collapse</code>	FALSE	If TRUE, knitr will collapse all the source and output blocks created by the chunk into a single block.
<code>echo</code>	TRUE	If FALSE, knitr will not display the code in the code chunk above its results in the final document.
<code>results</code>	'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.)
<code>error</code>	TRUE	If FALSE, knitr will not display any error messages generated by the code.
<code>message</code>	TRUE	If FALSE, knitr will not display any messages generated by the code.
<code>warning</code>	TRUE	If FALSE, knitr will not display any warning messages generated by the code.
Code Decoration		
<code>comment</code>	'##'	A character string. Knitr will append the string to the start of each line of results in the final document.
<code>highlight</code>	TRUE	If TRUE, knitr will highlight the source code in the final output.
<code>prompt</code>	FALSE	If TRUE, knitr will add > to the start of each line of code displayed in the final document.
<code>strip.white</code>	TRUE	If TRUE, knitr will remove white spaces that appear at the beginning or end of a code chunk.
<code>tidy</code>	FALSE	If TRUE, knitr will tidy code chunks for display with the <code>tidy_source()</code> function in the <code>formatR</code> package.

R Studio

Updated 10/30/2014

© 2014 RStudio, Inc. CC BY RStudio.

Figure 4.2.2: A schematic table of options available using *knitr* and *rmarkdown* packages.

4.3. USAGE

89

Class Methods

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

Once an *easyReporting* instance is available, with `mkdTitle` it 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 choices 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.

Otherwise it is possible to create an entire CC just with `mkdCodeChunkComplete` and assigning the entire function call as a `message`. This way of working is really useful with `function` creation, where inside a developed function a simple recursive call with parameters assignment can be done as a single `message`.

4.3 Usage

Here we report an example script where a general illustration of the *easyReporting* is described.

```

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

```

4. A REPRODUCIBLE COMPUTATIONAL RESEARCH R6 CLASS

EASYREPORTING

```

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$mkdCodeChunkEnd()
14
15 ## or i can create my own options for the chunk
16 rd$mkdTitle("Second Level Title", level=2)
17 optList <- makeOptionsList(includeFlag=TRUE)
18 rd$mkdCodeChunkSt(optionsList=optList)
19 rd$mkdCodeChunkEnd()
20
21 ## moreover I can add a list of files to source in che code
22 ## chunk
22 rd$mkdCodeChunkSt(optionsList=optList, source.files.list=c(
23   "R/cachingFunctions.R", "R/cachingFunctions.R"))
23 rd$mkdCodeChunkEnd()
24
25
26 rd$mkdCodeChunkComplete(message="a <- 1\nb <- 2\nnc <- a+b\n"
27   print(c)")
28
29 ## otherwise I can make a direct call with all the code
30 ## chunk and the comment in one call
30 optList <- makeOptionsList(includeFlag=TRUE, cacheFlag=TRUE)
31
32 rd$mkdCodeChunkCommented(commentMsg="This is the comment of
33   the following code chunk",
34   message="a <- 1\nb <- 2\n(c <- a+b)",
35   optionsList=optList,
36   source.files.list=NULL)
37 ## finally I can directly compile my report

```

4.3. USAGE**91**

38 `rd$compile()`

The previous R script leads to automatically produce the following *rmarkdown* file.

```

1
2 ---
3     title: "example_report"
4     author: "Dario Righelli"
5     date: "‘r Sys.Date()’"
6     output: rmarkdown::html_document
7 ---
8
9 ‘‘'{r global_options, include=FALSE}
10 knitr::opts_chunk$set(eval=TRUE, echo=TRUE, warning=FALSE,
11   message=FALSE, include=TRUE, cache=TRUE)
12 ‘‘‘
13 # First Level Title
14
15 Here I'm writing a simple paragraph useful to describe my
16 code chunk
17 ‘‘'{r eval=TRUE, echo=TRUE, warning=FALSE, message=FALSE,
18   include=TRUE, cache=TRUE}
19 variable <- 'variable'
20 print(variable)
21
22 ‘‘‘
23 ## Second Level Title
24 ‘‘'{r eval=TRUE, echo=TRUE, warning=FALSE, message=FALSE,
25   include=TRUE, cache=TRUE}
26 ‘‘‘
27

