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

One pill makes you larger, and one pill makes you small
And the ones that mother gives you, don't do anything at all
Go ask Alice, when she's ten feet tall

And if you go chasing rabbits, and you know you're going to fall
Tell 'em a hookah-smoking caterpillar has given you the call
And call Alice, when she was just small

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

When logic and proportion have fallen sloppy dead
And the white knight is talking backwards
And the red queen's off with her head
Remember what the dormouse said

Feed your head, feed your head

Add acknowledgements here

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 omics data explains a different aspect of cellular behaviour. In order to extrapolate relevant information from each omics, it is required to develop specific statistical methodologies for single data analysis and, at the same time, computational methodologies for handling huge amount of data. However, to give a comprehensive view of the cell regulatory mechanisms, it is necessary not only to analize a single omics but also to develop novel statistical and computational models for integrating different omics 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, it also contains a fourth tool (*easyReporting*) for reproducible computational research.

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 implementing 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.

IntegrHO (Integration of High-Throughput Omics data) is a Graphical User Interface (GUI), written in R and Shiny, aimed to analyze and integrate multi-omics data types. It provides a friendly interface to the above-mentioned tools and also incorporates a wide selection of methods and other tools available in the literature. This platform, through an easy point-and-click approach, enables the user to analyze and explore single omics data, such as RNA-seq, ChIP-seq and ATAC-seq and, moreover, it offers the possibility to integrate them at different levels, such as gene-peak annotation and functional annotation methods.

Finally, since in last decades the scientific community has experienced a deep crisis known as "irreproducible research", this thesis presents *EasyReporting*, an R package for an automatic report creation, developed to support the reproducibility of scientific research. Thanks to the R6 class paradigm on which is based on, it is easy to use and to extend.

Overall, this work proposes and combines several computational tools for properly analyzing, visualizing, comparing, integrating and tracing different omics

data types. The results are illustrated with downloaded data from the literature or from collaboration projects.

Contents

Chapter 1

Introduction

This chapter provides all the needed aspects to understand the context of this thesis work, motivates its contents and describes the main contributions.

In particular, it starts from describing some basic biological aspects and how it is possible to study some cellular behaviours from multiple points of view, using different sequencing techniques and how to integrate them. Moreover, it discuss the importance of keeping trace of the computational processes involved in the information extraction, known as "reproducible research".

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.

We distinguish between prokaryotic and eukaryotic cells. The first ones are typically unicellular organisms and don't encapsulate the genetic material (DeoxyriboNucleic Acid (DNA)) inside the nucleus. While the second ones protect the genetic material inside a specific organelle called nucleus, protected by membranes.

For any kind of cell, 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 comples*,*lisosomes* and the, for euokaryotic cells, *nucleus* (figure 1.1.1).

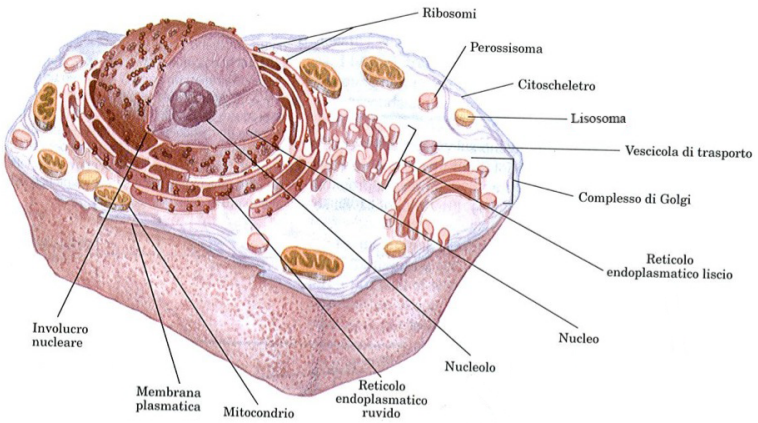


Figure 1.1.1: Schematic representation of an Eukaryotic Cell. It is surrounded by the plasmatic membrane, which encloses the cytoplasm within multiple organelles. One of these is the nucleus which protects the DNA.

1.1.2 The DNA

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

These two molecules are constituted by *nucleotides*, formed 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 *Thymine (T)* . Inside RNA the *Thymine (T)* is substituted by the *Uracil (U)*.

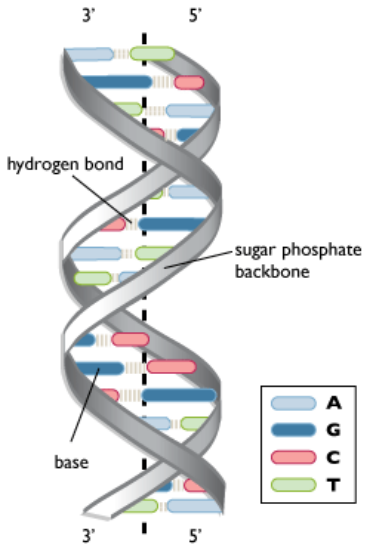


Figure 1.1.2: Schematic representation of double-stranded filament structure of the DNA. The legend reports the four nitrogen bases, Adenine (A), Guanine (G), Cytosine (C) and Thymine (T).

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 chemist-physicist Rosalind Franklyn. 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*, having 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 chromosome is constituted by DNA within thousands of genes, each one composed by introns and exons (figure 1.1.3).

It is important to underlying that since some decades ago the *Central Dogma*

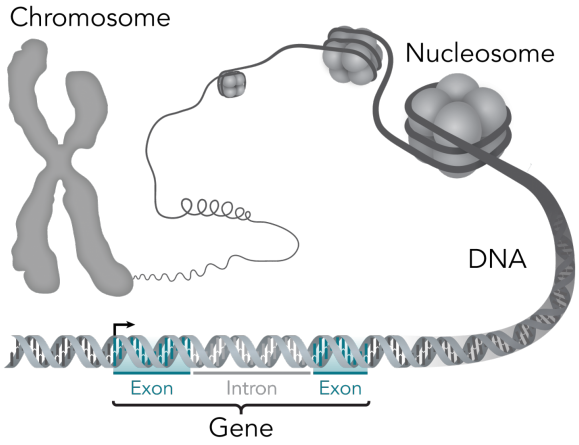


Figure 1.1.3: Representation of the relation between the chromosome and the gene. A gene stands on a portion of a chromosome, which is rolled around a nucleosome. Each gene is composed by introns and exons. When transcripted, the first ones are involved in post-transcriptional cellular mechanisms, while the second ones are destined to be translated into proteins.

of Molecular Biology was founded on the transcription-translation principle, where *exons* were transcribed into RNA, which subsequently it is 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, since few years ago, introns were considered junk DNA, but now they have been demonstrated to be involved in regulation mechanisms, such as post-transcriptional processes, while exons are destined to be translated into proteins (figure 1.1.3).

1.1.3 The Chromatin

In order to fit inside the eukaryotic cells nucleus, the DNA is condensed and wrapped around nuclear proteins (the nucleosomes) forming the chromatin.

Because of chromatin condensed form, for the transcription of a gene, there are some requisites to be respected, such as the accessibility of that specific part of the chromatin, or the binding of specific proteins enabling the accession to the gene region, or the histone modification processes, such as *Acetylation*, *Methylation*, *Phosphorylation* among others, which modify the state of a specific histone, influencing gene expression regulation.

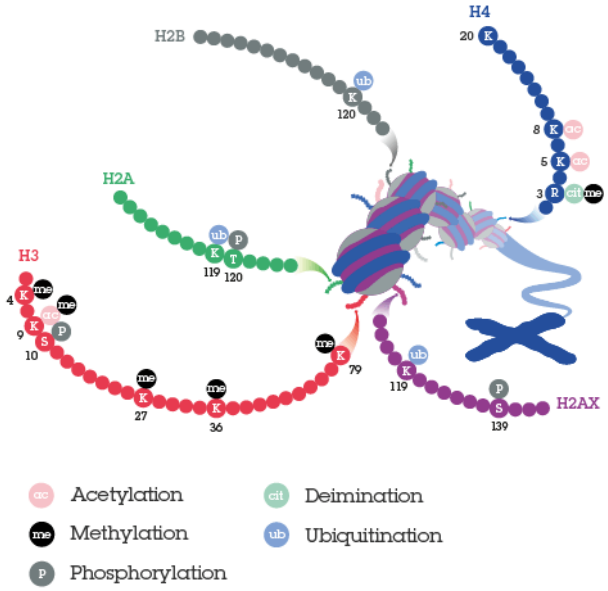


Figure 1.1.4: Representation of some processes involved in histone modification, influencing gene expression regulation, such as Acetylatyion, Methylation, Phosphorylation, Deimination and Ubiquitation.

In particular, the *Acetylation* mostly involves histones *H3* and *H4*, and leads to a lower grade of DNA compaction, subsequently increasing the gene expression. *Phosphorylation* highly involves positions *H3S10* and *T120* of *H2A*, which are associated to the chromatin compaction. It seems to be relevant for the chromosomic condensation during cellular division, for DNA damage repair and for transcription regulator. *Methylation* generally involves histones *H3* and *H4*, but the modification effects depend on the affected position, influencing the gene expression/repression. More in detail, acetylation and phosphorylation directly affect the nucleosome structure, while methylation recruits multiprotein complexes involved into the chromatin remodelling.

Each of these modifications affect the DNA structure, varying not only the chromatin compaction but also by promoting, together with proteins and non-coding RNA, the chromatin folding and influencing the 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* [1]), DNA methylation (*MeDIP-seq* or BS-seq [2]), chromatin accessibility (e.g. *ATAC-seq* [3]), global transcriptional and translational activities (e.g. *RNA-seq* [4]) and 3-D organization of chromatin (e.g. *Hi-C* [5]), 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 omics types within a unified study.

The common steps valid for almost every NGS data type starts from a library preparation (depending on the sequencing technique), an amplification of the genomic material and finally the sequencing of the samples.

The typical analysis requires several steps to achieve good quality results, regardless of the specific final aim of the investigation. A standard analysis starts with a quality control assessment of the raw sequenced fragments (reads), using software as *FastQC* [6]. After quality control assessment, in general, it is fundamental for the analysis to map the raw reads on a reference genome. But for the read mapping phase we have to distinguish the aligner depending on the sequencing technique we are studying. For DNA sequencing *Bowtie* [7, 8] and *BWA* [9, 10] are the most common ones, while for *BS-seq* there is *Bismark* [11].

For *RNA-seq*, during the last years, in order to obtain higher accuracy in read mapping (alignment), several software, such as *STAR* [12] and *HISAT2* [13], superseded broadly used tools as *TopHat2* [14]. Usually, an aligner produces Binary Alignment Map (BAM) files[1] [15] where all the reads (mapped and un-

[1] `http://samtools.github.io/hts-specs/SAMv1.pdf`

mapped) are collected with additional fields relevant for further steps of the analysis. Depending on the final aim of the analysis, BAM files can be further processed to delete or to split the reads. Usually the software used for this scope is *samtools* [15, 16], available also in R programming language with *rsamtools* [17]. From this point on, the analysis steps are dependent on the sequencing technique and from the experimental investigation.

Here we present the basics about the main sequencing techniques addressed in this thesis work, needed to better understand the outline of next chapters. Please refer to the following chapters for details about each data type analysis.

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* [4, 18–20] 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 [21].

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.

However, *RNA-seq* also offers the possiblity to identify and investigate isoforms abundance [22–25]. Although, such aspect will not be further treated in this thesis.

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).

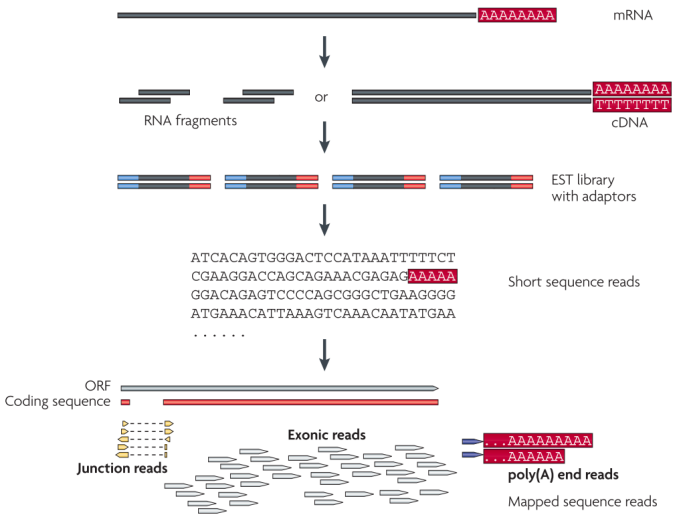


Figure 1.2.1: Representation of an RNA-seq experiment. The messanger RNA (mRNA) is converted into complementary DNA (cDNA) or RNA fragments, to be linked to short sequencing adaptors. Producing sequencing reads via high-throughput technology. The resulting reads are aligned to a reference genome or transcriptome, to be classified as exonic, junction or poly(A) end-reads. [4]

Afterwards, the so-obtained reads need to be analyzed with several tools, depending on the particular question the researcher is interested in [26, 27].

In particular we focused on differential gene expression in case of multiple biological conditions, due to stress, drug treatments, disease-specific, etc. Figure 1.2.2 shows a pipeline we designed inside the *RNASeqGUI* [28] software for *RNA-seq* data analysis. After the alignment of the reads on a species's reference genome, the software allows for 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.

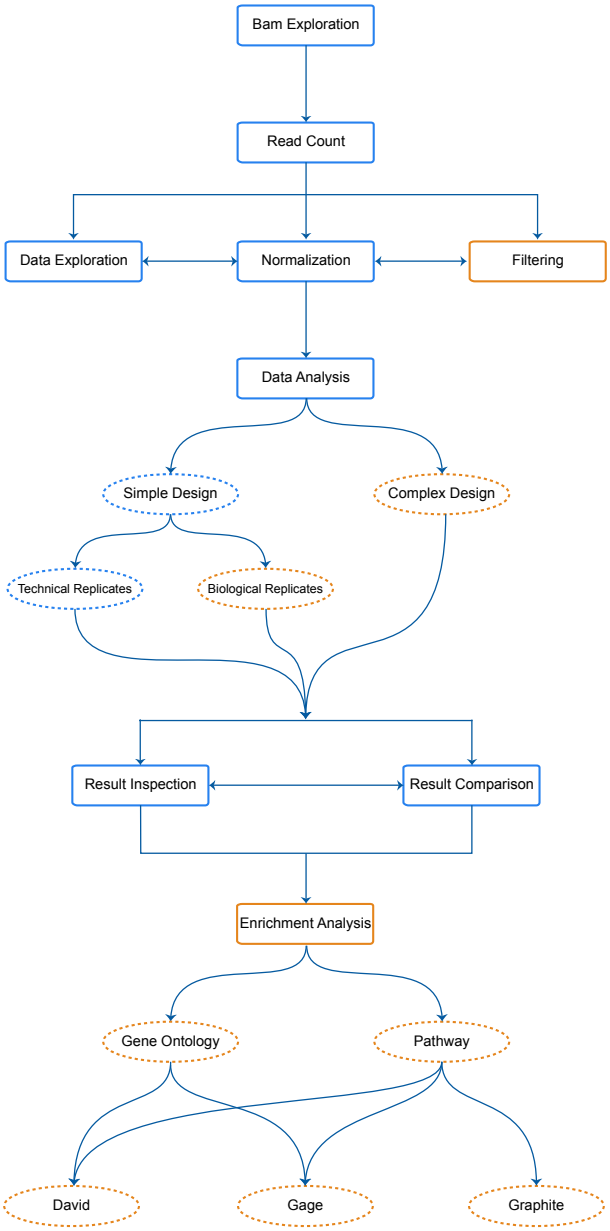


Figure 1.2.2: A representation of *RNA-seq* analysis pipeline, as implemented in *RNASeqGUI*, a graphical user interface for *RNA-seq* data analysis (Image from [29]). BAM files can be explored and their contained reads can be used to quantify the expression of genomic features. Afterwards, it is possible to filter low expressed features and normalize the resulting ones, in order to lower the noise that can affect further results. Subsequently, it is possible to analyze the data for inspect for differential expression of the genes between multiple conditions, this is relevant to understand how a "treatment" is affecting a specific condition respect to another one. Finally the results can be graphically inspected and/or integrated with Gene Ontology or Pathway databases to obtain a more detailed information about the cellular mechanisms affected by the "treatment".

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, depending on the experimental design (simple or complex), it is possible to choose between several methods for the detection of differential expression of the features between the conditions (see section 2.2.3). Finally, the significant features can be integrated with databases of biological functionalities in order to detect the mostly influenced ones.

1.2.2 ChIP-seq

As previously mentioned, epigenetic marks are a fundamental aspect 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 damaging/repairing, the second ones (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, the mostly used sequncing technology is the Chromatin ImmunoPrecipitation sequencing (*ChIP-seq*) [1]. This technique allows to use antibodies for selected proteins or nucleosomes, which enriches for specific DNA sequences bounded to these proteins/nucleosomes.

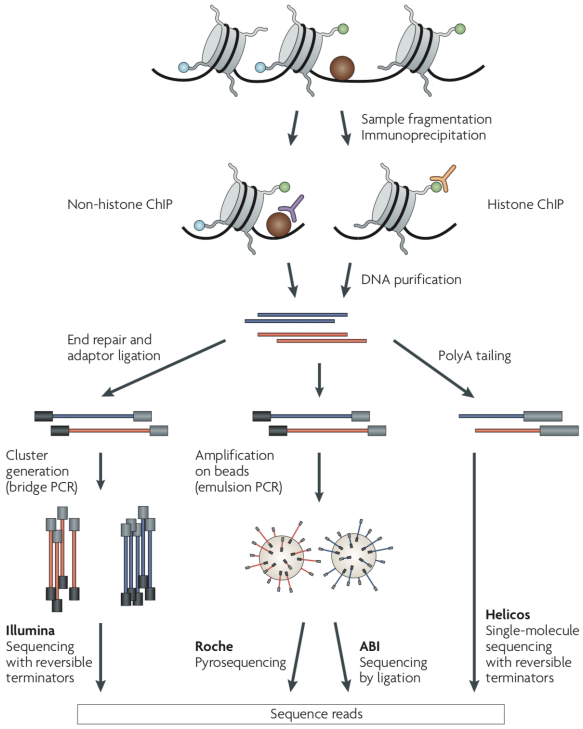


Figure 1.2.3: *ChIP-seq* experiments can be used to investigate DNA binding sites such histone modifications (Histon ChIP) and transcription factors (Non-histone ChIP), using specific antibodies for the specific interested protein (immunoprecipitation). After the DNA purification, the genetic matherial is linked to specific adaptors and then amplified in different manner, depending on the amplification and sequencing type used. [1]

The library preparation consists into locking 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, to be sequenced.

When analyzing the data, it is relevant to distinguish between HM and TF 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.

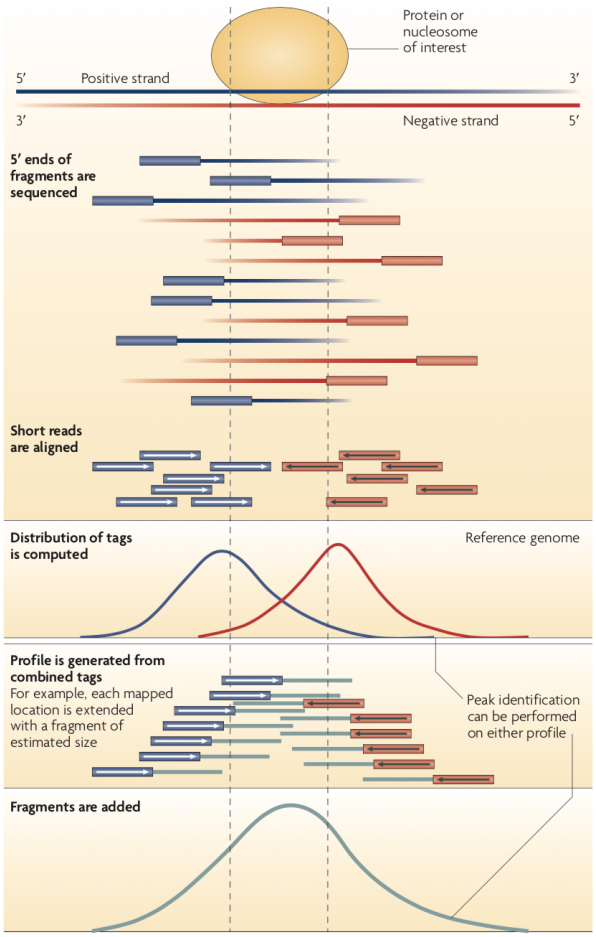


Figure 1.2.4: The DNA fragments at 5' are sequenced, producing a double profile for each DNA strand in correspondence of the surrounding area of the protein binding site. In such a way it is possible to identificate a unified peak combining the profiles with the reads.[1]

Subsequently to the peak detection, several aspects can be investigated about the signal, 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 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* [30], *DNase-seq* [31], *ATAC-seq* [3], etc.

