UNIVERSIDADE DE LISBOA
Faculdade de Medicina

Designing Transcriptomic Web Apps

Nuno Daniel Saraiva Agostinho

Orientador: Prof. Doutor Nuno Luís Barbosa Morais

Documento provisório
Tese especialmente elaborada para obtenção do grau de Doutor em Ciências Biomédicas, Ramo da Biologia Computacional

2021

Contents

Resumo **iv**

Summary **v**

Acknowledgements **vi**

List of Figures **vii**

List of Tables **viii**

Preface **ix**

1 Introduction **1**

1.1 The origin of life 1

1.2 Nucleic acids and protein synthesis 2

1.2.1 Alternative splicing 5

1.3 Bioinformatics 7

1.3.1 Transcriptomics 10

1.3.2 Publicly-available big data 12

1.3.3 Software development 14

2 Objectives **18**

3 psichomics **19**

3.1 Background 21

3.2 Materials and methods 22

3.2.1 Data retrieval 23

3.2.2 Gene expression pre-processing 24

3.2.3 Alternative splicing annotation 24

3.2.4 Alternative splicing quantification 26

3.2.5 Data grouping 27

3.2.6 Dimensionality reduction 27

3.2.7 Survival analysis 28

3.2.8 Differential splicing and gene expression analyses 28
3.2.9 Correlation between gene expression and alternative splicing quantifications 29
3.2.10 Gene, transcript and protein annotation and literature support . 29
3.2.11 Performance benchmarking 30
3.2.12 Alternative splicing quantification benchmarking 30
3.2.13 Continuous integration 31
3.3 Results 32
3.3.1 Case study 32
3.3.2 Time benchmarking 32
3.3.3 Alternative splicing quantification benchmarking 33
3.3.4 Impact 34
3.4 Conclusion 34

4 cTRAP **37**
4.1 Background 38
4.2 Materials and methods 39
4.2.1 ENCODE knockdown data 40
4.2.2 Ranking of similar CMap perturbations 41
4.2.3 Prediction of targeting drugs 43
4.2.4 Drug descriptor set enrichment analysis 44
4.2.5 Time and memory benchmarking 45
4.2.6 Continuous integration 46
4.3 Results 46
4.3.1 Case study 46
4.3.2 Time and memory optimisation 46
4.3.3 Graphical interface 46
4.4 Conclusion 51

5 CompBio app server **52**
5.1 Background 53
5.1.1 Desktop apps 53
5.1.2 Web apps 54
5.2 Materials and methods 55
5.3 Results 56
5.3.1 Design decisions 56
5.3.2 Docker Compose 57
5.3.3 ShinyProxy 58
5.3.4 Nginx 59
5.3.5 Background tasks 60

5.3.6 Website analytics 60
5.3.7 Resource monitoring 61
5.3.8 Server maintenance 61
5.4 Conclusion 61

6 Discussion **63**
6.1 PanAShé 63
6.2 Conclusion 63

Resumo

Summary

Abstract goes here

Acknowledgements

My PhD has been an incredible journey. A journey where I met fantastic people, some of which I could never imagine living without. A journey where I met painful challenges, some of which made me the person I am today. But no matter the obstacles, I am glad to have chosen this path for it was the one that led me to meet great new friends.

First of all, a big thank you to Nuno Morais for having me all these years: for inviting me to his lab and iMM, opening the doors to this crazy adventure.

My great ex-lab mates that taught me a lot, Marie Bordone and Lina Gallego.

- Elvira Leites, Eunice Paisana, Helena Pinheiro - João Sabino, Ana Duarte, Madalena Almeida, Inês Faleiro

For all the nights out and partying throughout the PhD, thank you Debanjan and Vasco Conceição.

And certainly to my greatest chaperon over this whole journey, through the road's ups and downs, Mariana Ferreira.

I will carry all my friends in my heart, together with those that were already there for a long time: Alicia Calvo-Villamañán, Inês Mendes, João Pereira.

Last, but not least, one special thank you to my family who have always been there. To my grandfather, my biggest supporter, a special thank you.