```

4. A REPRODUCIBLE COMPUTATIONAL RESEARCH R6 CLASS **EASYREPORTING**

```

28  ‘‘‘{r eval=TRUE, echo=TRUE, warning=FALSE, message=FALSE,
29   include=TRUE, cache=TRUE}
30   source("/Users/inzirio/Desktop/gDrive/works/coding/
31   easyreporting/R/cachingFunctions.R")
32   source("/Users/inzirio/Desktop/gDrive/works/coding/
33   easyreporting/R/cachingFunctions.R")
34   ‘‘‘
35   a <- 1
36   b <- 2
37   c <- a+b
38   print(c)
39   ‘‘‘
40 This is the comment of the following code chunk
41
42  ‘‘‘{r eval=TRUE, echo=TRUE, warning=FALSE, message=FALSE,
43   include=TRUE, cache=TRUE}
44   a <- 1
45   b <- 2
46   (c <- a+b)
47   ‘‘‘

```

Thanks to the `rd$compile()` command inside the first script, it automatically produces the final *HTML report* illustrated in figure 4.3.1.

4.4. FUTURE WORKS

example_report

Dario Righelli

2018-12-06

First Level Title

Here I'm writing a simple paragraph useful to describe my code chunk

```
variable <- `variable`  
print(variable)
```

```
## [1] 1
```

Second Level Title

```
source("/Users/inzirio/Desktop/gDrive/works/coding/easyreporting/R/cachingFunctions.R")  
source("/Users/inzirio/Desktop/gDrive/works/coding/easyreporting/R/cachingFunctions.R")
```

```
a <- 1  
b <- 2  
c <- a+b  
print(c)
```

```
## [1] 3
```

This is the comment of the following code chunk

```
a <- 1  
b <- 2  
(c <- a+b)
```

```
## [1] 3
```

Figure 4.3.1: An example HTML report with *easyReporting* R package.

4.4 Future Works

The package *easyReporting*, even if useful and easy to handle, can be improved with several additional functionalities. While it is really important to generate an *rmarkdown* file, it could be useful to introduce methods for its file editing. Indeed, if an analyst or a final user, want to correct an analysis, by changing an already inserted CC, he/she could have the possibility to do this, by editing that specific CC or by overwriting it. A possible approach to do this could be to trace the CCs, with a dedicated data structure, while inserting them, and give the possibility to the user to access them with specific class methods.

4. A REPRODUCIBLE COMPUTATIONAL RESEARCH R6 CLASS EASYREPORTING

At the same time, even if the *rmarkdown* options enable the user to store the input/output data with `cache` option, it could be better to provide caching methods, in order to give higher manageability of the data, storing them inside caching database files, as reported also in figure 4.2.1. Additionally, a caching store system (such as BiocFileCache⁷) could produce caching database files, easily shareable through the Internet or as supplementary data of a publication.

In order to reconstruct an entire analysis, giving it a graphical visualization, could be useful to equip *easyReporting* with methods for graph construction. In such a way, the final report could be, not only read by third party users, but also graphically impress the reader.

⁷<https://bioconductor.org/packages/release/bioc/html/BiocFileCache.html>

Chapter 5

Integration of High-Throughput Omics data IntegrHO

5.1 Introduction

Previous chapters showed tools for multiple omics data analysis, integration and visualization using command line tools. Even if the command line approach is pretty common inside the bioinformatics community, it is not for every scientist who is not so confident with programming languages or terminal. Moreover, when working with multiple omics data, there is an overabundance of available tools for each sequencing that can bewilder a beginner, up to the point to renounce approaching the analysis problem.