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

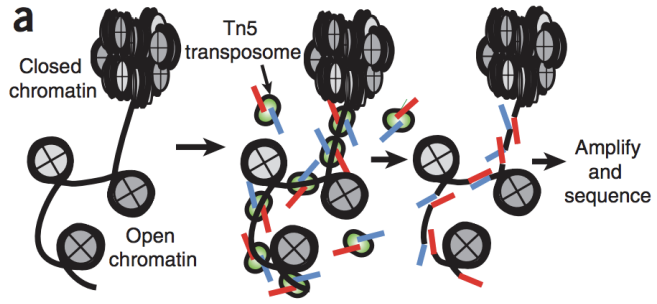


Figure 1.2.5: *ATAC-seq* library preparation uses *Tn5* transposome natural predisposition to link and cut to the DNA. In so doing it provides DNA sequences that can be easily amplified and sequenced to be mapped and processed [3].

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 standard analysis reached for the *ATAC-seq* analysis, but, inspired by the *ChIP-seq* analysis, the resulting reads, typically, are processed with tools (peak callers) for quantifying their amplification, which produces for each detected open chromatin region a feature: the peak (generally with an associated score) [32].

Depending on the used tool, the peaks can be represented in different data structures, but their representation is typically 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 [33–35].

1.3 Multi-omics Integration

All above-mentioned 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 omics at the same time.

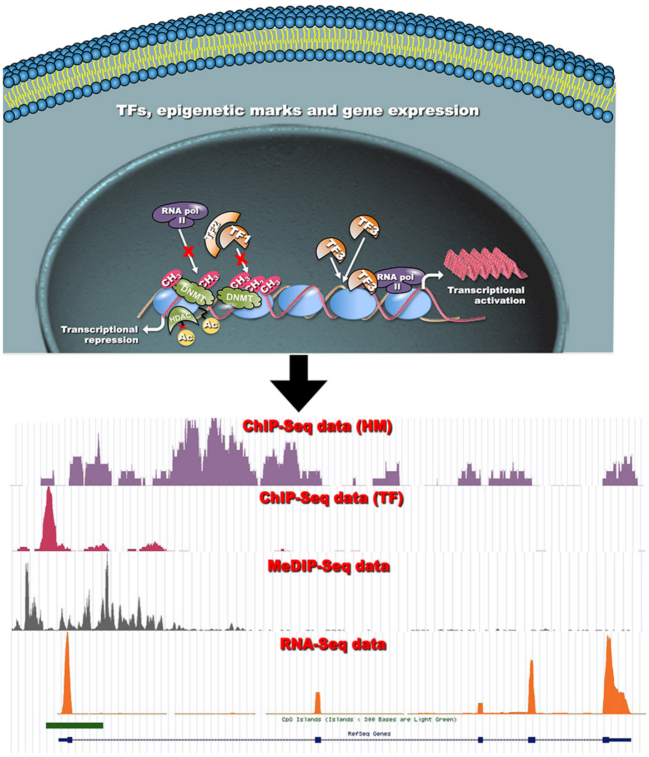


Figure 1.3.1: An example of different omics coming from the same biological experimental conditions. An HMs (such as *Acetylation*) can have inhibited the accession to bind specific nucleosomes, because already occupied by specific TFs, inhibiting, in such a way, the expression of those genes present in that particular portion of the chromatin (upper part). To investigate these processes, we can perform multiple experiments of *RNA-seq* and *ChIP-seq* (one for the HM and one for the *tf*) to study singularly each epigenomics factors and the gene expression (lower part). (image adapted from [36])

Indeed, we can imagine each omics 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 each device snapshots together.

The same idea can be adapted to the sequencing techniques, we need to integrate the information coming from different omics in order reconstruct (and understand) how multiple mechanisms orchestrate the cellular behaviour.

Indeed, we can imagine a typical combination of epigenomics factors affecting the chromatin in order to influence the transcription of one or more genes. An HMs (such as *Acetylation*) can have inhibited the accession to bind specific

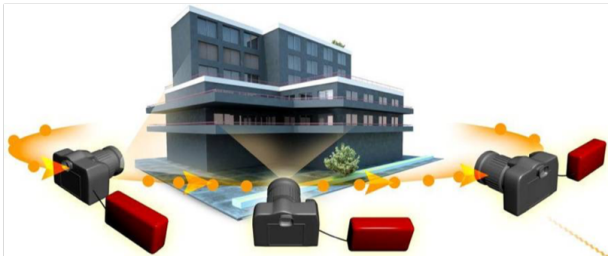


Figure 1.3.2: An illustrative exaple of a multi-view camera system pointed on a building. Each omics can be represented by a different camera pointed on the same building. The final snapshots can be integrated to reconstruct the building, which represents the cellular behaviour.

nucleosomes, because already occupied by specific TFs, inhibiting, in such a way, the expression of those genes present in that particular portion of the chromatin (upper part of figure 1.3.1). To investigate these processes, we can perform multiple experiments of *RNA-seq* and *ChIP-seq* (one for the HM and one for the *tf*) to study singularly each epigenomics factors and the gene expression (lower part of figure 1.3.1).

As figure 1.3.3 outlines, multi-omics data integration can be made at different levels, by graphical exploration, by functional annotation, by network fusion and by dimensionality reduction. We can distinguish two main approaches,; "analyze each omics then integrate the results" and "jointly analyze each omics to improve the power". The first one can be done also with few samples, where each omics is analyzed standalone and then the results can be combined to obtain a common response (levels 1 and 2 of figure 1.3.3). The second main approach needs a high number of samples to analyze all the omics together to obtain more power in the response prediction (levels 3 and 4 of figure 1.3.3) [37–40].

More in general, integration can be performed at different levels (i.e. descriptive, exploratory, inferencial, etc.).

With graphical exploration, we can visualize data coming from different sources (e.g. *RNA-seq* and *ChIP-seq*) using specific tools designed at this scope, such as *Genome Browser* [41] or Integrative Genomics Viewer (IGV) [42, 43] and looking to the overlapping regions, or where expressed epigenomic markers have corrispondence 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,

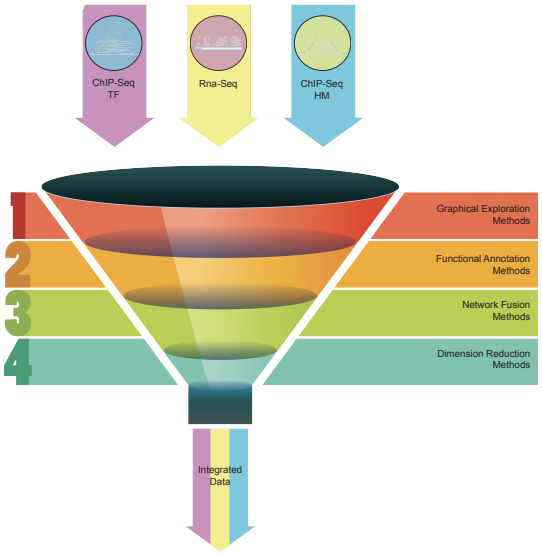


Figure 1.3.3: A schematic representation of multi-omic data integration levels. Graphical exploration can be used to visualize data coming from different sources (e.g. *RNA-seq* and *ChIP-seq*) using specific tools designed at this scope. Functional annotation integration combines analysis results (such as relevant lists of genes) with publicly available databases to detect functional responses. Network fusion techniques are used when a high number of data samples is available or when having multiple-omics data types coming from the same patients. Also dimensionality reduction techniques require several samples to be able to obtain relevant and reliable results.

like the Gene Ontology[2] and pathway (*KEGG*[3] or *Reactome*[4]) based ones, to detect functional responses highly related to the experimental condition under investigation.

It is possible to integrate multiple omics data types by constructing multiple regulation 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 [44].

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 [37,

[2]http://www.geneontology.org/
[3]https://www.genome.jp/kegg/
[4]https://reactome.org/

38].

1.4 Reproducible Research

During last years "irreproducibility" became a general problem, particularly in Medical-Biosciences where high-dimensionality data are collected from so many different omics to investigate the cellular behaviour, in order to propose new drugs and new patient-oriented treatments.

Due to the analysis complexity and to the not-so-well documented processes, several published papers [45, 46] became a landmark for irreproducibility, leading the scientific community to highlight the weaknesses in reproducing published scientific facts and to promote the Reproducible Research (RR).

The underlying idea of RR [47] is to provide public access to analytic data and related analysis code (combined with natural language documentation), in order to allow third-parties to entirely reproduce the findings.

In recent years, many efforts have been spent for sensitizing the scientific community on the relevance of RR, such as by producing tools [29] supporting reproducible research, by producing scientific works in reproducible research spirit [48] or by providing an overview of available methodologies for RR [28].

Anyway, to address RR in scientific works many cares are needed during the production of a software or when analyzing data. As figure 1.4.1 [49] shows, the author cannot only make an analysis, but needs to store all the components needed to a third-party user to reproduce the entire analysis. To do so, the author needs to give public access not only to the data accompanied but also to the code used to process them, in a "database-like" form, required to maintain the reproducibility across the time, which can bring to different code tool versioning. Moreover, it is fundamental to present the results in the same form as they can be produced with the "database", storing the data and the code.

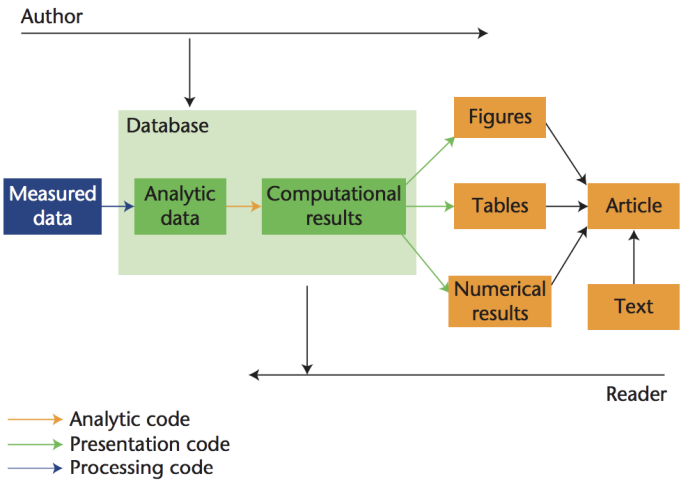


Figure 1.4.1: A scheme of Reproducible Research. The author produces a database within the analytic data and the computational results (green boxes) produced during the analytical phase (yellow arrows), in order to help the reader in reproducing the results presented (green arrows) inside a published article (in yellow). (Image from [49])

Of course there are several disadvantages trying to respect RR constraints, such as the time required not only to analyze the data but also to trace the code and the data in order to give them public access. Moreover, nowadays there are several tools [50–53] trying to help implementing RR inside the scripts and the tools, but it is not so simple to determine which one works better than another, and this can be due to the problem that a standard has not been reached yet.

Fortunately, some international journals are starting to require proof of transparency for the proposed works which include computational analysis. This is advantageous because from one side is leading the scientific community to propose always easier solutions for using tools for RR, and from another side the analysts are "forced" to release their data and code to demonstrate transparency and, consequently, more robust findings.

Unfortunately, despite all efforts of RR community, the Life Science society is still far from a standard for reproducibility [54], showing that the way is still long to be covered.

1.5 Aim of the Work

The complexity of NGS data analysis brought the scientific community to develop multiple methodologies and tools for their analysis, reaching in some cases (such as *RNA-seq*, *ChIP-seq*, etc.) standard pipelines for their analysis. However, despite such effort there are specific context that are much less developed and still require attention.

Nowadays, the challenge moved on the development of statistical and computational methodologies for the information extrapolation from multiple-omics datasets, in order to produce a unified view of the biological characteristics describing the cellular behaviour.

Motivated by these aspects, we have embarked on a path that goes through the analysis of different omics to flow into multiple projects, each one with the common aim of helping the single omics analysis in order to simplify reaching the goal of integrating them in a unified view.

This thesis is focused on the development of three main computational tools (*ticorser*, *DEScan2* and *IntegrHO*), firstly approaching single omics data analysis and then on their integration. Additionally, in order to face up the irreproducility problem in the scientific community, a fourth tool (*easyReporting*) is presented.

Ticorser (time course *RNA-seq* data analyzer) is a novel R package aimed to analyze time-course RNA-seq data, which still require more attention. 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, for which very few methods are available. It consists in the following three-step procedure: 1) It identifies candidate regions inside each sample implementing 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 con-

ditions and also for integrating this data type with other omics data, such as RNA-Seq.

IntegrHO (Integration of High-Throughput Omics data) is a Graphical User Interface (GUI), written in R and Shiny, aimed to analyze and integrate multi-omics data types. It provides a friendly interface to the above-mentioned tools and also incorporates a wide selection of methods and other tools available in the literature. This platform, through an easy point-and-click approach, enables the user to analyse and explore single omic data, such as *RNA-seq*, *ChIP-seq* and *ATAC-seq* and, moreover, it offers the possibility to integrate them at different levels, such as gene-peak annotation and functional annotation methods. Trying to provide a flexible and easy-to-use tool enjoyable also by non-expert analysts, such as biologists, or beginner analysts.

Finally, since in last decades the scientific community has experienced a deep crisis known as "irreproducible research", this thesis presents *EasyReporting*, an R package for an automatic report creation, developed to support the reproducibility of scientific research. Thanks to the R6 class paradigm on which is based on, it is easy to use and to extend.

Overall, this work proposes and combines several computational tools for properly analysing, visualizing, comparing, integrating and tracing different omics data types. The results are illustrated with downloaded data from the literature or from collaboration projects.

Chapter 2

Time Course RNA-seq analyzer: ticorser

When working with *RNA-seq* data, there are multiple aspect that can be studied, such as single nucleotide variations, alternative splicing, post-transcriptional alterations, fusion gene detections, etc.

In this chapter we focus on Time Course (TC) experimental design with *RNA-seq* data using Time Course RNA-Seq data Analyzer (*ticorser*) a tool developed for analyzing *RNA-seq* time-course data.

2.1 Introduction

When working with *RNA-seq* data it should be interesting to investigate the evolution in time of the gene expression between two or more conditions.

When defining a time-course experiment we refer to muliple samples, taken at subsequent time points, in multiple conditions. If we suppose to want to investigate the cellular gene expression effects of a drug treatment in 24 hours, it could be a good idea to perform multiple sampling (i.e. one per hour) of the treated and the untreated cells. Moreover, to obtain more power to be statistically significant with our results we'll need to have multiple replicates at each sampling. In such a way, we are able to collect multiple replicates at multiple times (24) for each condition (treated and untreated). By doing

sequencing of these samples and inspecting the resulting experimental sequences, we are able to explore with ad-hoc designed statistical and computational tools the differences in time between the conditions.

To afford problematic like this one, we designed *ticorser*, 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* and *DESeq2*, it offers the possibility to set up 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 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 terms and Pathways [55, 56]).

Overall, *ticorser* offers the possibility to analyze data using different R/-Bioconductor [52] packages, to compare the results in order to choose the best combination of tools for the user specific problem. Therefore, the software implements a vast amount of exploratory and diagnostic interactive plots to explore data not just at pre-processing step 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 [57] 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 de-

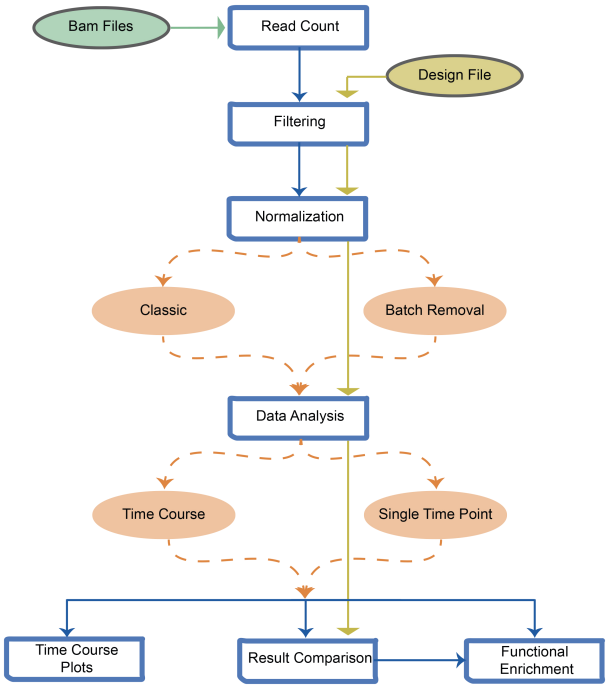


Figure 2.2.1: Main flow of ticorser R package. The package takes as input a list of bam files for producing a count matrix of the features (on the rows) and the samples (on the columns). Afterwards, with aid of a design file, *ticorser* allows the filtering of low expressed genomic features and their normalization with classic methodologies and with spefici ones for batch removal. Once the data are comparable it is possible to perform a differential expression at time-course level and/or at specific time point. The results can be investigated for plotting, comparison or for functional enrichment.

scribes 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 normalization methods but also the methodologies for batch effect removal.

In oder to inspect different approaches to analyze TC data, *ticorser* offers three different methodologies for analyzing TC *RNA-seq* data, and four 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 [58] influencing final results.

In order to address this task, *ticorser* offers multiple statistical methods for low expressed genes filtering. The `filterLowCounts` method is partially based on the `filtered.data` function of *NOISeq* R/Bioconductor package [59]. The method gives the possibility to apply four different filtering methods, the Counts per Million (CPM), the *proportion* and the *Wilcoxon* tests and an our-defined method named *quantile*.

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

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

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

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

2.2.2 Data Normalization

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

For this reason *ticorser* is particularly focused on this aspect. Indeed, through the `normalizeData` function *ticorser* provides five different normalization methods, 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 [60], 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

To investigate differences between multiple conditions, *ticorser* offers three different ways of analyzing TC RNA-Seq data, depending on the biological question under investigation.

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 [61] R/Bioconductor packages, *DESeq2* [62]; *edgeR* [63]; *NOISeq* [59].

All the selected methods model the *RNA-seq* data counts for each gene as independent Negative Binomial distributions, which has been demonstrated [64] 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/Biocnductor package, we compare two different formulas. The first one defines the *full* model where we put together the timepoints, the conditions and an interaction term between these two variables, while the second one is a reduced model where the interaction term is removed:

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

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

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

The underlying idea of the second method is the same of the first one, where the difference 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 variable appears:

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

Time-Course DE Method 3 - *LRT_NOInteraction*

Using always the *DESeq2* LRT we defined a third method for the identification of DEGs that have different expression between the conditions across all the time points, but that maintain the same profile in both conditions.

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

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

Single DE Methods

To account for fixed time point experiment we implemented functionalities for helping also the 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 always a good norm to understand how the applied transformations are influencing the features quantification. 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).