List of Figures

1.1 Thomas Morgan's fruit fly experiments 2
1.2 Figure drafts for a manuscript on DNA structure 4
1.3 Central dogma of molecular biology 4
1.4 Spliceosome assembly and splicing reactions 6
1.5 RNA-seq 11

3.1 psichomics screenshot 19
3.2 psichomics workflow 23
3.3 psichomics file structure 24
3.4 Alternative splicing quantification 26
3.5 Performance benchmark for alternative splicing analysis 33

4.1 cTRAP screenshot 37
4.2 cTRAP workflow 39
4.3 cTRAP file structure 40
4.4 Loading data from CMap perturbations 42
4.5 cTRAP similarity analysis 43
4.6 Welcome screen modal 48
4.7 User session workflow 49
4.8 cTRAP process running in Celery 51

5.1 Screenshot of CompBio's homepage 52
5.2 App server architecture 56
5.3 App server's file structure 58
5.4 Screenshot of app loading 59

List of Tables

3.1 Major psichomics milestones 20
3.2 Supported file formats in psichomics by source 25
3.3 Available alternative splicing annotations 25

4.1 Major cTRAP milestones 38

5.1 CompBio web services 55

Preface

There once was a boy who decided to take on a life-changing quest, an adventure where he had to climb the tallest of the mountains and dive into the deepest of the oceans. Although the end goal was not always clear in his mind, his heart was set: he would carry on and stand against everything in his path.

As the boy marched on, he saw a big old pyramid in the far-off distance. Hours and hours went by, yet they seemed to pass in a blink of his eyes. Deep inside the pyramid, within a dark room dimly lit by the weakling flame of a nearby torch, there was a beautiful door featuring a green-jade scarab beetle with its open wings made of a rainbow of precious gems. Above the beetle's raised forelegs laid a gold-carved sun. Although its façade was beautiful, the door was covered in centuries-old dust and its handle has long since forgotten the warm touch of a hand. The boy took a deep breath and opened the door.

Upon entering, he stood inside a big room with a very high ceiling and marvelling hieroglyphs feasting the walls. He was sure it was the finest Egyptian code he has ever seen. As he continued down the room, he started hearing some noises, noises which kept getting closer and closer, like if he was being followed. He quickly turned around. Nothing but darkness. He was alone as far as his eyes could see, yet the noise kept getting closer and closer. Suddenly, the boy looked up and saw countless bugs crawling from cracks above between the hieroglyphs. From one moment to the next, the bugs started swarming him. At first, the boy picked up his sword, swinged it around and tried to deal with all of the bugs, but they started fighting back, eating his patience, his time, his mind. The boy finally decided to simply run away and deal only with the bugs standing between him and the exit. After a tiresome challenge, the boy was able to finally run away from the bug-ridden hell. The boy learned that – no matter the time squatting each pesky bug – as long as there is code, there will be bugs.

After many days walking, he entered a big forest where the bushes stood like walls, creating a series of concurrent corridors that lead to different paths. The boy had to continuously decide which corridor to follow, but each decision seemed to him like a bad turn, no matter how right he was. He entered a labyrinth of decisions, where time kept running and running, whether he chose the right paths, the wrong ones or simply stood still wondering which paths to choose. As time went by, he started dashing, and

running, and sprinting in the maze, going back and forth through its branches. After a while, he found an exit for that decision-ridden hell. In the end, he learned to deal with the decisions of his past self: to learn from the bad ones and to smile at the good ones. No matter how much he wanted to go back in time, time always carries on and so should he.

As he continued his journey, he saw a small house in the distance. Starved and tired, he entered the house hoping to have some days to pull himself together. Inside, there was a single, almost naked room with mirrors all around the walls. In the middle, a big mirror covered by a linen sheet. The boy got closer to the mirror and removed its cover. A quick glance in the mirror was enough to reveal the boy's reflection and inner thoughts: his fears, his anxieties, his insecurities. The boy gave one step back, afraid of his own image. After all he went through, the boy was fuelled by his negative thoughts. He lowly murmured that there would be no end to his quest, that he would never achieve his goal – for how does one find something when not even knowing what to look for? Amidst his mournful inner monologue, he heard peaceful voices comforting him. He looked again into the mirror and realised he was not alone, for he knew that many helpful souls that helped him in his path were always there to cheer with him. The burden of this quest was his alone, but that did not mean that he couldn't walk tall aside others. So he ignored his own pessimistic thoughts and continued through his path with the hope of listening to the voices of those he loved once more.

Four years after his first step into this quest, the journey is now coming to an end. His tale ends as many others have: writing his story for others to learn from his past mistakes and glories. And by telling his story, by sharing his experience, by helping other travellers going through his former hurdles, he hopes to contribute to a better world, even if only by a little. The next door he opens will lead to new adventures, but the boy has now learned that no matter the challenges he faces, he will always be welcome in the arms of the ones he loves.