These considerations brought us to develop Integration of High-Throughput Omics data (*IntegrHO*) a web-based platform for helping also non-expert users in multiple omics data analysis, integration and visualization, without losing track of the analysis steps performed.

5.2 Methods

5.2.1 The interface

IntegrHO is a web platform fully developed in R with aid of Shiny libraries, combining the power of the R statistical instrument with HTML5/javascript flexibility.

Shiny apps are typically designed for small applications, allowing a very easy and versatile way for developing and releasing them. The basic structure of a shiny app is based on two main entities, the Shiny User Interface (SUI) and the Shiny Server (SS). The first one includes all the aesthetic components which the user interact with, while the SS processes all the computations.

Natively, shiny apps support only one server, but when the needs grow up and multiple interfaces are needed the things become more complicated.

Our case is composed of a high number of methodologies for multiple-omics problem solving and required a more complex implementation. To account our problematic, we choose to build *IntegrHO* as self-containing modules, by using recently born shiny modules technology¹. In such a way, the main shiny app can be shredded in multiple "mini apps", each one with its own SUI and SS. This approach is totally invisible to the final user but helps the developer for the maintainability and the extensibility of the entire tool. Indeed, when future needs arise for the implementation of novel functionalities, it is necessary just to implement a novel module.

Our tool presents itself with an upper menu of main topics organized by main scopes. For each of these topics, a sub-menu with specific functionalities is available. When additional functionalities are available, they appear in a left side menu. In order to well set up the parameters for each functionality, an additional side menu is presented with the parameters and their possible values for the right setup (figure 5.2.1 shows a general representation of the main interface).

¹<https://shiny.rstudio.com/articles/modules.html>

5.2. METHODS

97

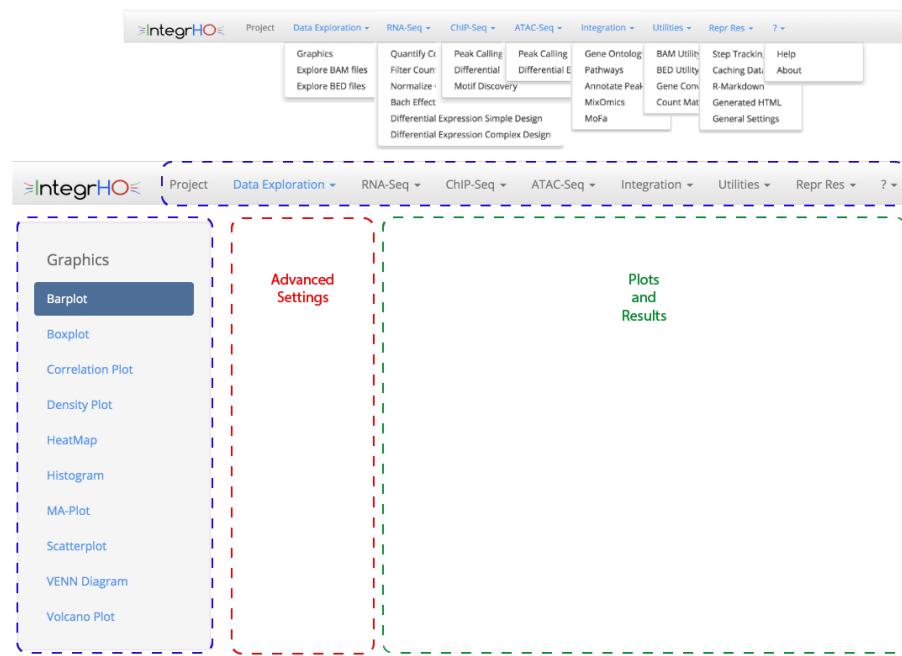


Figure 5.2.1: *IntegrHO* main interface description. A main menu in the upper part is presented with all the available main functionalities, and a left side menu is presented with additional functionalities (in blue). Red part indicates the settings for each functionality within the parameter setup. While in green results in table or graphical form are presented.