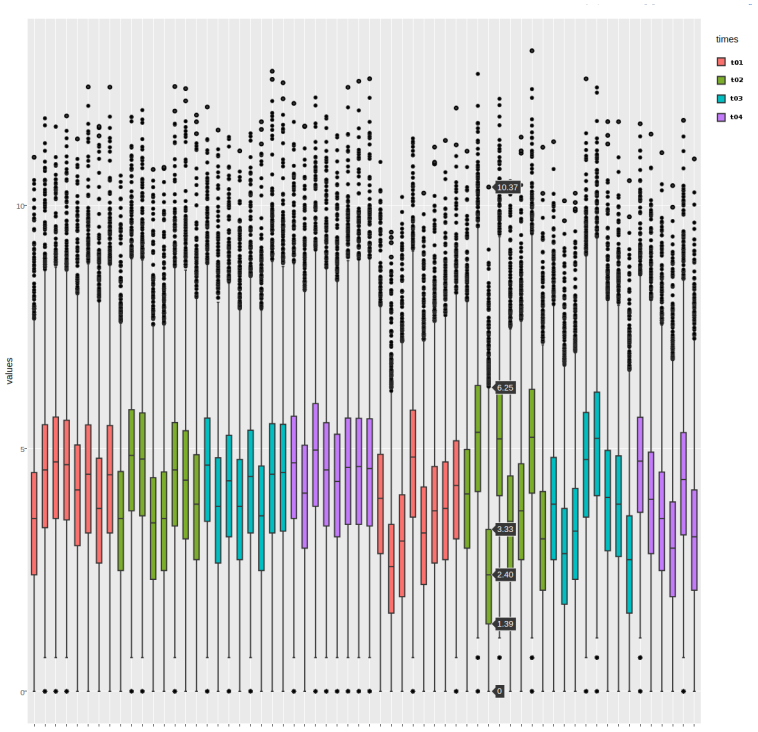


Figure 2.2.2: An example of interactive boxplot made with *ticorser* package. Each boxplot represents a specific sample, while each colour represents a specific time point. When passing the mouse over a boxplot it shows additional information about its quartiles.

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.

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.

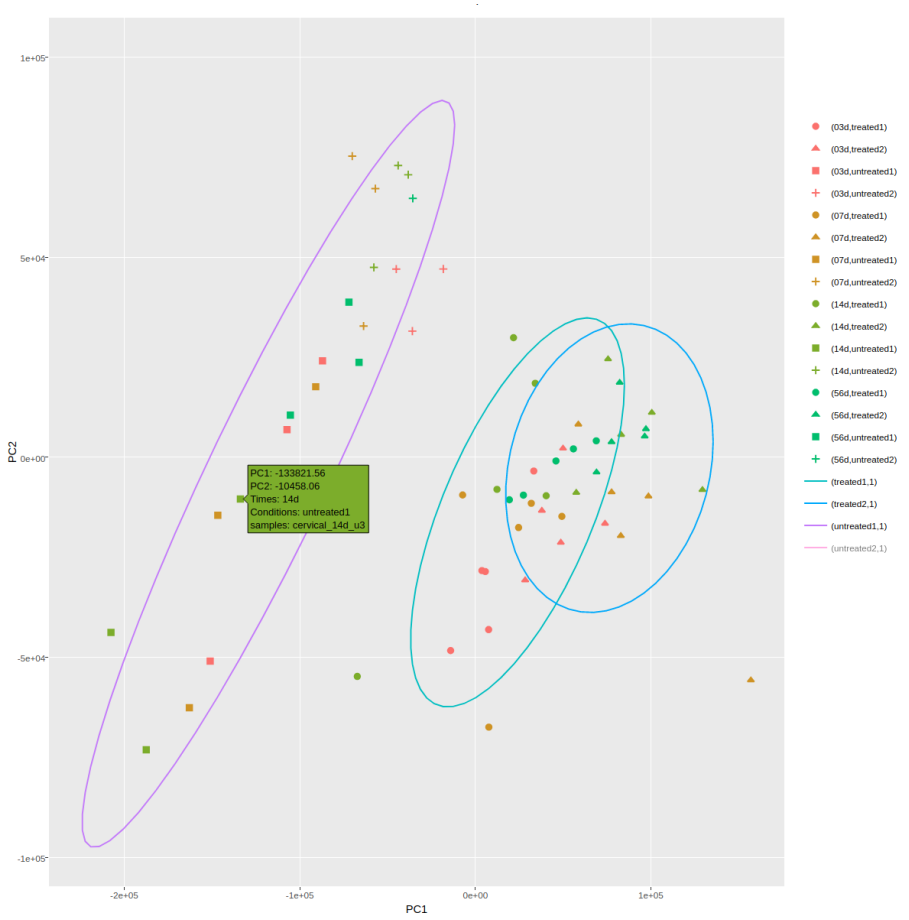


Figure 2.2.3: An example of interactive PCA made with *ticorser* package. Each dot represents a specific sample, while each colour represents a time point, each symbol represents a biological condition group. When passing the mouse over a dot it shows additional information about the selected sample,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

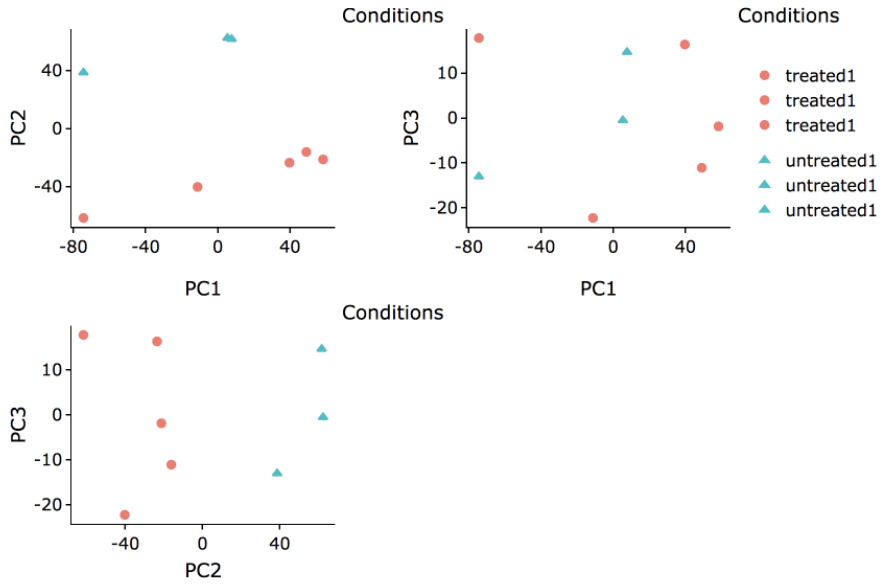


Figure 2.2.4: An example of interactive matrix PCA made with *ticorser* package. Each plot puts in relation two different Principal Components. While each dot represents a specific sample, while each colour represents a time point, each symbol represents a biological condition group. When passing the 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 $log_{10}(p-value)$ of each gene, in order to highlight the significant changes inside the data experiment.

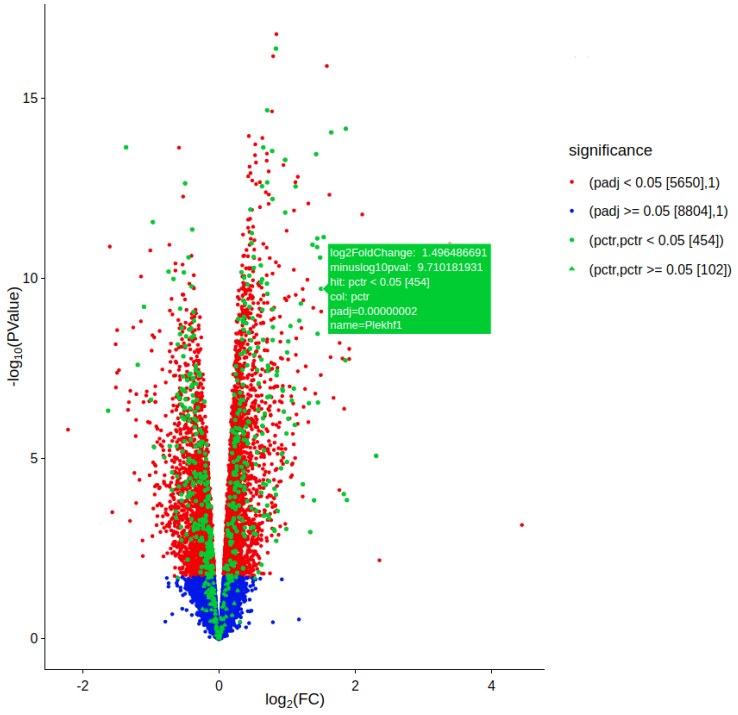


Figure 2.2.5: An example of interactive volcano plot made with *ticorser* package. Each dot represents a gene, while blue and red colours highlight the significance of the genes. In green, there are those genes coming from the positive control list. When passing the mouse over a dot it shows additional 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 $log_2(FC)$, 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).

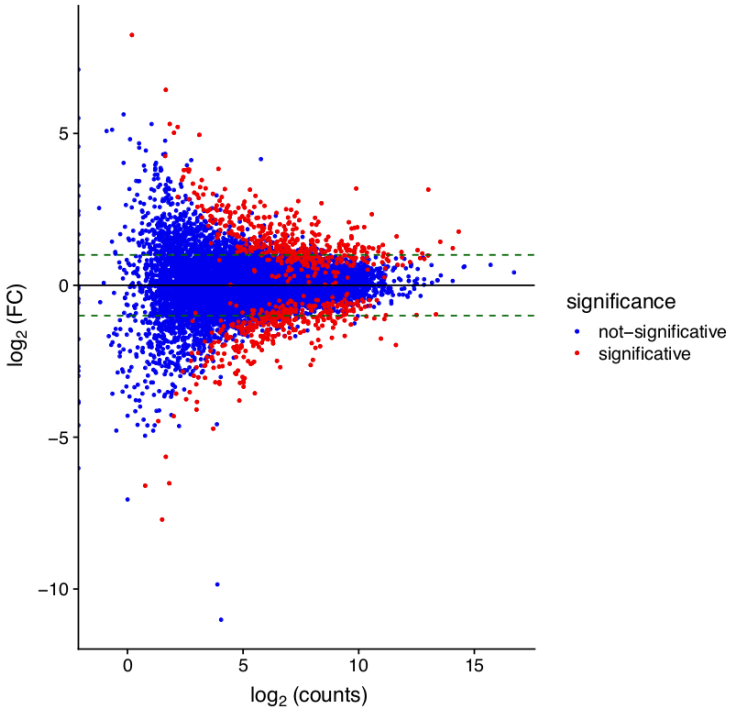


Figure 2.2.6: An example of interactive MA-plot made with *ticorser* package. Each dot represents a gene, while blue and red colours highlights the significance of the genes. When passing the mouse over a dot it shows additional information about the selected gene.

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 2.4.7).

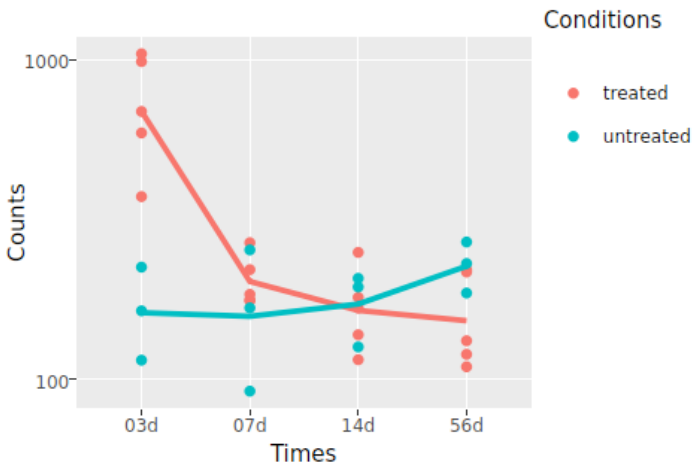


Figure 2.2.7: An example of interactive gene profile made with *ticorser* package. Each dot represents the counts value of the gene in a sample. Colours identifies the conditions of the samples. The lines represent the gene trend over all time points.

KEGG-Maps

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*[65] 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.

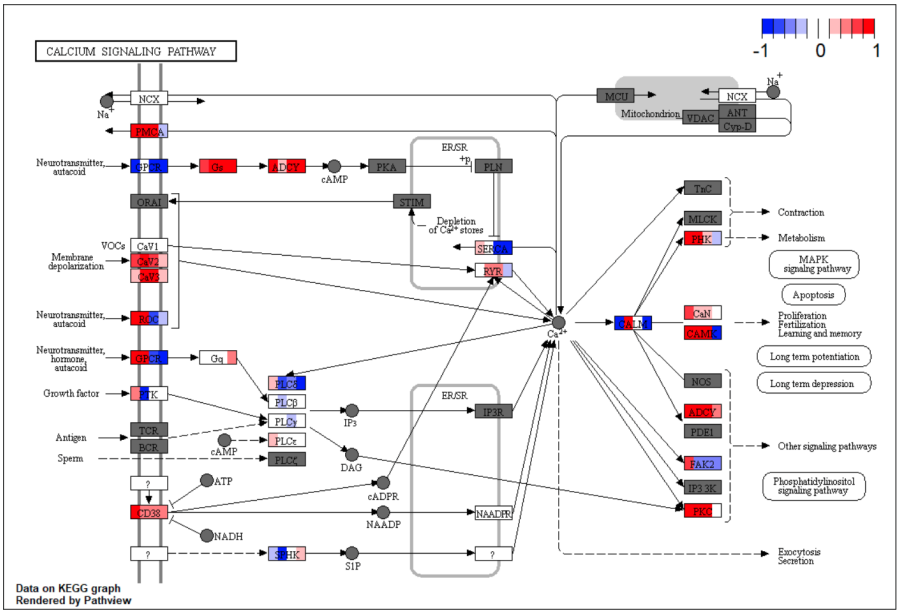


Figure 2.2.8: add description

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 [66], 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* [67] 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.

The *ticorser* pipeline, as described in figure 2.2.1, has been followed for this case study.

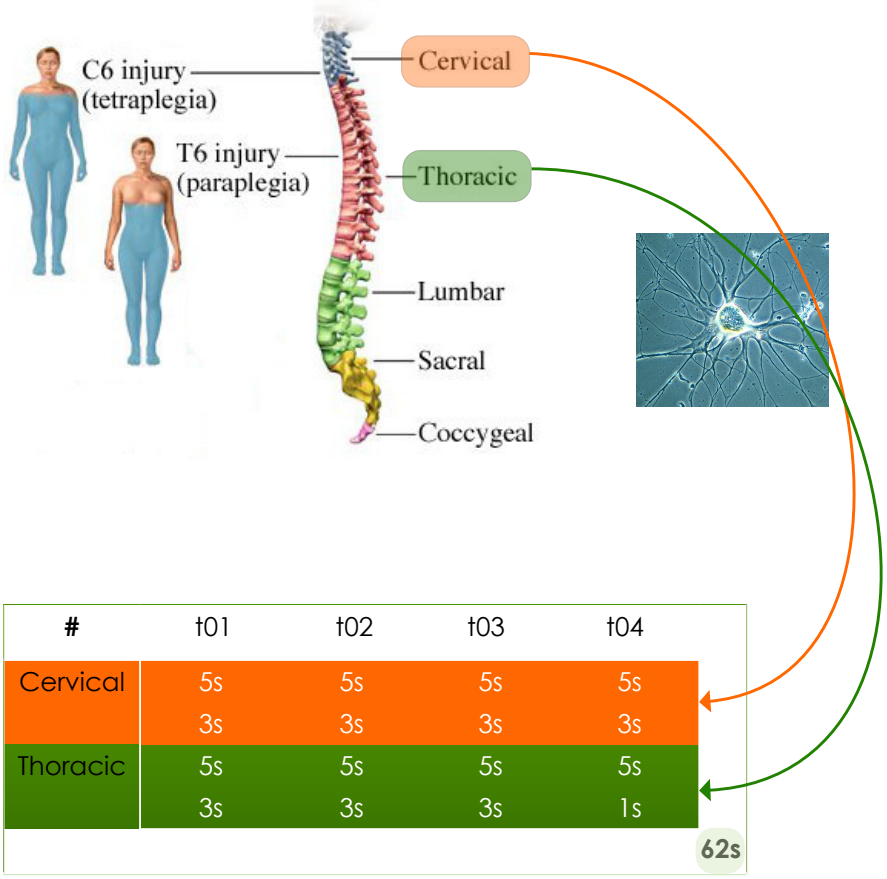


Figure 2.4.1: An illustrative example of the used dataset. Neurons of Cervical and Thoracic spinal cord injury, have been extracted and sequenced at 4 different time points. Inside the table, for each tissue (Cervical/Thoracic) the number of samples for the treatments (upper line) and controls (lower line) are reported at each time point.

Because the dataset is not yet accessible, the genes and the samples have been masked during the analysis and no further details will be provided, but we only use it as 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 `countBamFiltesFeatureCounts` method with the path of the *BAM* files and a GTF[1] file within the desired

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

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.

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

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

Table 2.1: An example of design matrix required for the right working of *ticorser* package. It provides 3 mandatory columns. The first column describes samples names equal to the column names of the count matrix. The *Times* column indicates the time points for each samples, while the *Conditions* column specifies the experimental conditions. Finally, the *Tissue* column (not mandatory) identifies possible multiple tissues.

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. In order to account for this, it is a good practice to normalize data, a step which affects the DEGs detection [68–70]. 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 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.

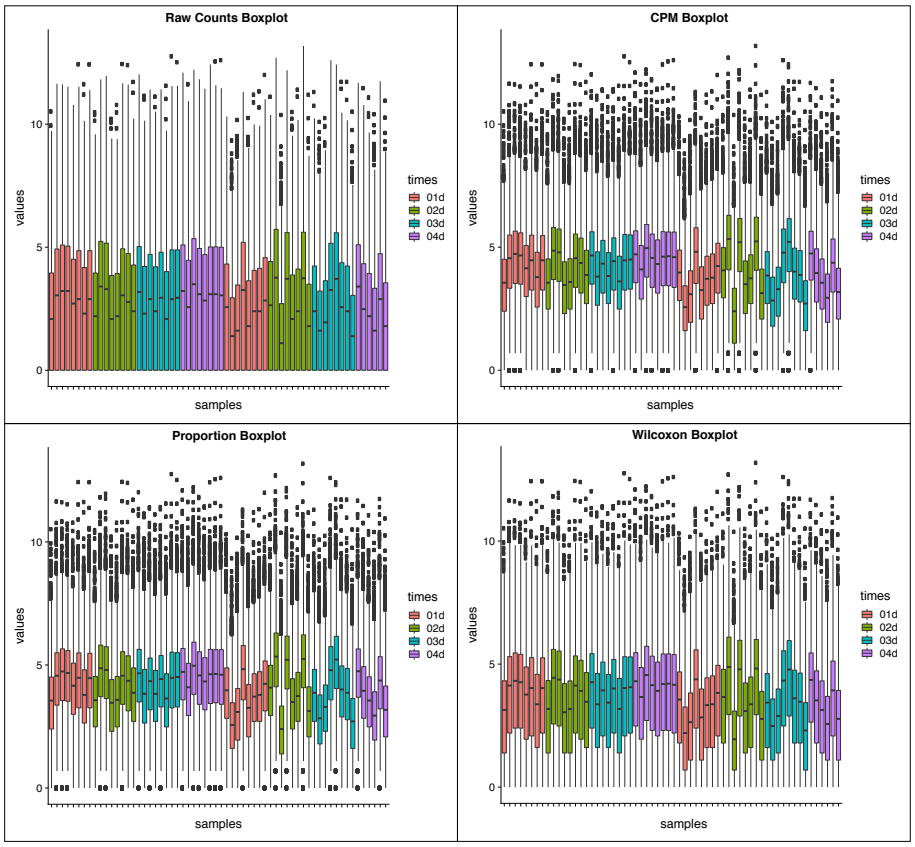


Figure 2.4.2: A comparison of how the filtering methods available in *ticorser* affect the data. The first panel shows the row counts with low expressed genes, while the other boxes show the filtering effect on the samples.

Moreover, looking at the *Wilcoxon* test, it is more conservative, because it 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 removes the biases from the samples by clustering the samples of the same group (figure 2.4.4).