Chapter 1

Introduction

1.1 The origin of life

To follow my work, we have to rewind back some years ago. Billions and billions of years ago. Once upon a time there was a violent, harsh and unwelcoming planet among countless others. Earth was lifeless. But as millions of years went by, it started being home to a complex recipe whose special sauce is still being studied to this day: the primordial soup [1]. These were the perfect conditions for a young, 500-million-year-old planet to brew life.

And what is life? Although this question is not easy to answer, living organisms as we know them are complex, carbon-based systems composed of nucleic acids, proteins, carbohydrates and lipids. Together with some smaller molecules, these molecules are known as biomolecules and are crucial for the survival of living organisms [2].

Amongst those biomolecules, my work focuses on two: proteins and nucleic acids. Proteins have many important functions in an organism, including catalysing chemical reactions (enzymes), signalling cellular processes (hormones) and playing a role in the immune system (antigens) [2]. Regarding nucleic acids, deoxyribonucleic acid (DNA) stores the genetic data, the blueprint required to generate the majority of the vital molecules in the cell, including ribonucleic acid (RNA) molecules for protein synthesis and regulation [2].

One possibility for the origin of life is based on the *RNA world*, an hypothesis that states that primitive life forms were based on self-replicating RNA that evolutionarily predates DNA and proteins [1, 2, 3]. After all, those RNA molecules could store genetic information like DNA and catalyse life-critical chemical reactions akin to enzymes, making RNA a prime candidate for life to take its first steps. As life evolved, these specific functions may have been overtaken by protein enzymes that were more effective as reaction catalysers, and DNA, a more stable and less error-prone nucleic acid to store genetic information [1, 2].

1.2 Nucleic acids and protein synthesis

The word *protein* was first used in a publication by Gerardus Mulder in 1838, following the suggestion by his colleague Jöns Berzelius. Protein comes from a Greek adjective that means of the *first rank or position*, reflecting the perceived importance of these molecules [4, 5]. In his publication almost two centuries apart from today, Mulder reports the chemical compounds of *les substances les plus essentielles du règne animal: la fibrine, l'albumine et la gélatine* [4].

Some decades later, Friedrich Miescher isolated in 1869 a mysterious, protein-like substance from the pus of fresh surgical bandages that he named *nuclein*, found to be present in the cell nucleus of diverse animals, plants and fungi. Miescher's work led him to believe that increased nuclein could be associated with the first stages of cell division in proliferating tissues [6]. Albrecht Kossel (a former professor of Miescher) and colleagues described 5 organic compounds from nuclein: adenine, cytosine, guanine, thymine, and uracil [7, 8, 9, 10]. Nuclein was eventually renamed *nucleic acid*, but its importance was not yet recognised at the time [6].

In the first decades of the 20th century, the scientific consensus was that proteins carried genetic information, but Boveri and Sutton theorised otherwise [6, 11]. According to Sutton:

> the association of paternal and maternal chromosomes in pairs and their subsequent separation during the reducing division (...) may constitute the physical basis of the Mendelian law of heredity. ([11])

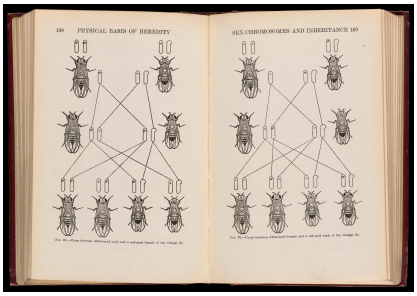

Figure 1.1: Thomas Morgan's fruit fly experiments.

The Boveri-Sutton chromosome theory of genetic inheritance followed Mendel's controversial [] work from 1865 [11] and was later supported by fruit fly experiments from an initially skeptic [] Thomas Morgan [12]. In 1915, Thomas Morgan, Hermann Muller and colleagues published a textbook with their findings describing genetic dominance, sex inheritance and chromosomal crossover. One chapter was provokingly titled *The Chromosomes as Bearers of Hereditary Material* [12]. In 1927, Hermann Muller discussed that exposure of fruit flies to X-ray radiation induced hundreds of genetic mutations, greatly contributing to the study of genetic mutations and evolution [13] [1].