Once the user set up the required parameters and used the section button the results are shown in graphical or table format in the main part of the interface.

Before to proceed to data analysis, it is mandatory for the user to set up the project with a dedicated interface. The user has to upload a design file 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 manually edit the design matrix directly from the interface (figure 5.2.2).

5. INTEGRATION OF HIGH-THROUGHPUT OMICS DATA

98 **INTEGRHO**

	path	file.type	experiment	paired	label_ID	treatment	platform.opt.	run.opt.
1	Path/to/file_1.bam	bam	rna-seq	no	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Path/to/file_2.bam	bam	rna-seq	no	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Path/to/file_3.bam	bam	rna-seq	no	2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	Path/to/file_4.bam	bam	rna-seq	yes	2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	Path/to/file_5.bed	bed	chip-seq	no	3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	Path/to/file_6.bed	bed	chip-seq	yes	3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 5.2.2: A screenshot of the project design *IntegrHO* interface.

Using the project interface, *IntegrHO* creates inside the working directory (returned by the `getwd()` function) a dedicated folder with all the required sub-folders and stores all the basic information of the project into an ad-hoc designed `R6ProjectClass`, which is re-used during the whole session to speed up the configuration of each step of the analysis.

5.2.2 Available Methodologies

Unlike tools as Galaxy [59] and Taverna [60], focused on analysis workflows, *IntegrHO* gives to the user high freedom of interaction, reporting all performed steps in a human-readable *HTML* report.

It implements not only the methodologies reported inside *ticorser* for *RNA-Seq* and *DEScan2* for *ATAC-Seq*, but implements also methodologies for *ChIP-Seq* data analysis, complementing these aspects by providing functionalities for their integration at different levels, such as functional enrichment with Gene Ontology and Pathways and peaks-genes annotation.

5.2. METHODS

99

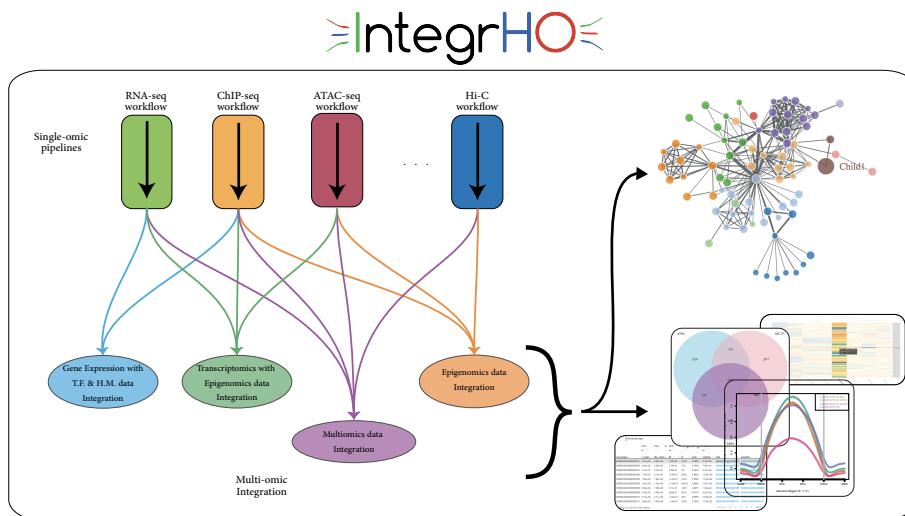


Figure 5.2.3: A schematical representation of *IntegrHO* underlying idea.

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

Moreover, not only our software offers lots of utilities for data management and analysis, but also a great variability of graphics, useful to explore data and results, both in pre-processing and post-processing phase, such as *barplots*, *correlation plots*, *heatmap*, *scatterplots*, etc.

In order to facilitate the multi-omic data integration, we assembled several R functions to be freely combined in order to analyse each single-omic data type and to use their results for multi-omic data integration (Figure 1).