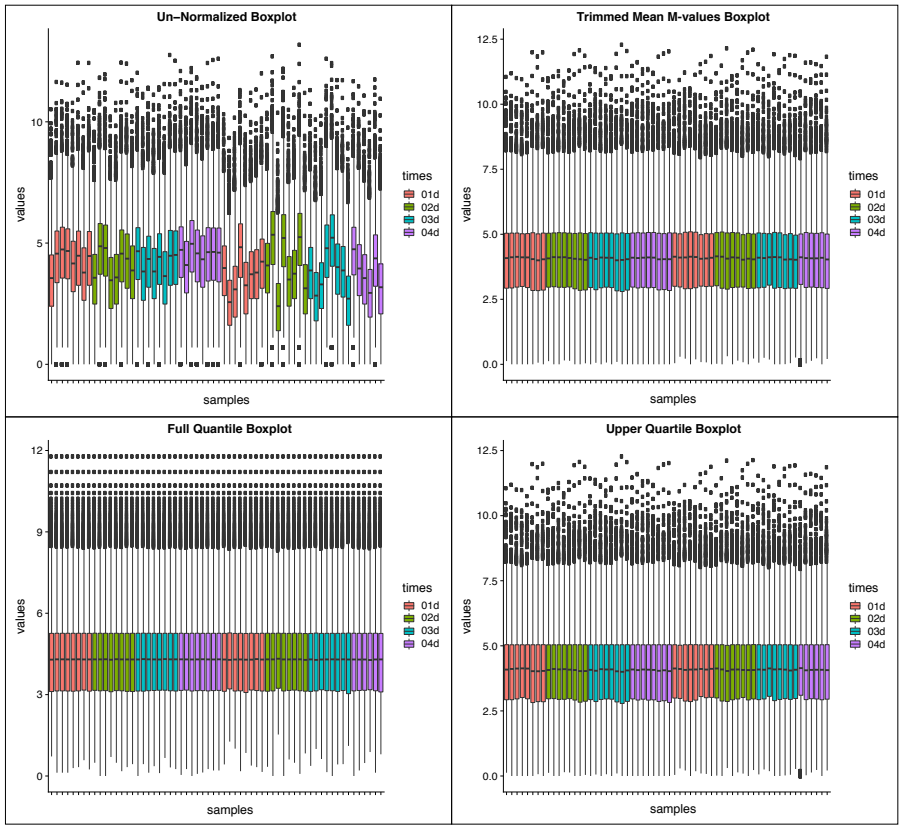


Figure 2.4.3: A comparison of how the filtering methods available in *ticorser* affect the data. The first panel shows the row counts with low expressed genes, while the other boxes show the filtering effect on the samples.

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

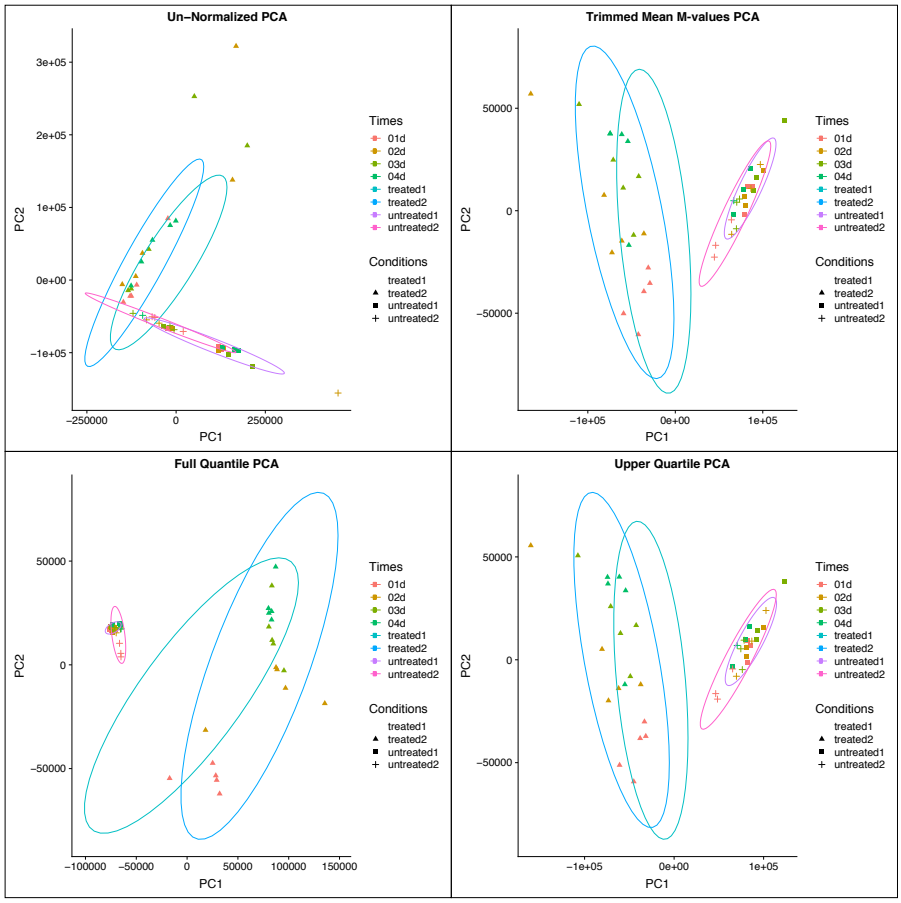


Figure 2.4.4: A comparison of how the normalizations methods available in *ticorser* affect the data. The first panel shows the clustering of row samples, while the other boxes show the normalization effect on the samples.

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 inter-

esting 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.

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: This table illustrates differences in catching DEGs between the three different methods on the same dataset (on the rows), 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.

It is also possible to inspect the *DE* results by plotting a *Volcano Plot* or an MA plot, as figure 2.4.5 shows.

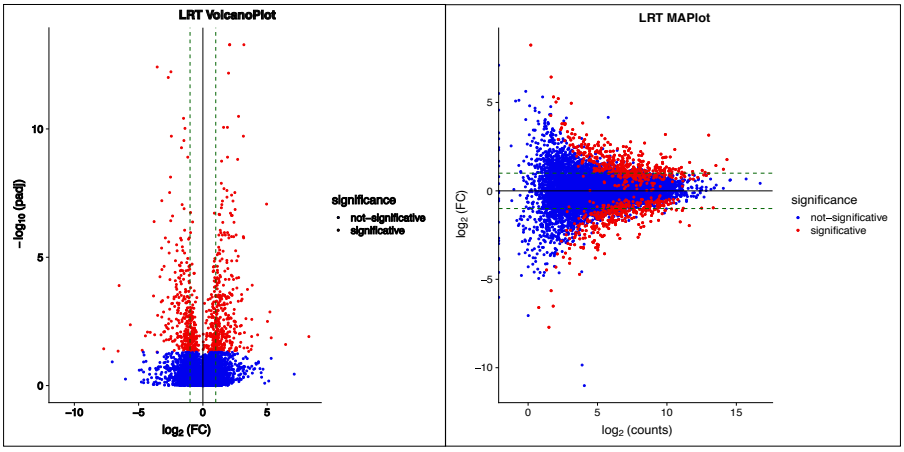


Figure 2.4.5: An inspection of the results using a volcano plot (on the left) and an MA-Plot (on the right).

In order to better compare the results coming from different analysis and methodologies approaches we can use one of the VENN diagrams available in *ticorser*, which, while intersecting the results, saves all the gene lists coming from the intersections and also from exclusions. Moreover, by setting the `enrich.lists.flag` to `TRUE`, the methods automatically starts functional enrichment for both pathways (on Reactome and KEGG databases) and Gene Ontology (on Biological Process, Cellular Components and Molecular Function), storing all the results in the `output.folder` specified (we remind to next section for further details on Functional Enrichment Analysis).

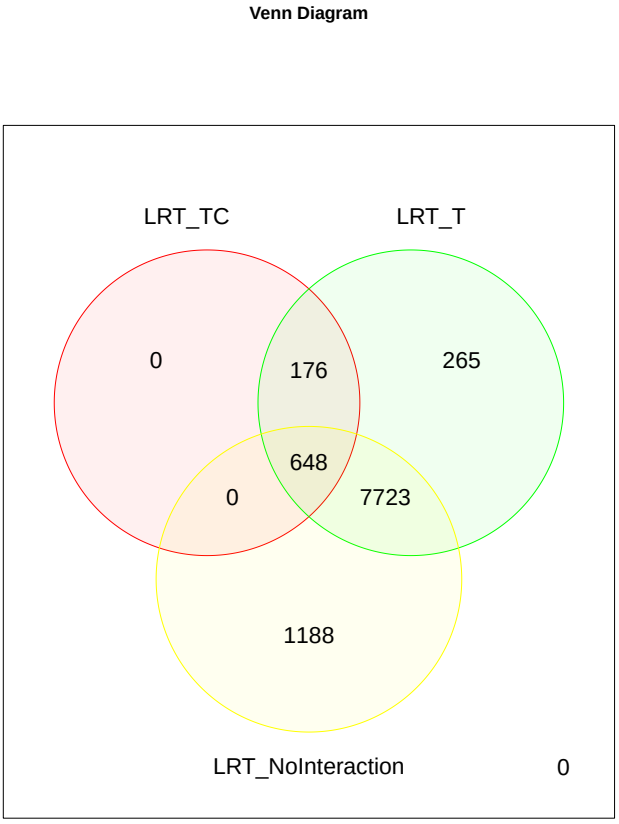


Figure 2.4.6: A VENN diagram to compare the DEGs detected by the three different TC methods of *ticorser*.

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.

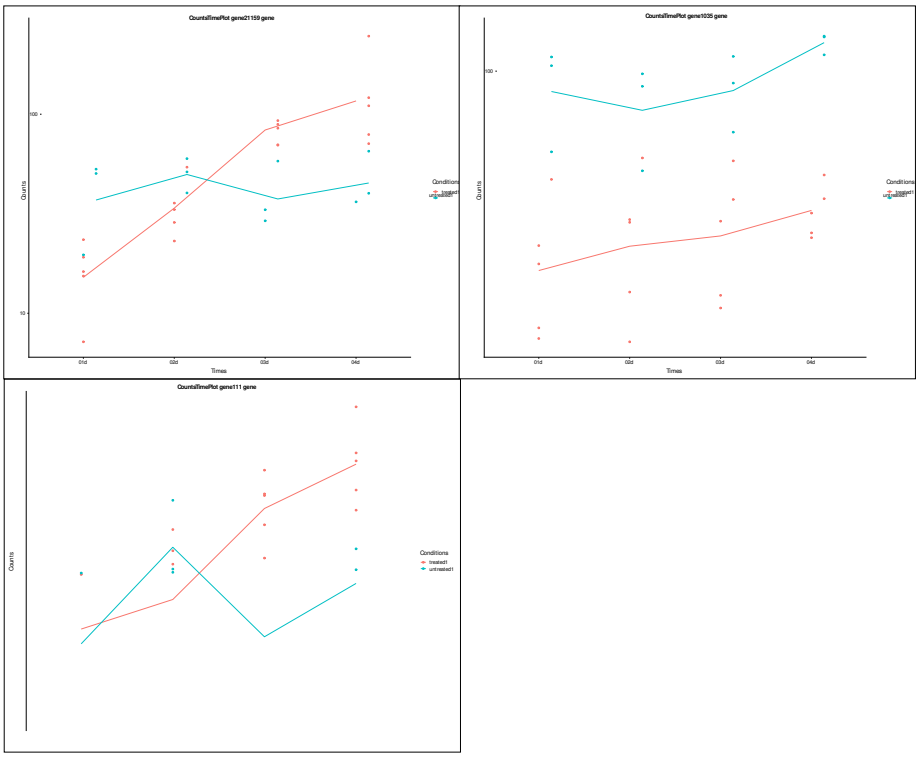


Figure 2.4.7: Three different trends examples of genes detected with *ticorser DE* TC methods.

For example looking at figure 2.4.7 we identified three different genes, one for each DE method, showing 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 s̈trangeb̈ehaviour, which is very oscillating across the time points and the conditions.

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

For the singular time point analysis, we can choose between three different methodologies, two present in the *NOISeq* package (accessible through the `ApplyNoiseq` function), while the third one is the *Wald test* present in the *DESeq2*

package. It is a good practice to use more than one method while working with transcriptomic data in order to be more confident 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: A comparison of the number of DEGs detected with *ticorser* at single time point.

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.

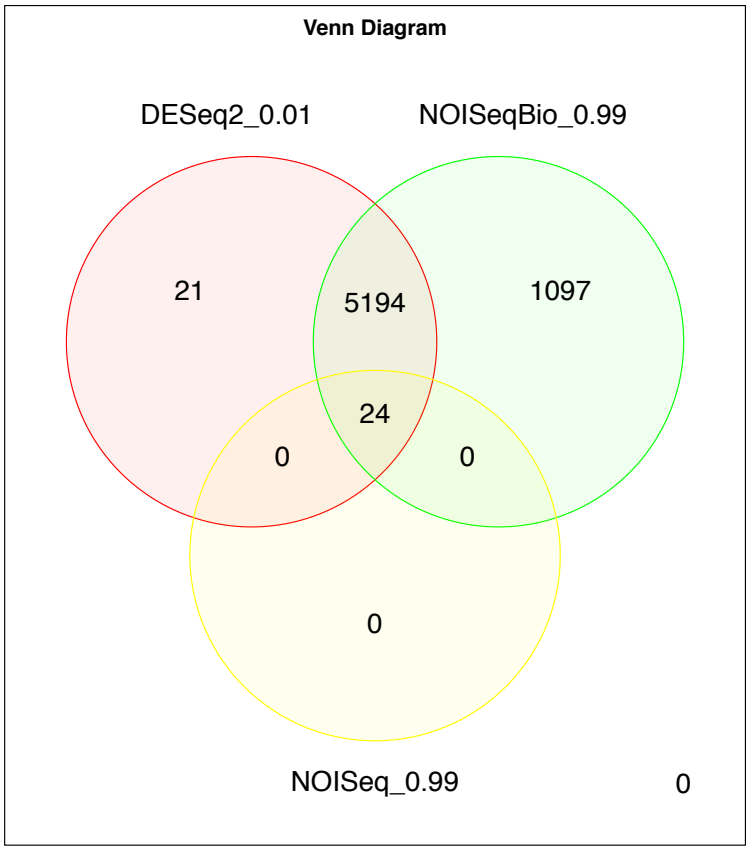


Figure 2.4.8: A venn diagram for comparying results coming from different methods used for DEG detection.

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 **??**)

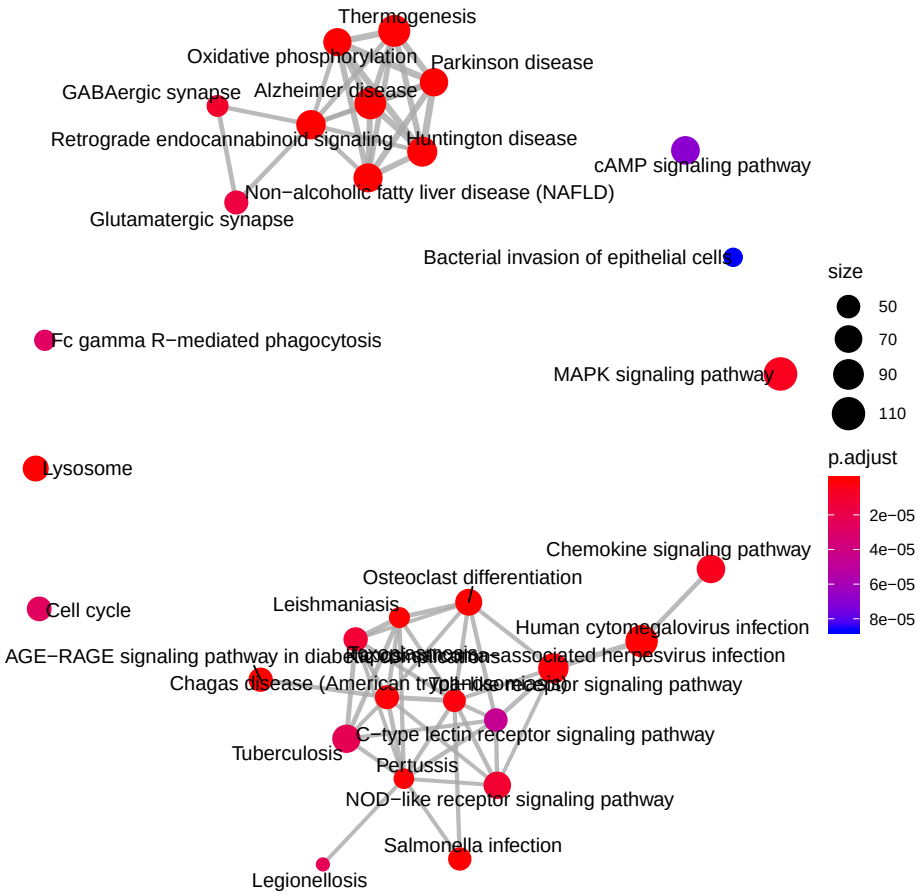


Figure 2.4.9: A network representation of the kegg results obtained with *clusterProfiler*package, by using *ticorser*

On the other hand, to perform a Gene Ontology functional enrichment analysis we use `enrichGOGProfiler` three times, one for each class of the `ontology`, `CC` for Cellular Components, `BP` for Biological Processes and `MF` for Molecular Functions. Analogously, we use `enrichGOFunction` for performing the same analysis with *ClusterProfiler*, which produces not only the table of the results but also a tree for the Gene Ontology terms significative in our dataset. (see figure 2.4.10).

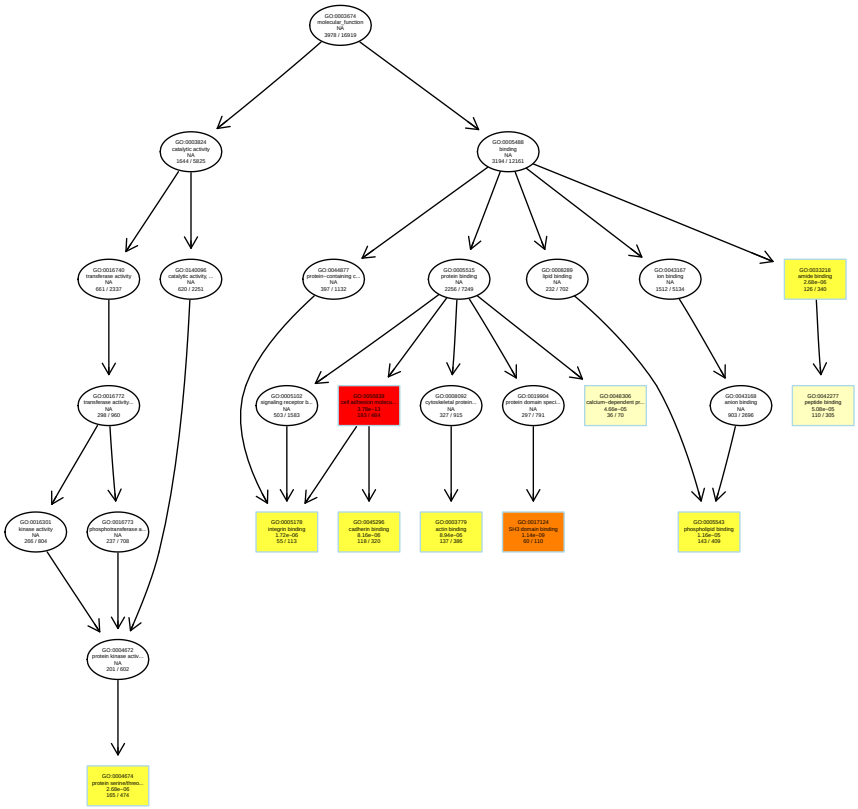


Figure 2.4.10: A hierarchical graphical representation of Molecular Function GO-terms obtained with *clusterProfiler*, by using *ticorser*. Graphical colours, from red (most significant) to yellow (less significant) indicates the p-value significance of each GO-term. Gray ones, are not significant at all.

Finally, when we identify one or more interesting *KEGG* pathways, we can 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 [71, 72].