[1] Muler's 1927 Science paper [13] that earned him the Nobel prize was not peer-reviewed, cited no references and lacked the methods section [14]. This may have happened not only because Muler wanted to be the first to share his hypothesis, but also because he agreed with the criticism by his long-time friend Edgar Alternburg that believed his data was not strong enough to confirm the induction of mutations [14]. It would take until 1930 for Muller to publish results addressing Altenburg's criticism, although Muller was aware of the issues before 1927 [14].

At the time, nucleic acids were classified as *thymus nucleic acid* found in animals (specially enriched in the thymus, hence its name) and *yeast nucleic acid* found in plants[2]. However, in 1933, Jean Brachet found evidence of *thymus nucleic acid* in the cell nucleus and of *yeast nucleic acid* in the cytoplasm of eukaryotic cells. His work suggested that both types of nucleic acids were present in the same cell with potentially different roles. During the 1930s, Phoebus Levene identified the phosphate backbone of nucleic acids, including its pentose sugars (deoxyribose and ribose) [18], which inspired their contemporary nomenclature: *thymus nucleic acid* is now known as deoxyribonucleic acid (DNA) and *yeast nucleic acid* as ribonucleic acid (RNA).

Although the word *gene* was used since 1909 to abstractly refer to Mendelian factors of inheritance (i.e. the units of heredity) [], Demerec tried to define it in his 1933 publication, *What is a Gene?*, alongside a figure of the *tentative structure of thymus nucleic acids* (DNA):

> (...) [A gene] is a minute organic particle, capable of reproduction, located in a chromosome and responsible for the transmission of a hereditary characteristic. ([19])

Later in 1941, Edward Tatum and George Beadle hypothesised that each gene is responsible for producing a specific enzyme and demonstrated that radiation-induced mutations could alter the resulting enzyme [20]. Together with Joshua Lederberg, Tatum demonstrated in 1946 that bacteria can exchange genetic material in a process called genetic recombination [**?**].

In 1952, Alfred Hershey and Martha Chase demonstrated that during viral infection by bacteriophage T2, its DNA, but not any viral protein, enter inside the bacteria [21]. The viral DNA is enough to produce the DNA molecules found in progeny virus particles. Amid the contemporary belief that protein were the carriers of hereditary information, the Hershey-Chase experiment complemented previous experiments suggesting that role belonged to DNA [21].

The work by Rosalind Franklin and Maurice Wilkins on analysing DNA using X-ray crystallography was crucial to the discovery of DNA's double helix structure, published in 1953 by Francis Crick and James Watson [22]. DNA is composed by two phosphate-sugar chains linked together by pairs of nucleotides via hydrogen bonds: adenine pairs with thymine and cytosine with guanine. Crick and Watson also proposed that this strand complementary could be important for DNA replication [22].

[2] *Yeast nucleic acid* was so named since first extracted from yeasts, considered from the plant kingdom by most scientists at the time. Starting with Ernst Haeckel in 1878, alternative proposed systems clumped fungi together with unicellular organisms instead (kingdoms of Protoctista, Protista, etc.) [15]. In 1959, Robert Whitakker suggested a fungi kingdom amid three others [16], a proposal that later blossomed into his popular five-kingdom classification system published in 1969 [17]. In his 1969 article, Whittaker explains why fungi should not be considered plants to his fellow peers.

Afterwards, Arthur Kornberg observed the proposed nucleotide pairing in synthesised DNA from an enzyme that replicates DNA using one of its strands as a template: the DNA polymerase [23].

During a time when not all scientists agreed that nucleic acids played a role in protein synthesis, George Palade described in 1955 the ribosome as *a small particulate component of the cytoplasm* that associates with RNA in the endoplasmic reticulum membrane to perform protein synthesis [24, 25]. The associated RNA was identified as of two types: ribosomal RNA (rRNA) that composed the ribosome itself and *soluble RNA* – transfer RNA (tRNA) –, found to carry the amino acids for protein synthesis [26, 25]. Multiple ribosomes were later found to bind to a single RNA molecule (polysomes), allowing for parallelised protein synthesis [27].

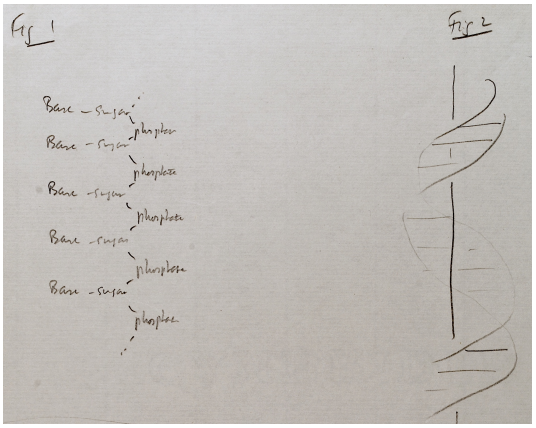

Figure 1.2: Figure drafts for a manuscript on DNA structure from Francis Crick and James Watson.