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 the 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 meth-

ods starting from BAM files, we implemented a dedicated interface for *csaw*[61] peak caller, which allows quantification for both broad and narrow peaks, typically resulting from HM and TF *ChIP-seq* data.

For the annotation we selected the *ChIPpeakAnno* and the *ChIPseeker* R/Bioconductor packages, 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 enrichment for Gene Ontology or for Pathways. These last two aspects implemented with aid of *gProfiler* [62] and *graphite* [63] R/Bioconductor packages.

5.3 Reproducible Computational Research

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

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

The auto-produced RMD file as it is, it’s not useful to show to third-parties, it needs to be enriched with natural language comments, that’s why we built a specific interface enabling the user to edit the automatically produced RMD file and to compile it on the fly (see sub-figure 5.3.1a).

The enriched output report can be produced in Hypertext Markdown Lan-

5.4. FUTURE WORKS

101

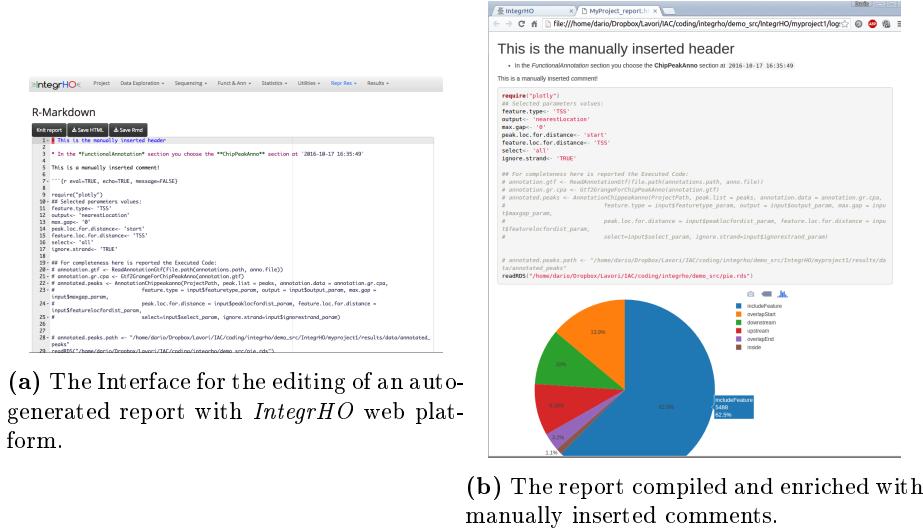


Figure 5.3.1: The reproducible research with *IntegrHO*.

guage (HTML) (sub-figure 5.3.1b) 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.

Furthermore, thanks to the CDF system, each code chunk is not dependent from on previous ones, enabling the final user to add/remove code chunks.

5.4 Future Works

We presented *IntegrHO*, a web platform for multi-omics data analysis, integration and their visualization. Moreover, it traces each step performed by the user by storing input/output data produced and saving executed data inside an RMD file, which can be edited and compiled through the interface itself.

Thanks to its modularity *IntegrHO* allows easy expandability with other methodologies. Indeed, we plan to implement other functionalities for project manipulation and tracing. Indeed, to extend the RR layer we planned to im-

5. INTEGRATION OF HIGH-THROUGHPUT OMICS DATA

102

INTEGRHO

plement methodologies for graph construction of executed modules, in order to produce also a graphical representation of the final enriched document.

Moreover, to deeper address the multiple omics integration phase, we are working on the implementation of methods like mixOmics and MoFa, which gives a graphical response of the samples or the features, when working with high-dimensional datasets.

Chapter 6

Conclusions

Multiple omics data information extraction became always more complex with the evolution of sequencing techniques and with the consequent development of methodologies for their analysis and integration.