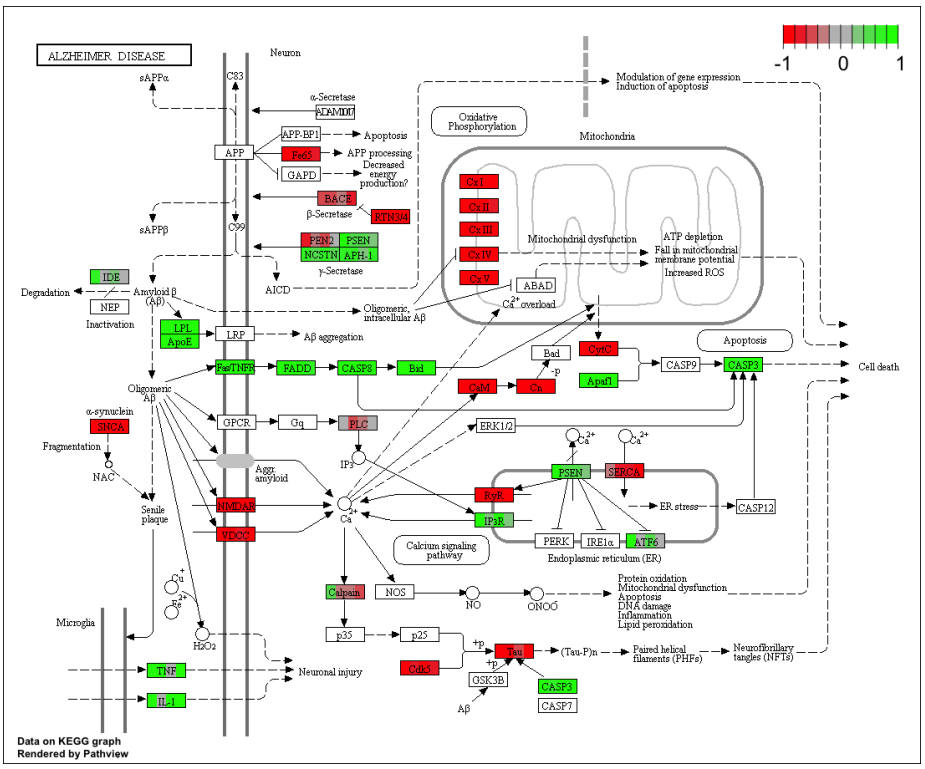


Figure 2.4.11: A keggmap for Alzheimer disease. Each DEG is coloured depending on the log2(fc) of its expression. Moreover, for each highlighted gene, the box is divided in 4 "mini-boxes", one for each time point.

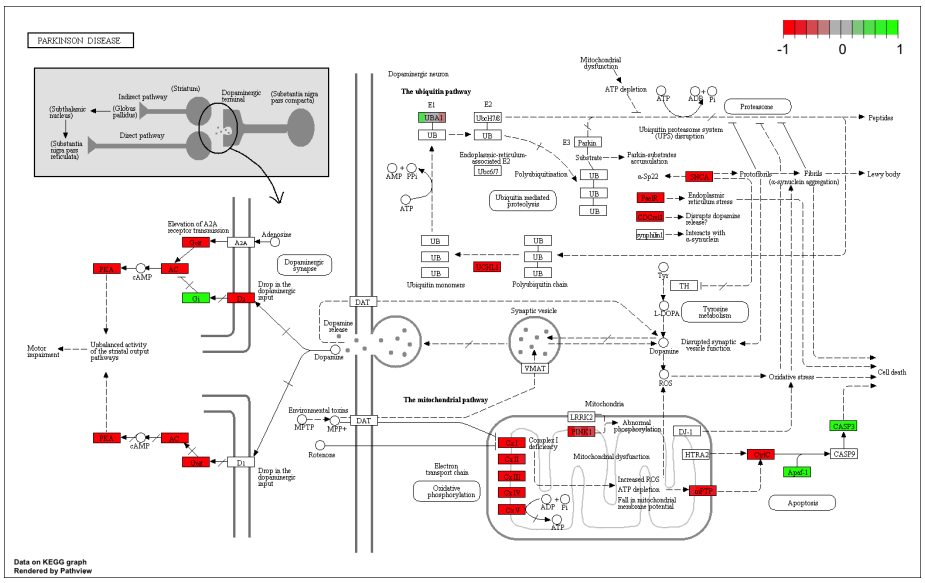


Figure 2.4.12: A keggmap for Parkinson disease. Each DEG is coloured depending on the log2(fc) of its expression. Moreover, for each highlighted gene, the box is divided in 4 "mini-boxes", one for each time point.

2.5 Discussions

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 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 (section 1.1), is a pretty wide and complex field, and the sequencing technology to adopt depends on the biological question under investigation.

Some studies [34, 73] 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 [34] 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 [74] 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 [52] 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.

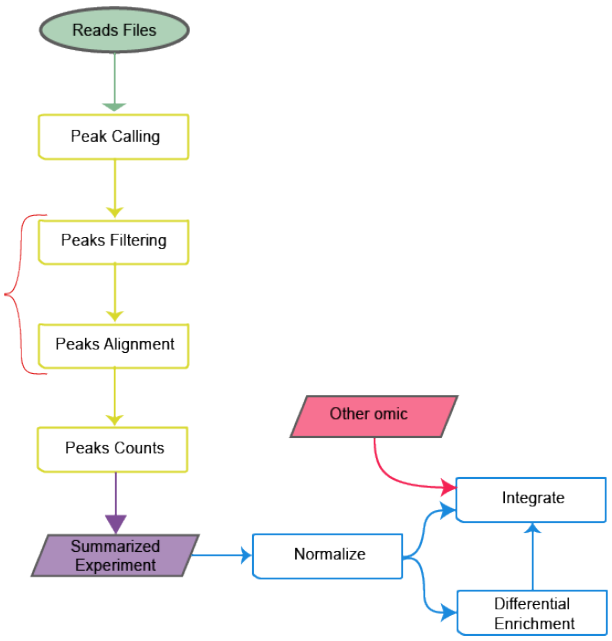


Figure 3.2.1: A differential enrichement flow representation. *DEScan2* steps are highlighed in yellow.

3.2.1 Peak Caller

The Peak Caller (defined by the `findPeaks` function) takes as input a set of alignment files (BAM [15] or BED format) with the code identifier of the 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* [75], 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.

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 [76], 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* [77] 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 settin 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* [78] 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 evoluted technique which enables to investigate the open chromatin regions at whole genome level. The capability of this technology has been demonstrated in the regulation of mouse brain activity under different conditions [79].

To illustrate the performances of *DEScan2* we chose a dataset [79] 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.

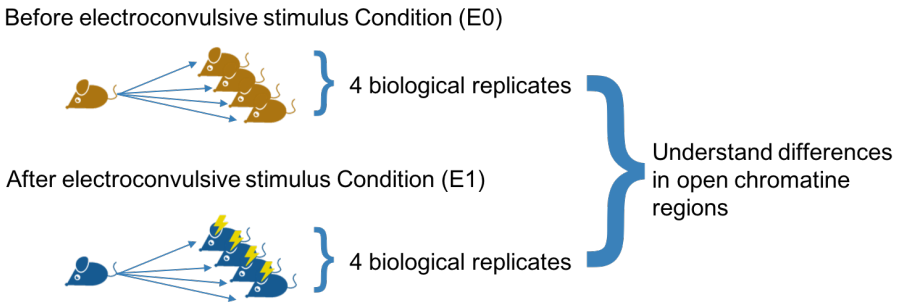


Figure 3.3.1: An illustration of our extraction of the GSE82015[79] dataset.

We downloaded the data from Gene Expression Omnibus (GEO) database [80, 81] with accession number GSE82015[1] and mapped the raw data using *STAR* [12] 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.

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

To be confident with our results we run *DEScan2* and *MACS2* [76] 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.

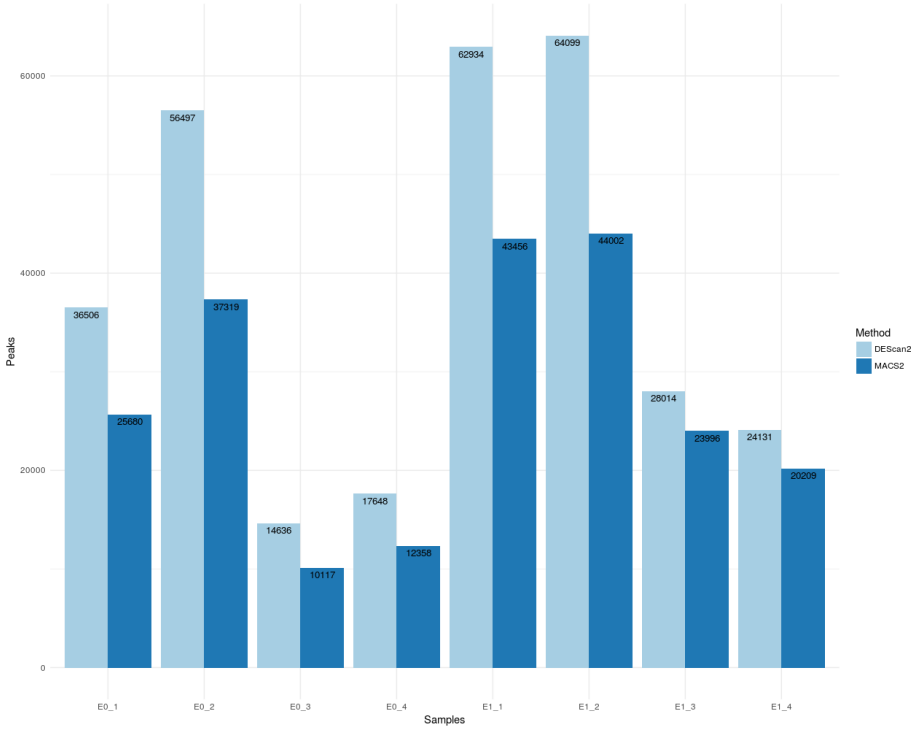


Figure 3.3.2: A comparison of *DEScan2* and *MACS2* detected peaks for each sample in the dataset.

To be more robust, we compared *DEScan2* detected peaks with the same validated regions (*Arc*[2] and *Gabrr1*[3]) of the original work [79]. The lower part of figure 3.3.3 shows the detected and validated regions (in blue and red) resulting differentially enriched between the E0 (in pink) and E1 (in green) conditions, while the upper part shows *DEScan2* filtered and aligned peaks (in blue) between the samples, highlighting a capability to catch not only the same regions of the published ones but also (gold circles) to be more accurate in the smaller peaks detection.

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

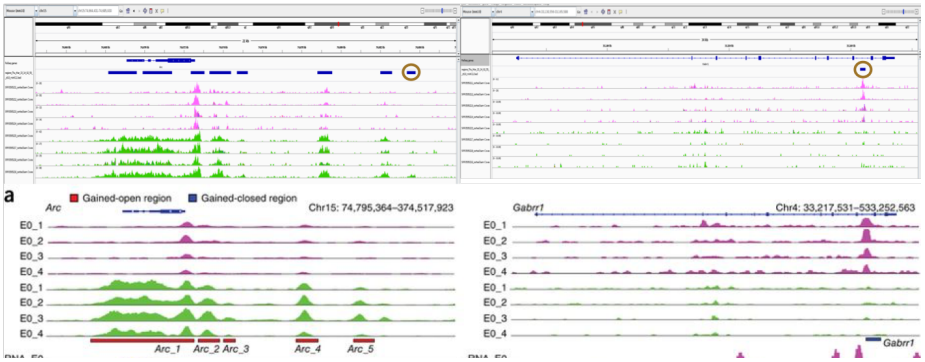


Figure 3.3.3: A comparison of *DEScan2* detected peaks with validated peaks in article [79].

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.

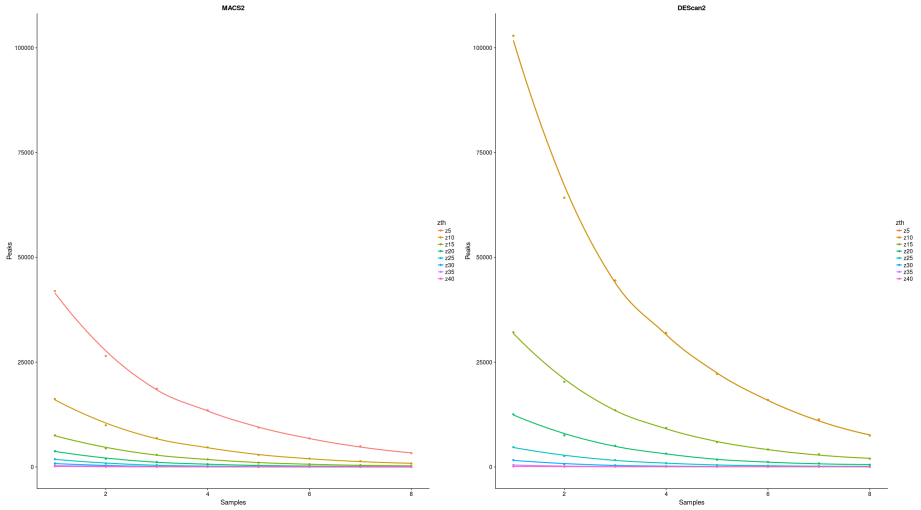


Figure 3.3.4: Filtering the detected regions with different thresholds on peak scores between *MACS2* and *DEScan2*.

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*.

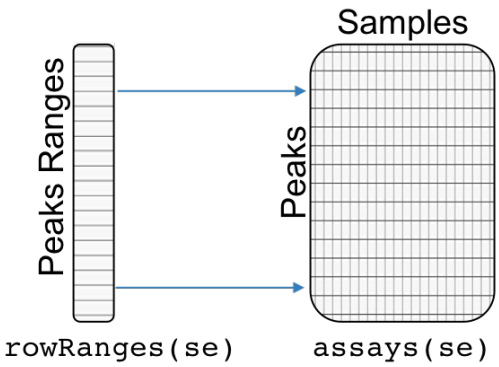


Figure 3.3.5: An illustration of the *SummarizedExperiment* data structure produced by *DEScan2*.

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 [68]. 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 [60, 82]. 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.

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

To better compare the normalization effects, we created a *null dataset* of 8

samples, by shuffling the original dataset samples as combinations of conditions took in pairs. The detection of DERs has been performed on each combination (18) of shuffled samples and then taking the median of the results.

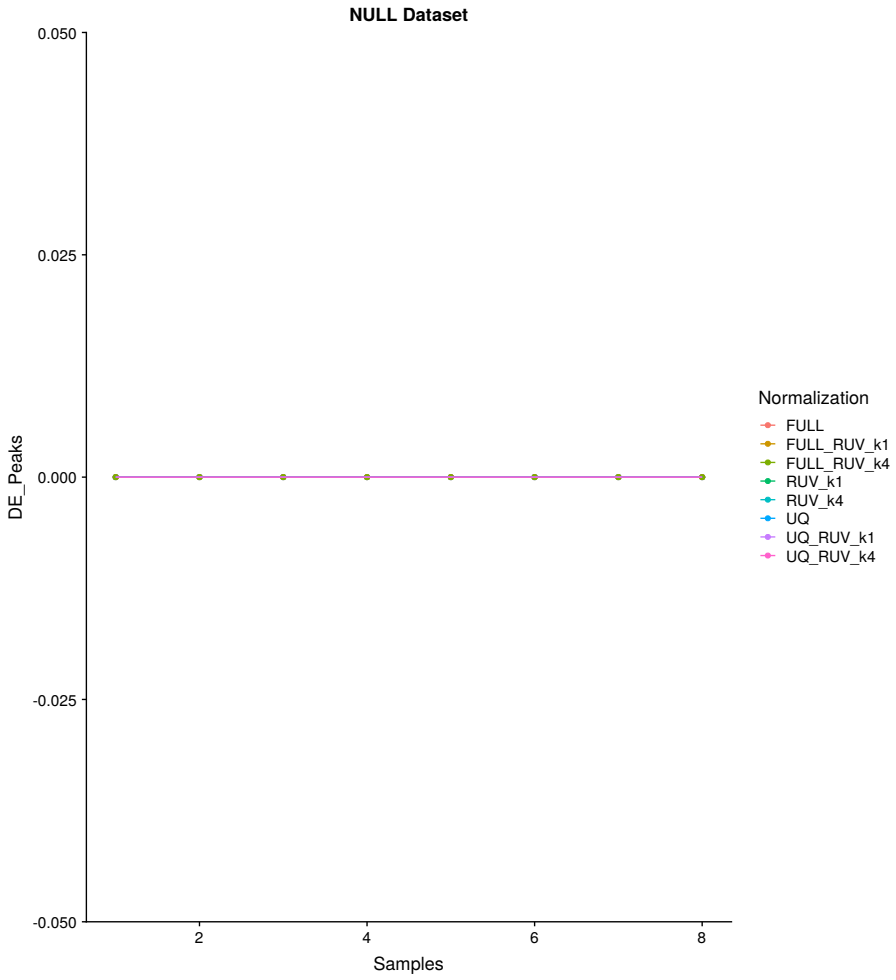


Figure 3.3.6: The figure shows the effects of different normalizations on a null dataset of epigenetic regions, putting in relation the DERs with the threshold on the samples used to align peaks.

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

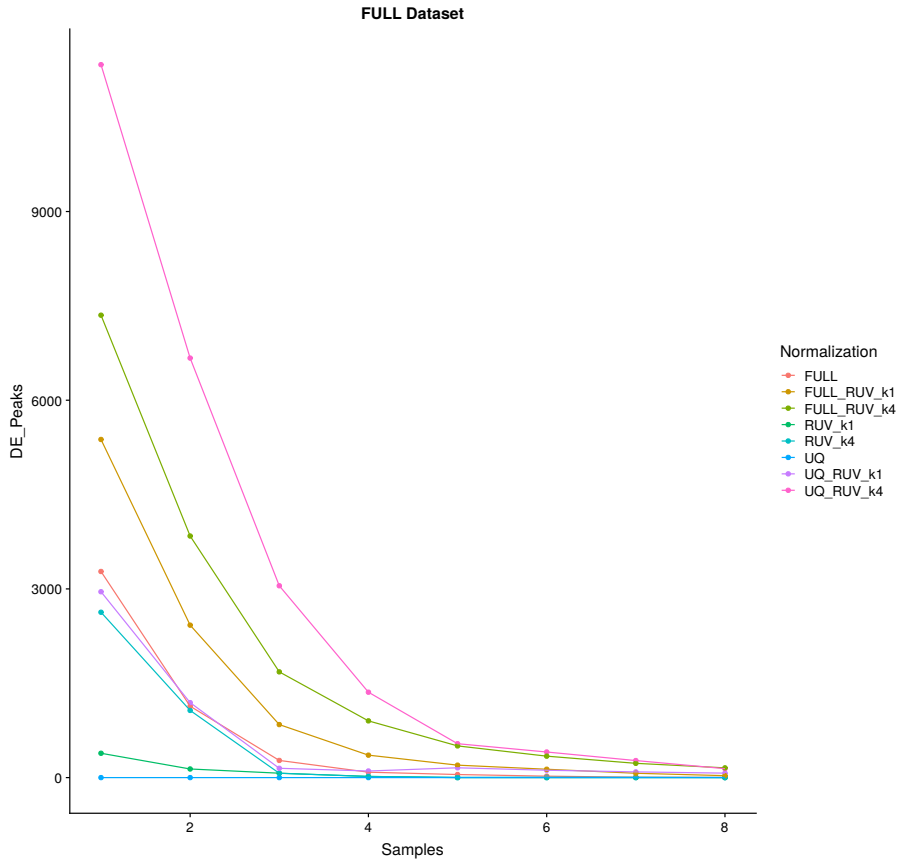


Figure 3.3.7: The figure shows the effects of different normalizations on a dataset of epigenetic regions, putting in relation the DERs with the threshold on the samples used to align peaks.

While figure 3.3.7, representing the "full dataset", shows that *upper-quantile*, by itself is not able to linearly detect any amount of DERs. When using *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 [59, 63, 83].

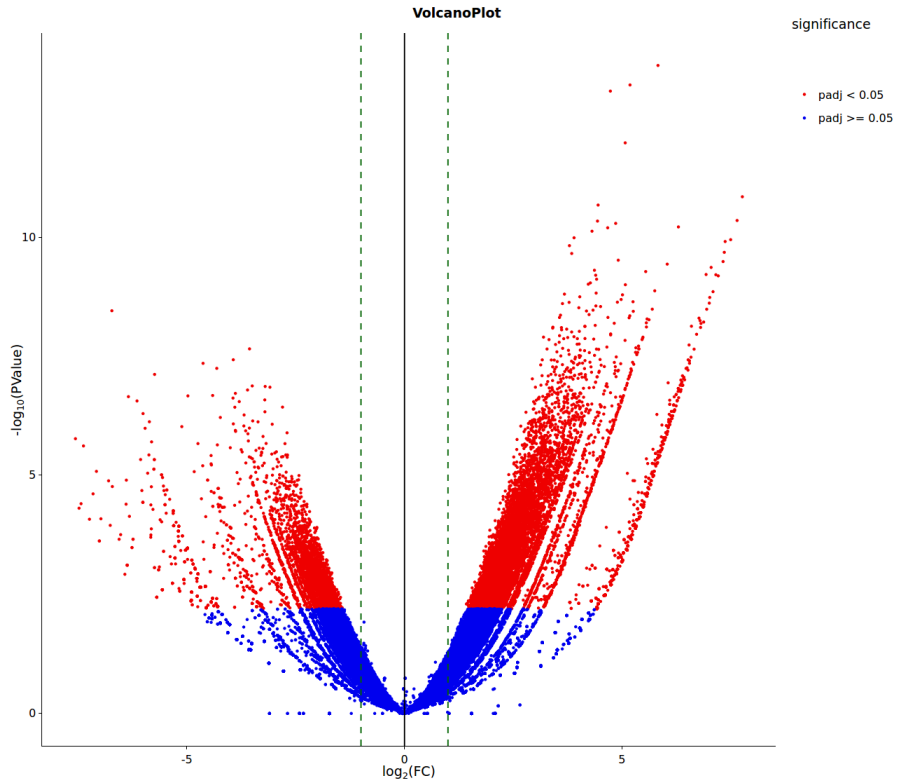


Figure 3.3.8: A volcano plot of Differential Enriched Regions. Blue dots represent the not significant DERs, while the red ones represent the significant DERs.