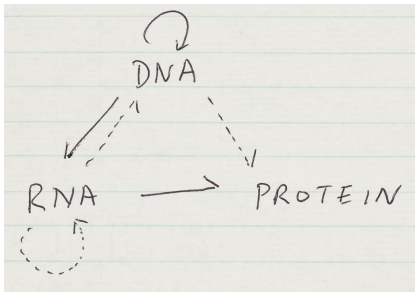

Figure 1.3: Central dogma of molecular biology.

Piece by piece, the role of DNA and RNA in protein synthesis was becoming clearer. Francis Crick proposed in 1958 that the genetic information flows from DNA to protein via RNA: the *central dogma of molecular biology*[28, 29]. It was also hypothesised at that time that triplets (*codons*) of the four nucleotides found in nucleic acids were necessary to produce each of the 20 universally-found types of amino acids that compose a protein [28, 30] and that those amino acids would be responsible for the protein's three-dimensional structure – and consequently, its functionality [28]. It took less than a decade to unravel which codons code for which aminoacid, an important breakthrough that allowed to predict protein sequences from DNA and RNA via the so-called universal genetic code [31, 32]. In 1959 and 1960, DNA-dependent RNA polymerase, an enzyme that synthesises RNA from DNA and common to all living organisms, was independently described by the labs of Samuel Weiss, Jerard Hurwitz and Audrey Stevens [33, 34, 35] [3].

François Jacob and Jacques Monod speculated in 1961 that ribosomal protein synthesis required an intermediate molecule with the template message to convert from DNA to protein and that would act as the *messenger* [25, 37]. Unlike many of their

[3] In 1955, Severo Ochoa and Marianne Grunberg-Manago discovered the polynucleotide phosphorylase (PNPase) enzyme that they thought to synthesise RNA polymers from DNA [36]. Ochoa was erroneously awarded a Nobel prize in 1959 for discovering the biological mechanism of RNA synthesis [].

contemporaries, they dismissed the known rRNA (and tRNA) molecules as the template for protein synthesis, given that they did not reflect the base composition of DNA, among other properties [25]. Based on contemporary experiments, Jacob and Monod proposed unstable RNA molecules as relevant candidates to their hypothesis and named them messenger RNA (mRNA) [25, 37]. Making the distinction between *structural genes* and *other, functionally specialized, genetic determinants*, Jacob and Monod also discussed the induced activation of repressors in mRNA synthesis [25].

In the 1969 publication entitled *Gene Regulation for Higher Cells: A Theory*, Roy Britten and Eric Davidson proposed that:

> Cell differentiation is based almost certainly on the regulation of gene activity, so that for each state of differentiation a certain set of genes is active in transcription and other genes are inactive. ([38])

Among the first publications of its kind, Britten and Davidson theorised about the intricate networks of gene regulation as fine-tuned systems in higher organisms based on the redundancy of different genomic elements and feedback loops. As they wrote, large genome sizes do not imply an increase in the number of genes compared to smaller genomes, but rather an increase in regulation complexity: *a large amount of DNA [including repeated DNA sequences] could be devoted to regulatory function*, for instance by sequence-specific binding of RNA of another gene [38].

In the beginning of the 1970s, the first studies on RNA processing were published. At the time, two types of RNA were distinguished inside the nucleus: ribosomal precursor RNA molecules that yield cytoplasmic rRNA and heterogeneous nuclear RNA (hnRNA) whose composition *resembles that of DNA* [39]. *Polyadenylic acid* (polyA) sequences ranging from 150 to 250 nucleotides were found to be added to the 3′ end of hnRNAs and cytoplasmic mRNAs, the first sign of eukaryotic RNA processing [39, 40]. James Darnell and colleagues thus proposed that hnRNAs and cytoplasmic mRNAs were related: the polyA sequence is added to hnRNAs post-transcriptionally in order to enable the export of nuclear RNAs to the cytoplasm, which are later found to be associated with ribosomes for protein synthesis [39, 40]. Later in 1974, the addition of a 5′-methylated cap was found in hnRNA and cytoplasmic mRNA and was proposed as a eukaryotic post-transcriptional RNA modification that protects the 5′-end of RNAs from degradation enzymes [41, 42].