With this thesis work, we focused on the development of novel strategies for several problems related to multiple omics sequencing data information extraction. Firstly, we approached the most studied genomic problem, transcriptomic, by focusing on TC *RNA-seq* data. We presented *ticorser*, an R command line package for TC *RNA-seq* data. With this instrument we propose a complete pipeline for the *tc* data analysis, presenting also ad-hoc designed approaches for this type of data visualization at different levels. Additionally, allowing a first integration level with functional annotation and also, in this case, their visualization.

Afterwards, because of the complexity of biological data, it has been almost mandatory to move our attention on epigenomics data, where we focused on *ATAC-seq* technology, an emerging technique for open chromatin region detection, which still needs appropriate solutions for its analysis. To address the lack of methodologies with this data we presented *DEScan2*, an R/Bioconductor package allowing detection of open chromatin regions, and subsequently pro-

posing a possible workflow for the detection of DERs and their integration with *RNA-seq* data, allowing, moreover, a functional annotation integration.

Due to the difficulty of keeping track even of a single omic analysis, we presented *easyReporting*, a possible approach to the Reproducible Computational Research inside R. A tool which helps not only software developers to include RR layers inside their tools, but also non-expert R analysts to easily produce reports for their analysis.

Finally, to speed up and help also non-expert users to analyse multiple omics data, we presented *IntegrHO*, a web graphical user interface for the analysis of omics data. Combining the flexibility of a point-and-click approach with the power of R/Bioconductor statistical approaches for the omics data analysis. Moreover, thanks to the Reproducible Computational Research layer it allows to keep track of all the steps performed during the analysis, and with aid of dedicated interface to live edit the enriched report.

Chapter **7**

Bibliography

1. Annunziato, A. T. DNA Packaging: Nucleosomes and Chromatin. *Nature Education* (2008).
2. Thermes, C. *Ten years of next-generation sequencing technology* 2014. doi:10.1016/j.tig.2014.07.001. arXiv: arXiv:1312.0570v2.
3. Wang, Z., Gerstein, M. & Snyder, M. *RNA-Seq: A revolutionary tool for transcriptomics* 2009. doi:10.1038/nrg2484. arXiv: NIHMS150003.
4. Costa, V., Angelini, C., De Feis, I. & Ciccodicola, A. *Uncovering the complexity of transcriptomes with RNA-Seq* 2010. doi:10.1155/2010/853916.
5. Ozsolak, F. & Milos, P. M. *RNA sequencing: Advances, challenges and opportunities* 2011. doi:10.1038/nrg2934. arXiv: NIHMS150003.
6. 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.
7. Pepke, S., Wold, B. & Mortazavi, A. Computation for chip-seq and rna-seq studies. *Nature Methods* **6**, S22. ISSN: 15487105 (2009).
8. 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.

9. Park, P. J. *ChIP-seq: Advantages and challenges of a maturing technology* 2009. doi:10.1038/nrg2641. arXiv: NIHMS150003.
10. 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).
11. 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).
12. 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).
13. Karolchik, D., Hinrichs, A. S. & Kent, W. J. The UCSC genome browser. *Current Protocols in Human Genetics*. ISSN: 19348258. doi:10.1002/0471142905. hg1806s71. arXiv: NIHMS150003 (2011).
14. Robinson, J. T. *et al.* Integrative genomics viewer. *Nature Biotechnology* **29**, 24–26. ISSN: 10870156 (2011).
15. Thorvaldsdóttir, H., Robinson, J. T. & Mesirov, J. P. Integrative Genomics Viewer (IGV): High-performance genomics data visualization and exploration. *Briefings in Bioinformatics* **14**, 178–192. ISSN: 14675463 (2013).
16. Wang, B. *et al.* Similarity network fusion for aggregating data types on a genomic scale. *Nature Methods* **11**, 333–337. ISSN: 15487105 (2014).
17. Rohart, F., Gautier, B., Singh, A. & Lê Cao, K. A. mixOmics: An R package for àĘomics feature selection and multiple data integration. *PLoS Computational Biology* **13**. ISSN: 15537358. doi:10.1371/journal.pcbi.1005752. arXiv: NIHMS150003 (2017).
18. Argelaguet, R. *et al.* MultiàĘomics Factor AnalysisâĂتا framework for unsupervised integration of multiàĘomics data sets. *Molecular Systems Biology* **14**, e8124. ISSN: 1744-4292 (2018).