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[63].

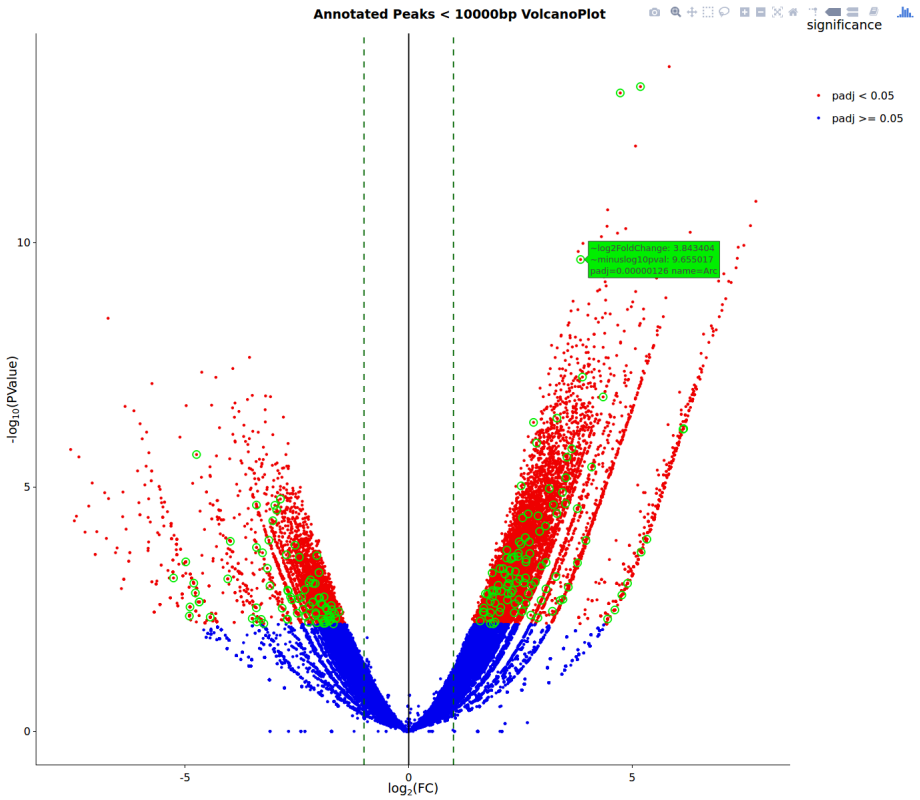


Figure 3.3.9: A volcano plot of DERs. Blue dots represent the not significant DERs, while the red ones represent the significant DERs. Green circles highlights the peaks with a DEG annotated.

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)[84] lower than 0.05, while blue dots highlight non significant regions.

Peaks Integration

The next task is to integrate the obtained results with other omic data types, as *RNA-seq*. Because of the low number of the samples, the easiest way to integrate the data is to annotate the DERs with DEGs resulting from the analysis of

RNA-Seq.

For the differential expression of the *RNA-seq* data, we first quantified the signal with the `featureCounts` methods available in the *Rsubread* [66] R/Bioconductor package. Then we filtered lowly expressed genes with the *proportion* test as implemented in *NOISeq* package, and applied the `noiseq` method for differential expression.

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

Then, to obtain a seconf level of data integration, we used the annotated genes to do functional annotation on Gene Ontology (GO) [55, 56] and Reactome pathways, which showed several interesting results for the neuronal regulation.

3.4 Discussions

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 *SummarizedExperiment* 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 tool: the R6 Class easyReporting

It has been claimed that many research findings in omics science are false (or partially false) due to accidental mistakes or mis-usage of methods. To prevent misleading results it is important to be able to inspect and reproduce the entire data analysis carried out in a unified product.

RR consists in making available both the analytic data and the associated code, so other researchers might reproduce the findings.

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

4.1 The idea of Reproducible Research

During last years, several approaches [28] have been proposed for helping to trace the analysis steps, using different programming languages; such as *Jupyter*[1] [85] in *Python* or *Rmarkdown*[2] in R. Or by building a web environment to

[1] `https://jupyter.org/`
[2] `https://rmarkdown.rstudio.com/`

encapsulate several tools made with different programming languages, such as *Galaxy*[3] [86–88].

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 (Literate Statistical Programming [89]). Additionally, it is mandatory to provide the analytic data needful to run through again the entire analysis. In such a way, a mixture of explaining natural language sentences combined with code instructions executed on the analytic data lead to produce a final product that can be easily reused also by a non-expert user.

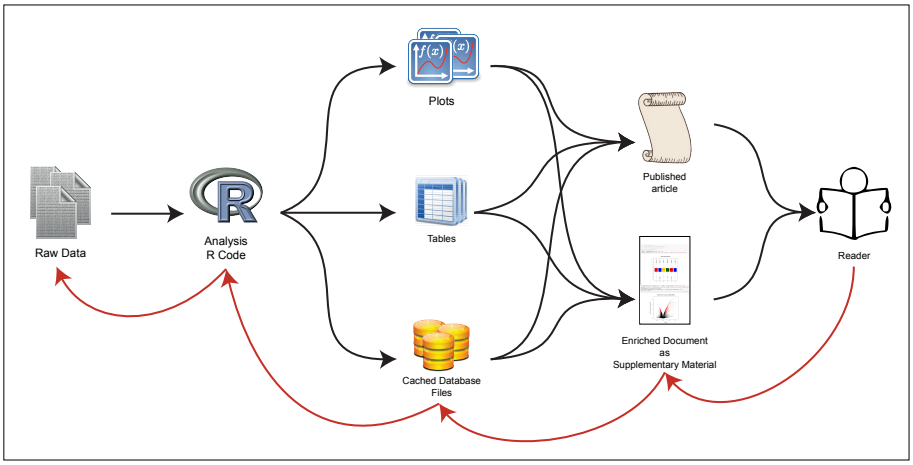


Figure 4.1.1: A general illustration of Reproducible Research concept. Raw data can be analysed with R code, producing plots, tables and additional caching database files. All together, R code, data and results can be inserted in an enriched document, which can be added as supplementary material to a valuable published article.
Image adapted from [29]

During last decades RR idea has been improved by the scientific community proposing several approaches in order to provide more accessibility to the analytic data, the code and the results at lower costs in terms of time and efforts. In particular the R community proposed several solutions, like *sweave* [90, 91] before, *knitr* [92] and *rmarkdown* later. Due to its easy interactive usage, *rmarkdown* became one of the most used instruments inside the R community, but its

[3] `https://usegalaxy.org/`

usability when developing automated instruments like Graphical User Interface (GUI) or packages becomes more difficult, leading developers to give up on using it.

In order to facilitate the widespread of RR through the analysts community, in the past we proposed a possible solution with the *RNASeqGUI* [28] project, a graphical interface software for analyzing *RNA-seq* data. The software, while the user interacts with the interfaces to analyze the data, traces the code inside an R Markdown (RMD) file and stores the data inside specific caching database files (CDF). At the end of the analysis, the RMD and the CDF contributes together at the compilation of the report, where all the code, the data and the results where presented.

This solution, although very useful, does not allow to insert personal comments at each CC. Indeed, each portion of the code was always automatically accompanied by default comments, that, to be edited required some experience by the user in RMD file editing. However, inserting personal comments for the CC is fundamental for the final knowledge transfer across researchers and lab members across time.

4.2 Methods

Here we present *easyReporting*, an *R6*[4,5] class[6] helping 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.

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

[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)`

supplementary material, helping the interested community to entirely reproduce the computational part of work (figure 4.1.1).

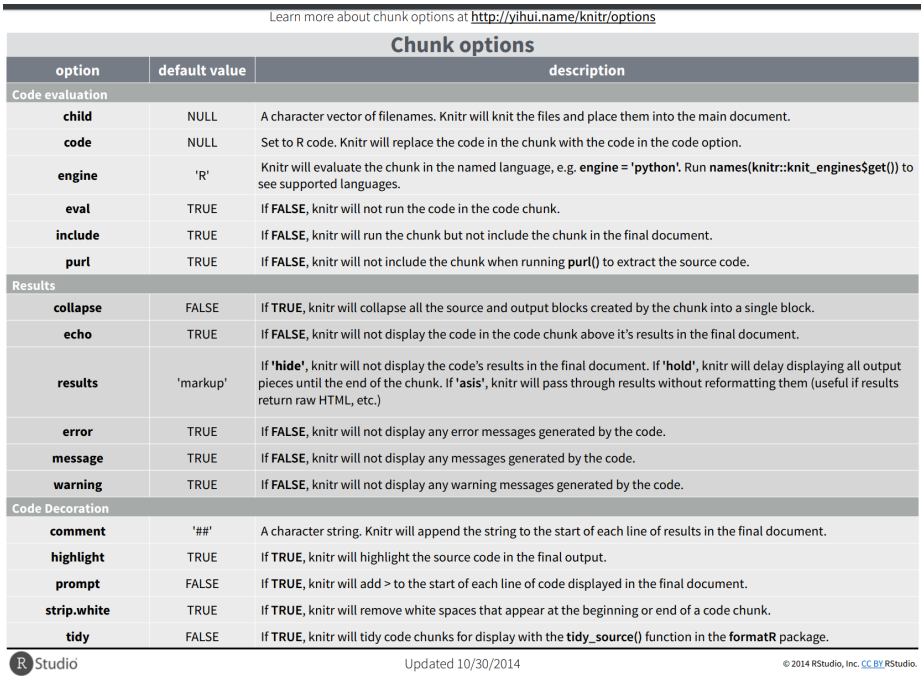
The package is accessible at the following link:
https://github.com/drighelli/easyreporting

General Description and Initialization

The class can to be imagined as a schematic representation of the *rmarkdown* file (*report*), indeed its attributes represent the *report* characteristics, which are typically inserted into the header of the file. But our class methods are not only for the attributes manipulations but also for insertion of CCs, comments and 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 4.2.1 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`.



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

Chunk options

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

Figure 4.2.1: A schematic table of options available using knitr and rmarkdown packages.

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 re-

cursive 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.

As already mentioned, the class allows to report not only the CC, but also to add personal titles and personal comments, enhancing the knowledge transfer between the "back-side" (the analyst/developer) and the "front-side" (the reader) users. Note that "back-side" and "front-side" can also be the data analyst and the collaborators, respectively.

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

```
22 ## Moreover I can add a list of files to source in che code
       chunk
23 rd$mkdCodeChunkSt(optionsList=optList, source.files.list=c("
       R/cachingFunctions.R", "R/cachingFunctions.R"))
24 rd$mkdCodeChunkEnd()
25
26
27 rd$mkdCodeChunkComplete(message="a <- 1\nb <- 2\nc <- a+b\n
       print(c)")
28
29
30 ## Otherwise I can make a direct call with all the code
       chunk and the comment in one call
31 optList <- makeOptionsList(includeFlag=TRUE, cacheFlag=TRUE)
32
33 rd$mkdCodeChunkCommented(commentMsg="This is the comment of
       the following code chunk",
34                          message="a <- 1\nb <- 2\n(c <- a+b)",
35                          optionsList=optList,
36                          source.files.list=NULL)
37
38 ## finally I can directly compile my report
39 rd$compile()
```

The previous R script leads to automatically produce the following *rmarkdown* file, where it is possible to see at rows 13 and 23 two titles at different levels. And additionally, starting from row 15 a describing paragraph, representing the additional comments that can be added to provide additional explanations to the following CC.

```
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,
       message=FALSE, include=TRUE, cache=TRUE)
11 ```
12
13 # First Level Title
14
15 Here I'm writing a simple paragraph useful to describe my
       code chunk
16
17 ```{r eval=TRUE, echo=TRUE, warning=FALSE, message=FALSE,
       include=TRUE, cache=TRUE}
18
19 variable <- `variable`
20 print(variable)
21
22 ```
23 ## Second Level Title
24 ```{r eval=TRUE, echo=TRUE, warning=FALSE, message=FALSE,
       include=TRUE, cache=TRUE}
25
26 ```
27
28 ```{r eval=TRUE, echo=TRUE, warning=FALSE, message=FALSE,
       include=TRUE, cache=TRUE}
29 source("/Users/inzirio/Desktop/gDrive/works/coding/
       easyreporting/R/cachingFunctions.R")
30 source("/Users/inzirio/Desktop/gDrive/works/coding/
       easyreporting/R/cachingFunctions.R")
31 ```
32
33 ```{r eval=TRUE, echo=TRUE, warning=FALSE, message=FALSE,
       include=TRUE, cache=TRUE}
34 a <- 1
35 b <- 2
````

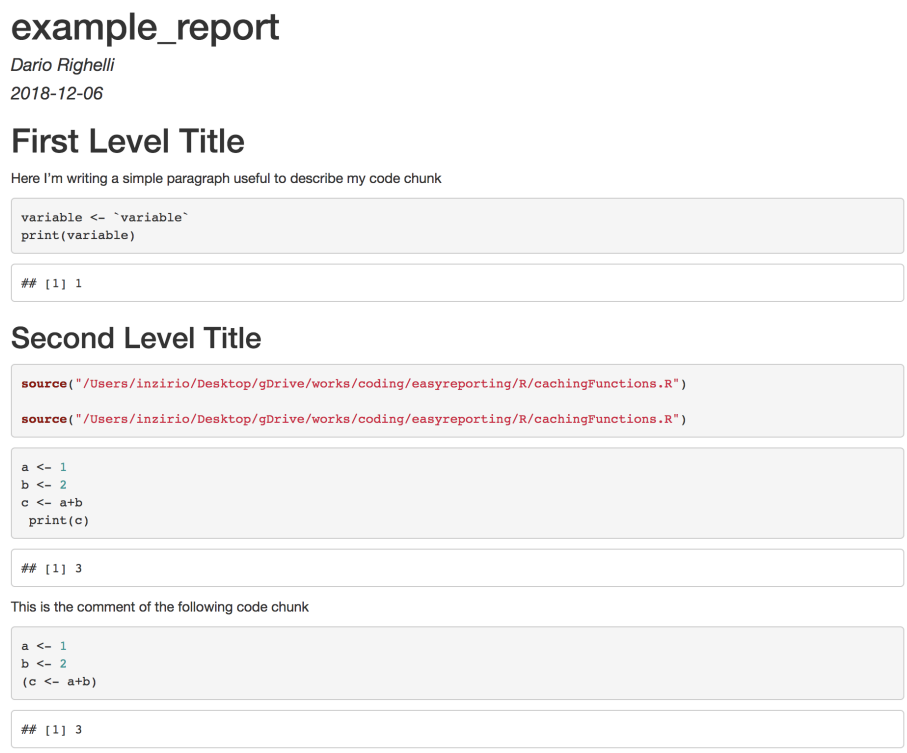
````
c <- a+b
 print(c)
```

This is the comment of the following code chunk

```{r eval=TRUE, echo=TRUE, warning=FALSE, message=FALSE,
    include=TRUE, cache=TRUE}
a <- 1
b <- 2
(c <- a+b)
```
````

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

Finally, the report in Hypertext Markdown Language (HTML) format provides a human readable document, where all the CC are highlithed in grey, and the results in white. Titles and comments are in plain text, highlighting the natural language format of a self-explaining document.
````



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 of HTML report produced with *easyReporting* R package.

4.4 Discussions

We presented *easyReporting*, an R package enabling the implementation of RR using *rmarkdown* language scripting, without any knowledge of it.

Despite other previously proposed solutions [93, 94], that always require too much commitment by the final user, potentially bringing him/her to totally renounce to include RR inside the scripts, our approach is versatile and easy to be implemented, leaving maximum freedom to the final developer/analyzer, automatically creating and storing an *rmarkdown* document, and providing also methods for its compilation.

It gives the possibility to obtain reproducible results and to publish better findings, increasing the scientific publications transparency by improving the knowledge transfer, and to support the methodologies integrity also into the scientific labs, where people can change position through the years.

On the other hand, the *easyReporting* package, 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, wants 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.

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 also reported in figure 4.1.1. Additionally, a caching store system (such as BiocFileCache[7]) could produce caching database files, easily shareable through the Internet or as supplementary data of a publication.

In order to provide a graphical representation of an entire analysis, it 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 final reader.

[7] `https://bioconductor.org/packages/release/bioc/html/BiocFileCache.html`

Chapter 5

Integration of High-Throughtput Omics data: IntegrHO

In previous chapters, we have described few tools (*ticorser*, *DEScan2*) for the analysis of specific experimental conditions. However, the practical usage of such tools requires a certain level of programming language from a user.

Moreover, it is clear that as soon as the analysis becomes more complex, and different types of integration have to be performed, data anlysis becomes more consuming and requires higher computational and statistical knowledge.

To face such limits, we noticed the growing interest by part of the scientific community in using interactive and user friendly software. This interest is led by multiple motivations, such as the need to analyze data very fast or the lack of time in learning programming languages and terminal-line based tools.

Several interface-based tools [95] have been proposed during last years but too often they are oriented to analyze singular-omics (such as *RNASeqGUI* [28]). However, in many cases such tools are organized as rigid pipelines, rather than interactive environments.

In this chapter we introduce Integration of High-Throghput Omics data (*IntegrHO*), our web-based platform for multi-omics data analysis and integration, built in a RR spirit.

5.1 Introduction

In order to deeply understand and reconstruct cellular mechanisms influenced by drug treatments, pathologies or diseases, it is fundamental to look at multiple omics data types at the same time.

Previous chapters already described some tools for multiple omics data analysis, integration and visualization using command line tools. Although, the command line approach is quite common inside the bioinformatics community, it is not weel suited for every scientist who is not so confident with programming languages or with the 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. Additionally justified if we think about the technical knowledge required when working with multiple different input/output data structures required for each tool.

Furthermore, even if the bioinformatics community has massively moved on the development of novel statistical and computational methods for multi-omics data integration, part of the scientific community is still anchored to the single-omics analysis side. Indeed, without entering into the merit of the obtained scientific results, it is still very common to read published papers based on single-omics data analysis without taking into account possible integrated solution with other omics data types.

Nowadays, it is more easily achievable to afford for multi-omics experiments because of the lowering costs of them. Additionally, there are international big projects for collecting biological data and providing public access to their biological data banks, such as GEO[1] [96] or The Cancer Genome Atlas (TCGA)[2] [97], where it is possible to retrieve as many data as needed.

On the other side, multi-omics data integration is an emerging research field, that still needs to become a standard of biological data analysis.