1.2.1 Alternative splicing

First reported in mammalian cells infected with a human adenovirus 2 [43, 44] and later observed in endogenous mammalian and eukaryotic genes [45, 46], mRNA-DNA hybridisation experiments starting in 1977 suggested that genes are composed by intervening non-coding sequences, based on experiments from Richard Roberts, Philip Sharp

and colleagues. During transcription of the precursor mRNA (pre-mRNA), non-coding sequences (introns) are excised from the transcript, remaining only the expressed segments (exons) [43, 44, 47]. Moreover, multiple different transcripts may be produced from the same primary transcript by alternative splicing of segments of their sequence, thus promoting transcriptome diversity [43, 44, 48, 49, 50].

In 1985, a RNA-protein complex composed by U1, U2, U4, U5 and U6 small nuclear ribonucleoproteins (snRNPs) was reported central for RNA splicing: the spliceosome [51]. The spliceosome recognises splice sites (conserved sequences located at the 5′ and 3′ ends of an intron) and the branch point sequence and polypyrimidine tract (located within the intronic region) [52, 53]. The spliceosome then catalyses the excision of introns from pre-mRNA in two transesterification steps: (1) the 5′ end of the intron is cleaved and united to the conserved adenosine in the branch point sequence, forming an intermediary intron lariat, and then (2) the 3′ end of the intron is cleaved, releasing the intron lariat, and the two flanking exons are ligated [51, 54, 55]. The intron lariat is debranched (i.e. converted to a linear form) before its degradation [54, 56].

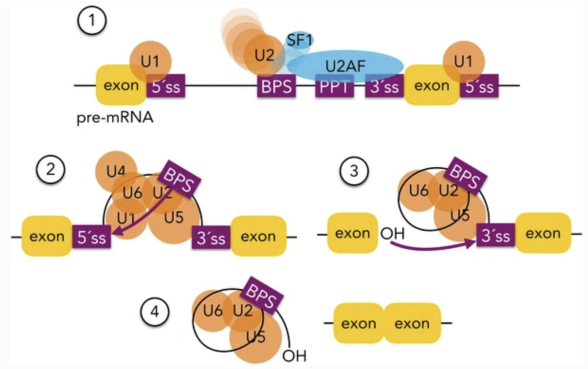

Figure 1.4: Spliceosome assembly and splicing reactions. (1) U1 snRNP binds to the 5′ splice site (5′ss), whereas the splicing factor 1 (SF1) and U2AF proteins bind to the branch point site (BPS), the polypyrimidine tract (PPT), and 3′ splice site (3′ss). The interaction between U1 and U2 snRNPs results in the formation of the pre-spliceosome. (2) The first splicing reaction is performed after the recruitment of the U4/5/6 snRNPs through a nucleophilic attack from the adenosine in the BPS to the 5′ss of the upstream exon. (3) The intron lariat is then formed. The free 3′ hydroxyl group performs a nucleophilic attack to the phosphate of the 3′ splice site of the downstream exon. (4) Finally, the intron lariat is released and both exons are ligated.

However, not all introns require the presence of the spliceosome to be spliced. During the 1980s, Thomas Cech identified rRNAs that underwent self-splicing by breaking and forming covalent bonds with no associated proteins [57], whereas Sidney Altman identified that the RNA subunit of a complex of proteins and RNAs (ribonulceoprotein complex) was essential for tRNA splicing and was able to cleave tRNAs *in the total absence of proteins* [58]. These RNAs with enzyme-like proteins were named ribozymes and they may have been a paramount mechanism that conferred evolutionary flexibility in life forms of yore [57, 1]. From fungi to plants and vertebrates, many spliced

genes across eukaryotic organisms share consensus sequences at their branch point, as well as their 5′ and 3′ splice sites, potentially making splicing one of the first molecular catalysts that appeared in living beings [3]. Primary transcripts from yeast can be spliced with the mammalian splicing machinery [3].

By 1989, it was known that alternative splicing is differentially regulated across cell types, development stages and tissues in eukaryotes, i.e. the same gene can lead to different, context-dependent spliced transcripts [59]. This regulation occurs via the interplay between trans-acting factors – RNA-binding proteins (RBPs) – and the cis-acting sequence elements to which they bind to, acting as splicing enhancers or inhibitors [59]. Multiple types of alternative splicing have also been described, including skipped exons, mutually exclusive exons, alternative 5′ and 3′ splice sites, alternative first and last exon and intron retention [59].