BIBLIOGRAPHY

107

19. Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research*. ISSN: 1362-4962. doi:10.1093/nar/gkh036 (2004).
20. Gene Ontology Consortium. Gene Ontology Consortium: going forward. *Nucleic acids research*. ISSN: 1362-4962. doi:10.1093/nar/gku1179 (2015).
21. 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).
22. 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).
23. 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.
24. 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).
25. 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).
26. 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.
27. 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).

28. 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).
29. 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).
30. Robinson, M. D. & Smyth, G. K. Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics* **23**, 2881–2887. ISSN: 13674803 (2007).
31. 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).
32. 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).
33. 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>.
34. 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).
35. Soneson, C. & Delorenzi, M. A comparison of methods for differential expression analysis of RNA-seq data. *BMC bioinformatics* **14**, 91. ISSN: 1471-2105 (2013).
36. 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).

BIBLIOGRAPHY

109

37. Yeh, T. S., Huang, Y. P., Wang, H. I. & Pan, S. L. Spinal cord injury and Parkinson’s disease: A population-based, propensity score-matched, longitudinal follow-up study. *Spinal Cord* **54**, 1215–1219. ISSN: 14765624 (2016).
38. Yeh, T. S., Ho, Y. C., Hsu, C. L. & Pan, S. L. Spinal cord injury and Alzheimer’s disease risk: A population-based, retrospective cohort study article. *Spinal Cord* **56**, 151–157. ISSN: 14765624 (2018).
39. 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).
40. Auerbach, R. K. *et al.* Mapping accessible chromatin regions using SonoSeq. *Proceedings of the National Academy of Sciences*. ISSN: 0027-8424. doi:10.1073/pnas.0905443106 (2009).
41. 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).
42. Li, H. *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics*. ISSN: 13674803. doi:10.1093/bioinformatics/btp352. arXiv: 1006.1266v2 (2009).
43. Lawrence, M. *et al.* Software for Computing and Annotating Genomic Ranges. *PLoS Computational Biology* **9** (ed Prlic, A.) e1003118. ISSN: 1553-7358 (2013).
44. Zhang, Y. *et al.* Model-based analysis of ChIP-Seq (MACS). *Genome Biology*. ISSN: 14747596. doi:10.1186/gb-2008-9-9-r137 (2008).
45. 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).
46. Morgan M, Obenchain V, Hester J, P. H. SummarizedExperiment: SummarizedExperiment container. doi:<https://doi.org/doi:10.18129/B9.bioc.SummarizedExperiment> (2018).

47. Su, Y. *et al.* Neuronal activity modifies the chromatin accessibility landscape in the adult brain. *Nature Neuroscience*. ISSN: 15461726. doi:10 . 1038/nrn.4494 (2017).
48. 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).
49. Barrett, T. *et al.* NCBI GEO: Archive for functional genomics data sets - Update. *Nucleic Acids Research*. ISSN: 03051048. doi:10 . 1093 / nar / gks1193 (2013).
50. Dobin, A. *et al.* STAR: Ultrafast universal RNA-seq aligner. *Bioinformatics*. ISSN: 13674803. doi:10 . 1093/bioinformatics/bts635. arXiv: 1201 . 0052 (2013).
51. 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).
52. 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).
53. 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).
54. Russo, F., \textbf{Dario Righelli} & Angelini, C. *Advantages and Limits in the Adoption of Reproducible Research and R-Tools for the Analysis of Omic Data in International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics* (2015), 245–258.
55. Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, J. D. T. in *Positioning and Power in Academic Publishing: Players, Agents and Agendas* 87 –90 (2016). ISBN: 9781614996491.