[1] `https://www.ncbi.nlm.nih.gov/geo/`
[2] `https://cancergenome.nih.gov/`

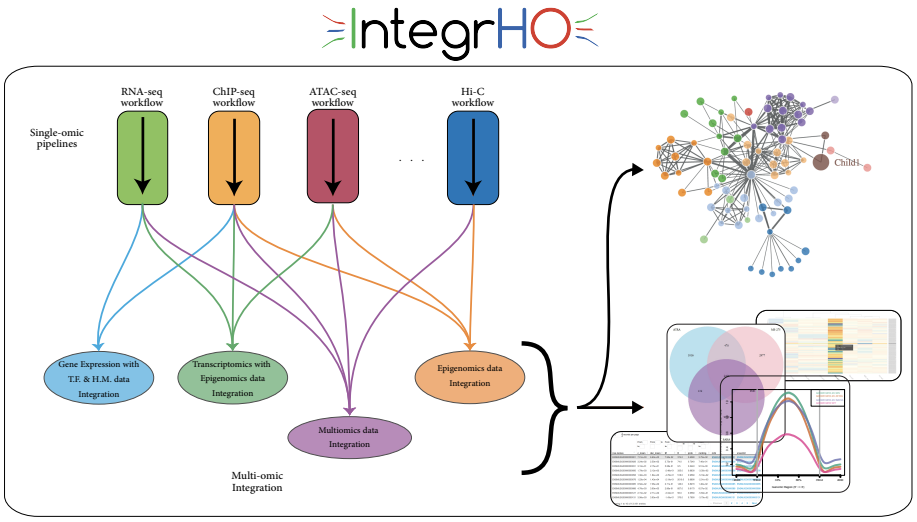


Figure 5.1.1: A schematical representation of *IntegrHO* underlying idea. Single-omics analysis methods are proposed in order to facilitate their multiple integration. This integration can lead to produce graphical results, in case of low dimension datasets, or to more sophisticated integration model such as regulation networks, in case of high-dimension datasets.

Based on the above mentioned considerations and in order to promote the multi-omics data integration, we are developing *IntegrHO*, a novel easy-to-use instrument which not only gives the possibility to analyze single-omics data types but also guides the user through multiple ways of integrating multi-omics data types (figure 5.1.1 gives an underlying idea of *IntegrHO*).

5.2 Methods

Roughly speaking, when working with multiple omics data, the integration phase requires that the data used for this step are ”connected” between the different omics.

For example, if we want to investigate the epigenetic marks involved in the regulation of the gene expression of a particular disease, we need to collect data from *ChIP-seq* and *RNA-seq* experiments under the same experimental conditions. To do that, we need to collect as many samples as possible for each omics (because as higher is the number of samples as higher will be the power for estimating good results) to evaluate a certain number of variables/features (i.s. gene expression, methylation, TF, etc.) for each sample.

Moreover, when investigating multiple conditions, we need multiple replicates for each condition, in order to be able to well estimate and remove biases affecting the experiment and distinguish technical or biological variability as well as other sources of variation.

To summarize, when working with multi-omics data, we need to collect samples of the interested cellular line for the disease/condition we want to study. Moreover, to understand multiple mechanisms affected by the disease we need to collect multiple omics for each biological aspect (i.e. *ChIP-seq* and *RNA-seq*). Additionally, for each omics, we need to collect multiple samples for each condition (i.e. healthy and tumor). Therefore, typically multi-omics datasets consist in a bunch of short reads files (each of them corresponding to a specific sample in specific conditions). After some preprocessing (alignment, qunatification, etc.) results are combined in omics specific matrices X_k with $k = 1, ..., m$, where m denotes the number of omics, such that $dim[X_k] = [n_k, p_k]$, where n_k is the number of samples and p_k the number of variable osserved on the specific omics.

Once we collected all required samples, we can either start analyzing each omics individually and/or integrating them to better understand the cellular behaviours involved in the studied disease. Depending on the number of collected samples we can choose from different methodologies to apply, if we have few numbers of samples, we can decide to work on single omics analysis and then to integrate the results. Otherwise, when having a high number of samples we can work use deeper statistical methodologies, such as network fusion techniques for constructing multiple levels regulatory networks [36–38].

In order to account for these kind of studies, inside *IntegrHO* we are implementing several aspects and interfaces to help and guide the user to achieve integrated results.

5.2.1 The main 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 interacts 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 technology[3]. In such a way, the main shiny app can be shredded into 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).

[3] `https://shiny.rstudio.com/articles/modules.html`

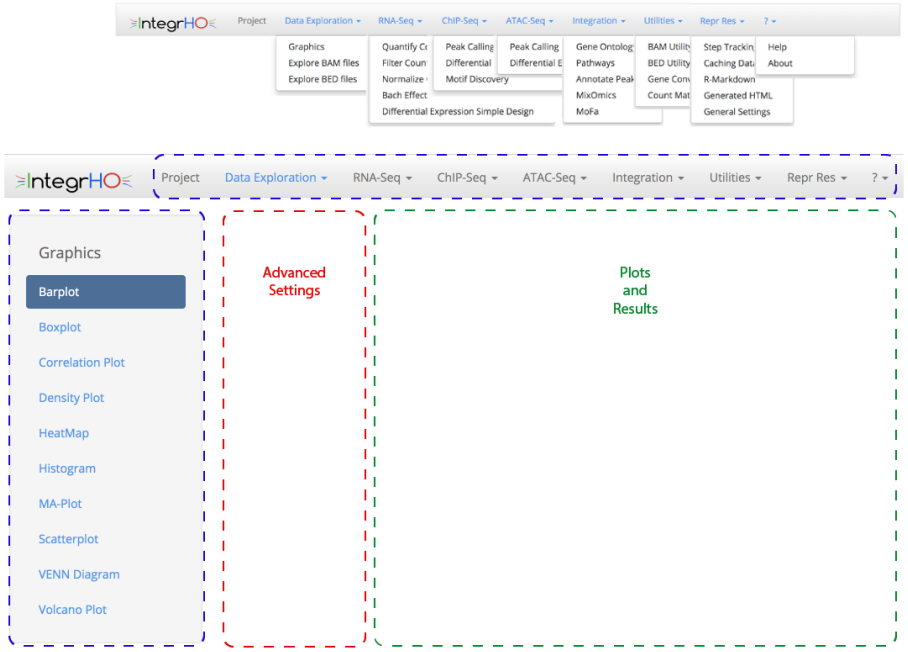


Figure 5.2.1: *IntegrHO* main interface description. A main menu in the upper part is presented with all the available main functionalities, and a left side menu is presented with additional functionalities (in blue). Red part indicates the settings for each functionality within the parameter setup.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

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

We decided to implement *IntegrHO* as a design file based software. Because of the high number of samples needed for the data analysis, the user needs to specify several variables describing each sample. This is useful for defining samples belonging to the same condition, and to specify which is the omics for each sample, and some additional describing variables. The design file information will be used during all the following steps of the analysis.

Before to proceed to the data analysis, it is mandatory for the user to set up the project with a dedicated interface. The user has to upload the design file which describes the information related to its samples. Some of them are mandatory, such 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 file directly from the interface (figure 5.2.2).

The choice of using a mandatory variable describing the path of BAM filenames is due to a *Shiny* library limitation. Because of its "UI-server" structure, shiny applications, when using a `fileInput` widget, make a temporary copy of the selected file(s). But this aspect becomes really problematic when a file can reach several gigabytes of memory on disk, as one or more BAM files can. Even if there are alternative packages, such as *shinyFiles* [98], offering the possibility to solve the problem, they are not very well tested over multiple platform architectures, bringing us to choose for this workaround solution.

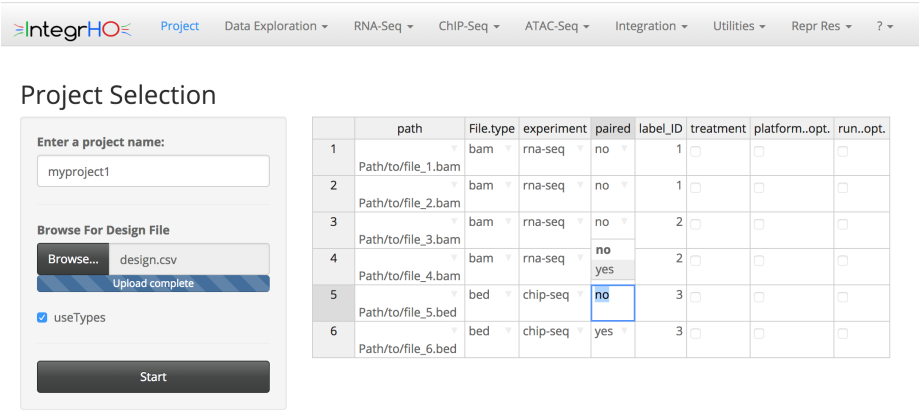


Figure 5.2.2: A screenshot of the project design *IntegrHO* interface.

Using the project interface, *IntegrHO* creates inside the working directory (returned by the `getwd()` function) a dedicated folder with all the required subfolders 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 (Figure 5.2.3 shows a representation of a typical folder tree produced for a project).

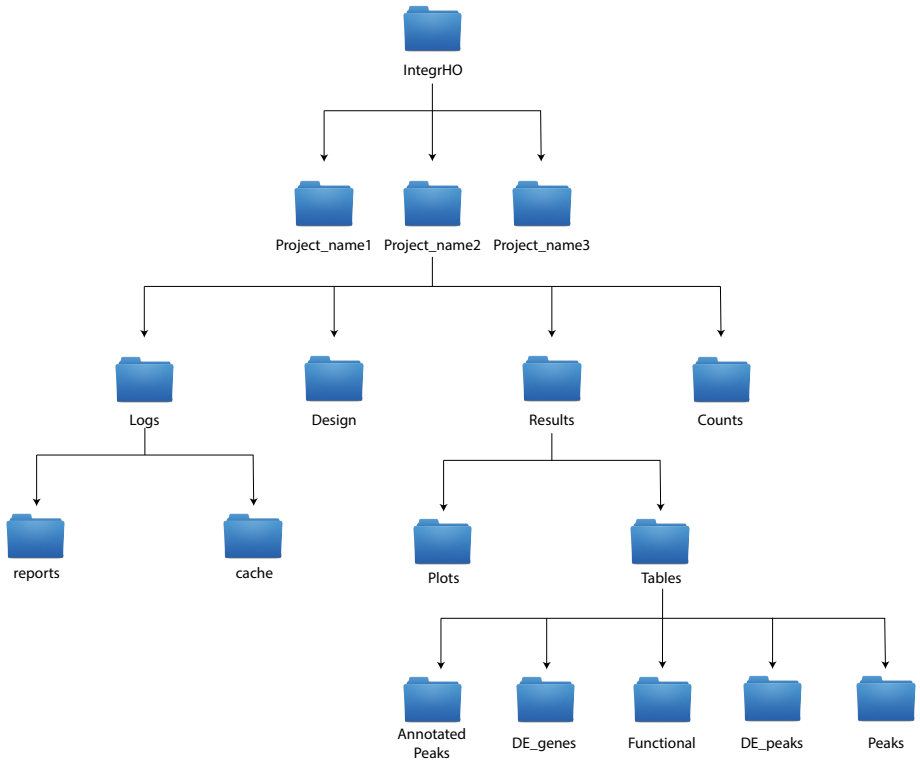


Figure 5.2.3: A schematical representation of *IntegrHO* project folder tree.

In particular, we designed a `R6ProjectClass`, which stores all the paths represented inside the figure 5.2.3. This allows fast access to the files stored in each folder, that will be processed specifically for each selected interface.

5.2.2 Available Methodologies

Unlike tools as Galaxy [99] and Taverna [100], focused on analysis workflows, *IntegrHO* gives to the user high freedom of interaction, reporting all performed steps in a human-readable *HTML* report.

Actually, *IntegrHO* methodologies have been organized in a way that the user can analyze every single omics, and afterwards, combine the results to obtain an integrated view of them.

In particular, we defined specific interfaces for *RNA-seq*, *ChIP-seq* and *ATAC-seq* data, providing additional interfaces for their integration, such as *functional enrichment* and *gene-peak* annotations.

Moreover, through the `graphics` interface the user have the possibility to

between 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*, *keggmap*, etc.

Inside *IntegrHO*, we take from granted that every omics data type is already mapped on a reference genome, with already available BAM files for every sample the user wants to investigate.

Figure 5.2.4 shows a schematic representation of implemented packages inside *IntegrHO*. The upper part shows the single-omics packages used for the analysis of *RNA-seq*, *ChIP-seq* and *ATAC-seq*, while the lower part refers to the multi-omics integration, showing the packages used for *functional analysis* and *peak-gene annotations*.

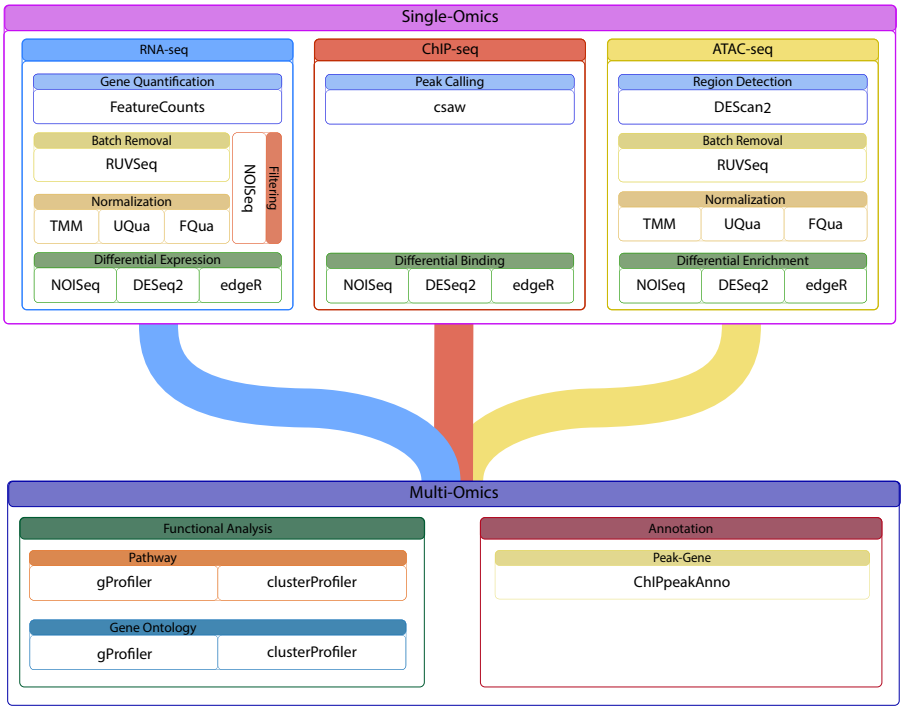


Figure 5.2.4: A schematical representation of packages included inside *IntegrHO* web-platform. The upper part shows the packages for the the single-omics analysis, while the lower part shows the packages for the multi-omics analysis.

RNA-seq

For *RNA-seq* we chose multiples methods for allowing the user to obtain a good overview of this omics analysis, focusing on *Differential Expression* aspect. In such a way, the user have several methodologies for accounting for a complete *RNA-seq* pipeline (refer to 1.2.1 section for a typical *RNA-seq* pipeline description).

When working with *RNA-seq* data it is really important to quantify the reads contained inside the BAM files associating them to genomic features. To do that, it is necessary a file with the genomic features description, typically a GTF file (defined in 2.4 section) describing the genomic coordinates for each feature (genes, transcripts etc.).

To account for genomic feature quantification, the user has to select the `Quantify Counts` interface, designed for `featureCounts` of `Rsubread` R/Bioconductor package function [57] appears. The BAM file paths for the *RNA-seq* data are automatically selected from the design matrix loaded during the project creation phase.

We choose proper widgets for selecting the reference genome and its annotation, it is also possible to configure the `featureType` to use, while the `attributeType` is set by default to `gene_id`. Additionally, parameters can be selected for `allowMultiOverlap` and to select if the reads are `paired`. Finally, the number of threads to use can be set from 1 to 8.

At the end of the processing, the resulting count matrix is presented in the main part of the interface as a `data.table` format, and automatically stored inside the `IntegrHO/project_name/counts` folder.

When working with *RNA-seq* data, several features are too low expressed to be kept, because their expression will affect the further analysis, leading to biased results. When selecting the `Filter Counts` the designed interface automatically loads the counts files inside the `IntegrHO/project_name/counts`, otherwise allowing, with a `fileInput` widget, to chose another count matrix file. It is possible to switch between three different filtering methods (*Proportion Test*, *Wilcoxon Test*, *CPM*), as defined inside the `filtered.data` function of *NOISeq* R/Bioconductor package [59].

The filtered-in features need to be normalized in order to be compared across

multiple conditions. Also in this case, a not proper data normalization can head to misleading results.

The same type of interface built for the filtering step is loaded when the `Normalize Counts` menu voice is selected. Three different normalization methods can be selected, the *Trimmed Mean of M-values* [101], the *Upper Quartile* and the *Full Quantile*.

Additionally, a main aspect of *RNA-seq* data analysis is to account for "unwanted factors" (batch effects) that, very often, affect the data. This noises are part of the data because reflecting batch, library preparation, and other nuisance effects.

When selecting the `Batch Effect` section there will be the possibility to apply the *RUVs* or *RUVg* methods from *RUVSeq* R/Bioconductor package [60]. Moreover, when selecting one of this methods, an additional file can be loaded, in order to upload an optional list of negative control genes.

To graphically inspect the normalized data two different plots will be presented into the main interface, the PCA and the *boxplots* for before and after normalization of the samples. This comparison is dictated by the need of inspecting the effects of noise removal from the data.

Finally, at the end of execution the filtered/normalized features, as tsv file format, are stored inside the `IntegrHO/project_name/counts` folder, adding at the end of the file name the "transformation" applied.

Finally, when the data are well normalized, the user can address for differential expression inspection of the samples.

In order to detect DEGs, we designed the `Differential Expression Simple Design`, where it is possible to choose between four different methods.

The available count matrix files are automatically loaded from the `IntegrHO /project_name/counts` folder. Otherwise, the user can additionally load another count matrix file using the apposite file selector.

In order to address for the factors to analyze, the interface automatically gives the possibility to select the design matrix column name where to find them. Additionally, two `selectInput` widgets appear where to select the two factors to contrast.

We selected four different methods implemented inside *edgeR*, *DESeq2*, *NOISeq*

and *NOISeqBio* R/Bioconductor packages [59, 62, 63].

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.

The dedicated interface automatically shows a `tabsetPanel` of 3 different tabs, within, respectively, a *VolcanoPlot*, an *MA-Plot* and the table of the DEG results.

ChIP-seq

ChIP-seq is a very common technique to investigate multiple epigenetic factors, the sequencing technique by itself identifies chromatin regions where specific proteins bind on. Considering to have multiple BAM samples of *ChIP-seq* data, we constructed specific interfaces for peak calling and Differential Binding Sites (DBSs) detection. Also in this case, we are focusing on the problem of detecting how the signal differs between multiple biological conditions. We refer to DBSs because of the nature of the biological data, coming it from proteins binding specific regions of the chromatin.

Assuming that the samples are in BAM format, we assume that the user needs to reconstruct the signal of the chromatin regions binded by the interested protein. We underline that a *ChIP-seq* experiment is "protein specific", because it involves using an antibody that is specific for a known protein.

For the **peak calling**, because of the lack of specific methods starting from BAM files in R, we implemented a dedicated interface for *csaw* [102] peak caller, which allows quantification for both broad and narrow peaks, typically resulting from HM and TF *ChIP-seq* data.

Firstly, it is possible to select the design matrix (stored inside `IntegrHO/project_name/design` folder) and the column specifying the experiment. In such a way, *IntegrHO* is able to pick up the BAM filenames associated to the *ChIP-seq* experiments by itself.

Afterwards, the interface enables the specification of several parameters useful for the `windowCounts` method of *csaw* R/Bioconductor package. The method

constructs a count matrix where the rows indicate the detected windows of length `window width` and the columns indicate samples.

Additional parameters selected for the analysis with the *csaw* method are the `paired end reads`, a flag for ignoring duplicated reads with `ignore duplicates`, and the possibility to make the analysis only on a list of chromosomes, using the `chromosomes list` implemented with a `selectizeInput`. The possibility to make parallel computing is implemented with the `core number` widget (from 1 to 8).

At the end of the processing, the count matrix is shown into the main part of the interface, and it is saved as tsv format, into the `IntegrHO/project_name/results/tables/peaks` folder. In order to distinguish the produced output from others, the results file is saved by adding the method name at the end of the filename.

Once constructed a count matrix of genomic windows (features) on the rows and samples on the columns, in order to detect DBSs across the conditions, we are able to apply the same methodologies designed for *RNA-seq* data (see section 5.2.2 for further details). For this reason, we used the same methods as defined for *RNA-seq*, showing the results table as output.

Other methodologies for differential binding have been developed in R, such as the package *DiffBind* [103], but at the moment we didn't implemented them yet.

Moreover, it is relevant for *ChIP-seq* data analysis to account for the "right normalization" to use. The count matrix form allows to apply the same normalization methodologies developed for *RNA-seq* data, but at the moment we didn't implemented this aspect. Additionally, it is still an open issue to study for the "correctness" of applied normalization on *ChIP-seq* data, even if some approaches have already been presented [104].