Among other molecular mechanisms, alternative splicing made clear that an organism complexity is not limited to the genome size or the number of protein-coding genes []. After all, the Australian lungfish (*Neoceratodus forsteri*) is the animal with the largest genome size (44 000 million base pairs), 14 times larger than that of humans (3 000 million base pairs) and 244 times larger than the fruit fly *Drosophila melanogaster* (180 million base pairs) [60, **?**]. And yet, their genomes harbour 31 000, 20 000 and 14 000 protein-coding genes, respectively, numbers in the same order of magnitude, whereas the remaining genome of these species are composed of intergenic regions and introns with high repeat content [60, **?**]. Notably, the fruit fly has 38 000 alternative transcripts generated from a single gene (Dscam) [50]. The multiple isoforms of this gene play a role the immune system of the fruit fly and may lead to more antigen diversity, thus increasing the evolutionary flexibility of the fruit fly.

Alternative splicing is deregulated in multiple disease contexts, including cancer, neurodegeneration and muscular dystrophies [61]. For instance, changes in splicing factors and subsequent perturbations to splicing can affect multiple hallmarks of cancer. Therefore, multiple potential splicing-targeting therapies have been developed based on antisense oligonucleotides, small molecules and novel techniques [61].

1.3 Bioinformatics

Following Sanger's work in 1959 on identifying the amino-acid composition of the protein insulin in multiple species, many proteins started being sequenced [62, 63, 64, 65]. Pointing to such studies, Crick predicted in 1958 that:

> Biologists should realize that before long we shall have a subject which might be called 'protein taxonomy' - the study of the amino acid sequences of the proteins of an organism and the comparison of them between species.

([28])

For that to come to fruition, more proteins from different species would first need to be fully sequenced for comparison. A technique known as Edman degradation was popularly used at the time to sequence proteins: the amino acids were identified by chemically fragmenting the protein, identifying the first 50-60 amino acids of each fragment and then reconstructing the full sequence based on the overlapping fragments, a long and tedious process performed by hand [65, 66]. All these limitations meant that only 6 different proteins were fully sequenced by 1962 [67]. To overcome the difficulties in this reconstruction step, Margaret Dayhoff and Robert S. Ledley published the first bioinformatic program: Comprotein [65, 67]. By 1965, the number of published protein sequences grew up to 65 and were published by Dayhoff in the first protein database, otherwise known as the book entitled *Atlas of Protein Sequence and Structure*[68].

In 1963, Linus Pauling and Emile Zuckerkandl discussed that cross-species comparative analysis of protein sequences could help determine the original protein sequence of their common ancestor and measure the evolutionary distance of the sequence of each species to that of their common ancestor [69]. However, these protein comparison methods were performed by hand, which meant that they were only practical for closely-related proteins such as homologs from different mammals [65]. Starting in 1970, computer-assisted phylogenetics started being a reality with the introduction of the Needleman-Wunsch algorithm [70] and variants, such as the Smith-Waterman algorithm [71]. These programs allowed to computationally measure the distance between two sequences by pair-wise comparison of amino acid of each protein sequence [70, 71].

Years after automatic protein sequencing machines being available based on Edman degradation – *protein sequenators* as first called in 1967 [72] –, DNA sequencing methods were presented based on electrophoresis: the enzymatic Sanger method (also known as dideoxy method) [73] and the chemical degradation method Maxam-Gilbert sequencing [74]. These tedious and slow processes required manual intervention *at both the experimental and interpretative levels* [75].

Leroy Hood and Lloyd Smith published a 1987 report on an instrument to automate the experimental procedure based on the Sanger method followed by computer analysis to determine the sequence of DNA fragments (i.e. base calling): the Applied Biosystems DNA sequencer, the first commercialised automated machine to sequence DNA [75]. Their article also discusses experimental issues with sequencing repetitive DNA regions, storing and sharing the big amount of data produced in the following years in *data banks*, as well as the algorithms required to quickly retrieve sequence from those databases – obstacles that impaired large-scale DNA sequencing, specially of the whole human genome [75].

Later, the advent of Next-Generation Sequencers (NGS) allowed the massive parallel sequencing of amplified DNA and RNA.