BIBLIOGRAPHY**111**

- doi:10.3233/978-1-61499-649-1-87. <<http://ebooks-iospress.nl/publication/42900>>.
56. Blankenberg, D. *et al.* Galaxy: A web-based genome analysis tool for experimentalists 2010. doi:10.1002/0471142727.mb1910s89. arXiv: NIHMS150003.
 57. Giardine, B. *et al.* Galaxy: A platform for interactive large-scale genome analysis. *Genome Research* **15**, 1451–1455. ISSN: 10889051 (2005).
 58. Goecks, J. *et al.* Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology* **11**. ISSN: 1474760X. doi:10.1186/gb-2010-11-8-r86. arXiv: arXiv:1011.1669v3 (2010).
 59. Hillman-Jackson, J. *et al.* Using galaxy to perform large-scale interactive data analyses. *Current Protocols in Bioinformatics*. ISSN: 19343396. doi:10.1002/0471250953.bi1005s38. arXiv: NIHMS150003 (2012).
 60. Wolstencroft, K. *et al.* The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic acids research* **41**. ISSN: 13624962. doi:10.1093/nar/gkt328. arXiv: arXiv:1011.1669v3 (2013).
 61. Lun, A. T. & Smyth, G. K. Csa: 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).
 62. 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).
 63. 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

1.1.1 The Cell	14
1.1.2 the DNA	15
1.1.3 Chromosomes and DNA	16
1.1.4 Histon modification	17
1.2.1 RNA-seq experiment	19
1.2.2 RNA-seq analysis	20
1.2.3 ChIP-seq experiment	21
1.2.4 ChIP-seq peak detection	22
1.2.5 ATAC-seq experiment	23
1.2.6 Omics Representation	25
1.2.7 Integration cameras	25
1.2.8 Integration Funnell	26
2.2.1 ticorser mainflow	29
2.2.2 ticorser boxplot	34
2.2.3 ticorser pca	36
2.2.4 ticorser pca	37
2.2.5 ticorser volcano	39
2.2.6 ticorser MAplot	40

2.2.7 ticorser gene profile	41
2.2.8 ticorser keggmap	42
2.4.1 ticorser dataset	45
2.4.2 ticorser filtering methods	48
2.4.3 ticorser normalizing methods	50
2.4.4 ticorser normalizing methods	51
2.4.5 ticorser Volcano-MA plots	53
2.4.6 ticorser venn diagram	54
2.4.7 ticorser genes trends	55
2.4.8 ticorser venn diagram single time point	57
2.4.9 ticorser pathway network	59
2.4.10ticorser hierachial GO-terms	60
2.4.11ticorser alzheimer keggmap	61
2.4.12ticorser parkinson keggmap	62
3.2.1 DEScan2 workflow	67
3.3.1 DEScan2 dataset illustration	71
3.3.2 The <i>DEScan2</i> and <i>MACS2</i> peaks detection	73
3.3.3 <i>DEScan2</i> peaks detection	74
3.3.4 <i>DEScan2</i> and <i>MACS2</i> filtering comparison	75
3.3.5 <i>DEScan2</i> counts illustration	76
3.3.6 Normalizations applied to null dataset	78
3.3.7 Normalizations applied to detected regions	79
3.3.8 Differential Enrichment Regions Volcano	81
3.3.9 Annotated Differential Enrichment Regions Volcano	82
4.2.1 Reproducible Computational Research illustration	87
4.2.2 knitr options	88
4.3.1 html report	93
5.2.1 <i>IntegrHO</i> main interface	97
5.2.2 integrho design interface	98
5.2.3 integrho representation	99

LIST OF FIGURES

117

5.3.1 <i>IntegrHO</i> report editing	101
------------------------------------------------	-----

List of Tables

2.1	<i>ticorser</i> Design Matrix example	47
2.2	<i>ticorser</i> DE methods results	53
2.3	<i>ticorser</i> Single Time Points DE methods results	56