ATAC-seq

ATAC-seq is a sequencing technology for investigating chromatin accessible regions, which are typically associated on the identification of the instrumental epigenetic changes responsible for differential gene expression, cell proliferation, functional diversification and disease development (see 1.2.3 for further details).

Thanks to the work made with *DEScan2* (chapter 3), we had the possibility

to deeply investigate *ATAC-seq* data, presenting a specific analysis workflow. We tried to give access to the same methodologies, defined for *DEScan2*, herein *IntegrHO*, in order to simplify and spread the proposed work.

As for the *ChIP-seq*, the user needs to reconstruct the signal across the entire genome by detecting the genomic regions.

In particular, to account for the regions detection, we built a specific interface for the peak caller defined in *DEScan2* (see section 3.2.1).

As for the *csaw*, the user can select the design matrix (stored inside `IntegrHO/project_name/design` folder) and the column associated with the experiment definition. Additional parameter selection is required for the selection of the `genome name`, the `bin size` for the genome binning, and for the `max window` and `min window` lengths. Also, in this case, we gave the possibility to select for a multi-core computation with the `core number` sliderInput widget.

The user can also select the `peak score` threshold for filtering out the regions with a lower score than the threshold, and the `samples threshold` for the alignment phase (see section 3.2.2).

In order to better compare *DEScan2* results with similar methods, such as *casw*, we preferred to include also the *DEScan2* filtering method inside the *region detection* interface. In such a way, the presented output is a count matrix of detected regions (peaks) on the rows and samples on the columns.

After the reconstruction of the signal, it is a good norm to normalize the data, in order to remove unwanted factors biasing the data leading to misleading results. Also in this case, we took advantages of the count matrix form to apply the same methodologies developed for *RNA-seq*, and doing the same for the detection of DERs (section 5.2.2).

Integration

The previously studied omics, even if not so many, already give the possibility to construct an integrated view of multiple cellular aspects.

For example, if we are able to collect samples of *ChIP-seq* for an HM involved into the chromatin accessibility, and *ATAC-seq* samples with *RNA-seq* samples. We already are able to understand which regions of the genome are influenced by that specific HM, obtaining a crossed inspection with resulting region of *ATAC-*

seq. Moreover, we can look at how the gene expression has been influenced by these chromatin accession mechanisms.

To account even for simple integration problems like this one, we implemented functionalities for the **peak-gene annotation** of DERs/DBSs with DEGs using *ChIPpeakAnno*.

The interface allows for the selection of a `peaks` file (DER/DBS), automatically loaded from `IntegrHO/project_name/results/tables/peaks`. Additionally, an `annotation file` is required to retrieve the genomic features to annotate the peaks. In alternative, a list of DEGs can be selected. Other relevant parameters, such as the gene `feature` or the `max distance` between the peaks and the genes, are needed for the right working of `annotatePeakInBatch` function, defined inside the `ChIPpeakAnno` R/Bioconductor package.

After that the server side computed the results, they are showed with two plots organized in two panels of a `tabsetPanel` widget. A *pie chart* showing the percentage of detected features organized on their distance, and table of annotated regions with corresponding genes.

Finally, a tsv file with annotated peaks is saved inside the `IntegrHO/project_name/results/tables/annotated_peaks` folder, while the volcano plot, in HTML format, is saved into the `IntegrHO/project_name/results/plot` folder.

Another kind of integration is given by **functional annotation** analysis, where list of genes (annotated or resulting from differential experession analysis) can be used to detect cellular functional behaviours, thanks to databases born to annotate multiple gene contributing to specific cellular mechanisms (called pathways or gene ontology terms).

In order to interrogate these databases, several methodologies for functional analysis born [105–109]. The underlying idea of them is to use a list of genes to see which of them mostly *up-regulate*/*down-regulate* annotated cellular functional mechanisms

To account for this integration aspect we developed one interface for allowing the user to choose between two methods *gProfiler* [107] and *clusterProfiler* [108]. Both methodologies allow to investigate multiple databases, for pathways *KEGG* and *REACTOME* and for Gene Ontology we give the possibility to choose between Biological Process (BP), Molecular Function (MF) and Cellular

Component (CC).

The interface automatically loads the file inside the `IntegrHO/project_name/results/tables/annotated_peaks` and `IntegrHO/project_name/results/tables/DE_genes` folders, giving the possibility to select the number corresponding to the column within the gene list.

In order to choose the method to apply a `selectInput` shiny widget gives the possibility to choose between the *clusterProfiler* and the *gProfiler* methods. Moreover, the user has to select the type of functional annotation to do (pathway or Gene Ontology) and the database to use.

At the end of computations *IntegrHO* will present the table of the results in the main interface and will store them into the `IntegrHO/project_name/results/tables/functional` folder.

5.3 Reproducible Computational Research

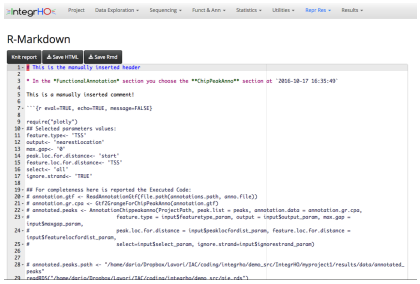
The most difficult part when using a GUI is to keep track the executed functionalities during the analysis. To face this need we equipped *IntegrHO* of a RR hidden layer (with aid of *easyReporting* package) able to trace all executed code by the user.

In combination with a system of CDF, *IntegrHO* stores code chunks and the input/output data of each analysis step inside an RMD file.

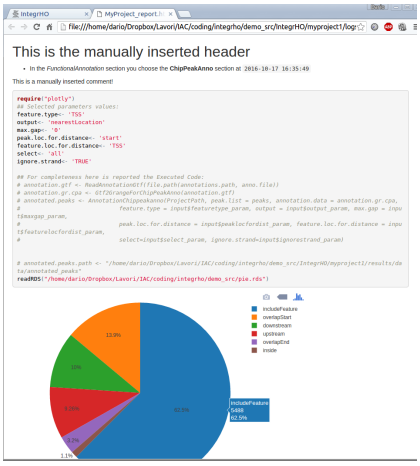
The auto-produced RMD file as it is, it's not useful to show to third-parties, it needs to be enriched with natural language comments, that's why we built a specific interface enabling the user to edit the automatically produced RMD file and to compile it on the fly (see sub-figure 5.3.1a).

The enriched output report can be produced in 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.



(a) The Interface for the editing of an auto-generated report with *IntegrHO* web platform.



(b) The report compiled and enriched with manually inserted comments.

Figure 5.3.1: The reproducible research with *IntegrHO*.

5.4 Discussions

We presented *IntegrHO*, a web-platform for multi-omics data analysis, integration and their visualization, tracing 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.

IntegrHO is a module-based software allowing easy expandability with other methodologies by implementing additional shiny modules. Indeed, a platform as *IntegrHO* can be expanded with several functionalities, but in particulare we are focusing on the implementation of methodologies for high-dimensionality samples integration methods, such as *mixOmics* [37] and *MoFa* [38].

Moreover, to allow the user to account for better quality analysis of *ChIP-seq* data, we are working on novel interfaces for the normalization and on methodologies for accounting for the right normalization to use.

Additionally, *IntegrHO* at the moment traces the code and stores the input/output data into caching database files, in order to construct an RMD file live editable from a dedicated interface. To improve *IntegrHO* RR layer we would like to implement additional methodologies, such as the graph construction of executed modules, in such a way the user will have also a graphical representation of the final enriched document, that can be presented as a graphical

abstract of the analysis.

In general, at the moment *IntegrHO* allows to deeply investigate single-omics data and, at the same time, it allows for an integration at functional analysis and a peak-gene annotation levels, offering various modules for data viasualization. But, in order to promote a wider overview of integration methodologies,it still needs some work to do.

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 time-course *RNA-seq* data. We presented *ticorser*, an R command line package for time-course *RNA-seq* data. With this instrument we propose a complete pipeline for the time-course 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, 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 proposed a possible workflow for the detection of DERs and their integration with *RNA-seq* data.

Due to the difficulty of keeping track even of a single omic analysis, we presented *easyReporting*, a possible approach to the Reproducible 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 analyze 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. Park, P. J. *ChIP-seq: Advantages and challenges of a maturing technology* 2009. doi:10.1038/nrg2641. arXiv: NIHMS150003.
2. Frommer, M. *et al.* A genomic sequencing protocol that yields a positive display of 5-methylcytosine residues in individual DNA strands. *Proceedings of the National Academy of Sciences* **89,** 1827–1831. ISSN: 0027-8424 (1992).
3. 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).
4. Wang, Z., Gerstein, M. & Snyder, M. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics* **10,** 57–63. ISSN: 1471-0056 (2009).
5. Van Berkum, N. L. *et al.* Hi-C: A Method to Study the Three-dimensional Architecture of Genomes. *Journal of Visualized Experiments.* ISSN: 1940-087X. doi:10.3791/1869. http://www.jove.com/index/Details.stp?ID=1869 (2010).
6. Andrews, S. FastQC. *Babraham Bioinformatics,* http://www.bioinformatics.babraham.ac.uk/projects/. ISSN: 00029548 (ISSN) (2010).

7. Langmead, B., Trapnell, C., Pop, M. & Salzberg, S. L. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology.* ISSN: 14747596. doi:10.1186/gb-2009-10-3-r25 (2009).

8. Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nature methods.* ISSN: 15487091. doi:10.1038/nmeth.1923. arXiv: {\#}14603 (2012).

9. Li, H. & Durbin, R. Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics.* ISSN: 13674803. doi:10.1093/bioinformatics/btp698. arXiv: 1303.3997 (2010).

10. Li, H. & Durbin, R. Fast and accurate short read alignment with BurrowsââĆňâĂIJWheeler transform. *Bioinformatics.* ISSN: 1367-4803, 1367-4811. doi:10.1093/bioinformatics/btp324 (2009).

11. Krueger, F. & Andrews, S. R. Bismark: A flexible aligner and methylation caller for Bisulfite-Seq applications. *Bioinformatics.* ISSN: 13674803. doi:10.1093/bioinformatics/btr167 (2011).

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

13. Kim, D., Langmead, B. & Salzberg, S. L. HISAT: A fast spliced aligner with low memory requirements. *Nature Methods* **12,** 357–360. ISSN: 15487105 (2015).

14. Kim, D. *et al.* TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome biology* **14,** R36. ISSN: 1474760X (2013).

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

16. Li, H. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics* **27,** 2987–2993. ISSN: 13674803 (2011).

17. Morgan, M., Pagès, H., Obenchain, V. & Hayden, N. *Rsamtools*

18. Thermes, C. *Ten years of next-generation sequencing technology* 2014. doi:10.1016/j.tig.2014.07.001. arXiv: arXiv:1312.0570v2.

19. Costa, V., Angelini, C., De Feis, I. & Ciccodicola, A. *Uncovering the complexity of transcriptomes with RNA-Seq* 2010. doi:10.1155/2010/853916.

20. Ozsolak, F. & Milos, P. M. *RNA sequencing: Advances, challenges and opportunities* 2011. doi:10.1038/nrg2934. arXiv: NIHMS150003.

21. 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.

22. Trapnell, C. *et al.* Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology* **28,** 511–515. ISSN: 10870156 (2010).

23. Roberts, A., Pimentel, H., Trapnell, C. & Pachter, L. Identification of novel transcripts in annotated genomes using RNA-seq. *Bioinformatics* **27,** 2325–2329. ISSN: 13674803 (2011).

24. Roberts, A., Trapnell, C., Donaghey, J., Rinn, J. L. & Pachter, L. Improving RNA-Seq expression estimates by correcting for fragment bias. *Genome Biology* **12.** ISSN: 14747596. doi:10.1186/gb-2011-12-3-r22. arXiv: 1111.6189v1 (2011).

25. Trapnell, C. *et al.* Differential analysis of gene regulation at transcript resolution with RNA-seq. *Nature biotechnology* **31,** 46–53. ISSN: 10870156 (2013).

26. Pepke, S., Wold, B. & Mortazavi, A. Computation for chip-seq and rna-seq studies. *Nature Methods* **6,** S22. ISSN: 15487105 (2009).

27. 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.

28. Russo, F., Righelli, D. & 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.

29. Russo, F., Righelli, D. & Angelini, C. Advancements in RNASeqGUI towards a Reproducible Analysis of RNA-Seq Experiments. *BioMed Research International* **2016.** ISSN: 23146141. doi:10.1155/2016/7972351 (2016).

30. 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).

31. 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).

32. Wei, Z., Zhang, W., Fang, H., Li, Y. & Wang, X. esATAC: an easy-to-use systematic pipeline for ATAC-seq data analysis. *Bioinformatics.* ISSN: 1367-4803. doi:10.1093/bioinformatics/bty141 (2018).

33. Righelli, D. *et al.* Differential Enriched Scan 2 (DEScan2): a fast pipeline for broad peak analysis. *PeerJ.* ISSN: 2167-9843. doi:10.7287/peerj.preprints.27357v1. https://peerj.com/preprints/27357/ (2018).

34. 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).

35. Ou, J. *et al.* ATACseqQC: A Bioconductor package for post-alignment quality assessment of ATAC-seq data. *BMC Genomics* **19.** ISSN: 14712164. doi:10.1186/s12864-018-4559-3. arXiv: arXiv:1311.3268v1 (2018).

36. Angelini, C. & Costa, V. Understanding gene regulatory mechanisms by integrating ChIP-seq and RNA-seq data: statistical solutions to biological problems. *Frontiers in Cell and Developmental Biology* **2,** 1–8. ISSN: 2296-634X (2014).

37. Rohart, F., Gautier, B., Singh, A. & Lê Cao, K. A. mixOmics: An R package for âĂŸomics feature selection and multiple data integration. *PLoS Computational Biology* **13.** ISSN: 15537358. doi:10.1371/journal.pcbi.1005752. arXiv: NIHMS150003 (2017).

38. Argelaguet, R. *et al.* MultiâĂŘOmics Factor AnalysisâĂŤa framework for unsupervised integration of multiâĂŘomics data sets. *Molecular Systems Biology* **14,** e8124. ISSN: 1744-4292 (2018).

39. Jia, B., Xu, S., Xiao, G., Lamba, V. & Liang, F. Learning gene regulatory networks from next generation sequencing data. *Biometrics* **73,** 1221–1230. ISSN: 15410420 (2017).

40. Meng, C. *et al.* Dimension reduction techniques for the integrative analysis of multi-omics data. *Briefings in Bioinformatics* **17,** 628–641. ISSN: 14774054 (2016).

41. 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).

42. Robinson, J. T. *et al.* Integrative genomics viewer. *Nature Biotechnology* **29,** 24–26. ISSN: 10870156 (2011).

43. 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).

44. Wang, B. *et al.* Similarity network fusion for aggregating data types on a genomic scale. *Nature Methods* **11,** 333–337. ISSN: 15487105 (2014).

45. Baggerly, K. A. & Coombes, K. R. Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology. *Annals of Applied Statistics* **3,** 1309–1334. ISSN: 19326157 (2009).

46. Potti, A. & Nevins, J. Misconduct in Science: An array of errors. *The Economist,* 9–11. ISSN: 00130613 (2011).

47. Fomel, S. & Claerbout, J. F. Reproducible Research. *Computing in Science & Engineering* **11,** 5–7. ISSN: 1521-9615 (2009).

48. Costa, V. *et al.* Distinct antigen delivery systems induce dendritic cells' divergent transcriptional response: New insights from a comparative and reproducible computational analysis. *International Journal of Molecular Sciences* **18.** ISSN: 14220067. doi:10.3390/ijms18030494 (2017).

49. Peng, R. D. & Eckel, S. P. Distributed reproducible research using cached computations. *Computing in Science and Engineering* **11,** 28–34. ISSN: 15219615 (2009).

50. Aranguren, M. E. & Wilkinson, M. D. Enhanced reproducibility of SADI web service workflows with Galaxy and Docker. *GigaScience* **4.** ISSN: 2047217X. doi:10.1186/s13742-015-0092-3 (2015).

51. 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).

52. 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).

53. Shepherd, L. & Morgan, M. *BiocFileCache: Manage Files Across Sessions* 2018.

54. Iqbal, S. A., Wallach, J. D., Khoury, M. J., Schully, S. D. & Ioannidis, J. P. Reproducible Research Practices and Transparency across the Biomedical Literature. *PLoS Biology* **14.** ISSN: 15457885. doi:10.1371/journal.pbio.1002333. arXiv: arXiv:1011.1669v3 (2016).

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

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

57. 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).

58. 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.

59. 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).

60. 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).

61. 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.

62. 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).

63. 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).

64. Robinson, M. D. & Smyth, G. K. Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics* **23,** 2881–2887. ISSN: 13674803 (2007).

65. 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).

66. 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).

67. 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.

68. 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).

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

70. 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).

71. 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).

72. 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).

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

74. 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).

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

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

77. 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).

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

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

80. 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).

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

82. 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).

83. 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).

84. 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).

85. 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. doi:10.3233/978-1-61499-649-1-87. http://ebooks.iospress.nl/publication/42900.

86. Blankenberg, D. *et al. Galaxy: A web-based genome analysis tool for experimentalists* 2010. doi:10.1002/0471142727.mb1910s89. arXiv: NIHMS150003.

87. Giardine, B. *et al.* Galaxy: A platform for interactive large-scale genome analysis. *Genome Research* **15,** 1451–1455. ISSN: 10889051 (2005).

88. 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).

89. Knuth, D. E. Literate Programming. *The Computer Journal* **27,** 97–111. ISSN: 0010-4620 (1984).

90. Leisch, F. in *Compstat* (2002). ISBN: 3790815179 9783790815177. doi:10.1007/978-3-642-57489-4_89.

91. Leisch, F. Sweave, Part I: Mixing R and LaTeX. *R News.* ISSN: 1662-4025. doi:10.1159/000323281 (2002).

92. Xie, Y. knitr: A General-Purpose Tool for Dynamic Report Generation in R. *R package version.* ISSN: 1687-1472. doi:http://yihui.name/knitr/ (2012).

93. Napolitano, F., Mariani-Costantini, R. & Tagliaferri, R. Bioinformatic pipelines in Python with Leaf. *BMC Bioinformatics* **14.** ISSN: 14712105. doi:10.1186/1471-2105-14-201 (2013).

94. Napolitano, F. repo: An R package for data-centered management of bioinformatic pipelines. *BMC Bioinformatics* **18.** ISSN: 14712105. doi:10.1186/s12859-017-1510-6 (2017).

95. Poplawski, A. *et al. Systematically evaluating interfaces for RNA-seq analysis from a life scientist perspective* 2016. doi:10.1093/bib/bbv036.

96. Services, H. GEO : the Gene Expression Omnibus. *Gene Expression* **23,** 2–3 (2007).

97. The Cancer Genoma Atlas. *TCGA* 2013.

98. Pedersen, T. L. *shinyFiles* 2018.

99. 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).

100. 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).

101. Robinson, M. D. & Oshlack, A. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology.* ISSN: 14747596. doi:10.1186/gb-2010-11-3-r25. arXiv: PMC2864565 (2010).

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

103. Ross-Innes, C. S. *et al.* Differential oestrogen receptor binding is associated with clinical outcome in breast cancer. *Nature* **481,** 389–393. ISSN: 00280836 (2012).

104. Angelini, C., Heller, R., Volkinshtein, R. & Yekutieli, D. Is this the right normalization? A diagnostic tool for ChIP-seq normalization. *BMC Bioinformatics* **16.** ISSN: 14712105. doi:10.1186/s12859-015-0579-z (2015).

105. Subramanian, A. *et al.* Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences* **102,** 15545–15550. ISSN: 0027-8424 (2005).

106. 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).

107. 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).

108. Yu, G., Wang, L.-G., Han, Y. & He, Q.-Y. clusterProfiler: an R Package for Comparing Biological Themes Among Gene Clusters. *OMICS: A Journal of Integrative Biology* **16,** 284–287. ISSN: 1536-2310 (2012).

109. Huang, D. W., Sherman, B. T. & Lempicki, R. A. Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nature Protocols* **4,** 44–57. ISSN: 17542189 (2009).

List of Figures

List of Tables