As the techniques to retrieve biological data were optimised, more and more data started being generated and published in public databases such as the Protein Data Bank [76] and GenBank [77]. Computers were no longer optional to survive the tsunami of biological data and new popular bioinformatic algorithms started to emerge. More advanced sequence aligners, such as CLUSTAL in 1988, efficiently allowed to align multiple sequences of amino acids or nucleotides from pairwise sequences [78]. In 1990, BLAST was presented as an efficient algorithm to quickly compare a DNA or protein sequence against the ever-increasing number of molecular sequences from biological databases [79].

From 1995 to 2000, the genomes of multiple organisms were published, including the bacterium *H. influenzae*, the yeast *S. cerevisiae*, the nematode *C. elegans*, the fruit fly *Drosophila*, and the plant Arabidopsis. In 2004, the Human Genome project was considered finished with its goal of publishing the human genome to the scientific community, leaving 8% to be determined due to technical limitations [80, 81]. Many advantages come from sequencing whole genomes, specially the *near complete* human genome:

> It allows systematic searches for the causes of disease – for example, to find all key heritable factors predisposing to diabetes or somatic mutations underlying breast cancer – with confidence that little can escape detection. It facilitates experimental tools to recognize cellular components – for example, detectors for mRNAs based on specific oligonucleotide probes or mass-spectrometric identification of proteins based on specific peptide sequences – with confidence that these features provide a unique signature. It allows sophisticated computational analyses – for example, to study genome structure and evolution – with confidence that subtle results will not be swamped or swayed by noisy data. At a practical level, it eliminates tedious confirmatory work by researchers, who can now rely on highly accurate information. At a conceptual level, the near-complete picture makes it reasonable for the first time to contemplate systems approaches to cellular circuitry, without fear that major components are missing. ([80])

More recently, the Telomere-to-Telomere (T2T) consortium exploited current high accuracy long-read sequencing (third-generation sequencing), along with other sequencing technologies, to publicly publish the gapless assembly of the human genome for use in biomedical research [81] [4].

[4] Although the most recent T2T pre-print manuscript from 2021 describes ongoing work for the missing chromosome Y [81], the latest assembly published in 24 January 2022 (CHM13 T2T v2.0) includes the full human genome sequencing: `ncbi.nlm.nih.gov/assembly/GCA_009914755.4`.

1.3.1 Transcriptomics

The term *omics* encompasses all fields in life sciences that analyse large-scale data to better understand the molecular world [82]. The first word using the *-omics* suffix dates back to a 1986 conference meeting among peers and beers. While discussing the name for a new journal intended to include sequencing data, gene mapping and new genetic technologies, Thomas Roderick proposed a name to illustrate a *new way of thinking about biology*: genomics [82, 83]. The genomics field is concerned with the study and cross-species comparison of genomes [83].

In the same vein, transcriptomics is the field that studies the transcriptome: the set of RNA transcripts[5], as first defined in 1996 [84]. One of the first studies to analyse the whole human transcriptome shared their efforts by publishing their quantitative hybridisation data with cDNA arrays from infant human brains [84]. Transcriptomics is usually performed based on data from high-throughput technologies that allow to simultaneously analyse the expression of multiple transcripts, including:

- **Serial Analysis of Gene Expression (SAGE)**
- **Expressed Sequence Tags (EST)**: Sanger sequencing method to identify random sequences
- **Microarrays**
- **RNA sequencing (RNA-seq)**: sequencing of bulk RNA using short reads.

The first step to analyse RNA is to obtain it from cells. RNA is isolated by disrupting cells and their RNA-degrading enzymes (RNases) and by concentrating RNA away from DNA and other biomolecules. As over 90% of the extracted RNA is ribosomal, depletion of rRNAs and/or enrichment of the desired species are required for proper analysis. It is usual to filter for polyadenylated RNAs (i.e. mostly mature mRNAs).

After extraction, transcripts are (indirectly) sequenced. RNA sequencing (RNA-seq) is a standard practice to better understand what features (genes, transcripts, exons, etc.) are expressed in the moment of RNA extraction, like taking a snapshot of a sample to later analyse it. Before the snapshot, the whole family of transcripts is prepared to look good in the photo: RNAs are fragmented in multiple sequences and converted to (complementary) DNA via reverse transcription enzymes. Finally, cDNA is sequenced in order to obtain reads, computer strings of nucleotides of the fragmented RNA sequences.

The number of sequencing reads per sample (known as sequencing depth or library size) depends on the intent of the experiment. The quantification of lowly expressed

[5]Depending on the context, the term *transcriptome* may exclusively refer to the study of mRNA transcripts instead of all RNA transcripts.

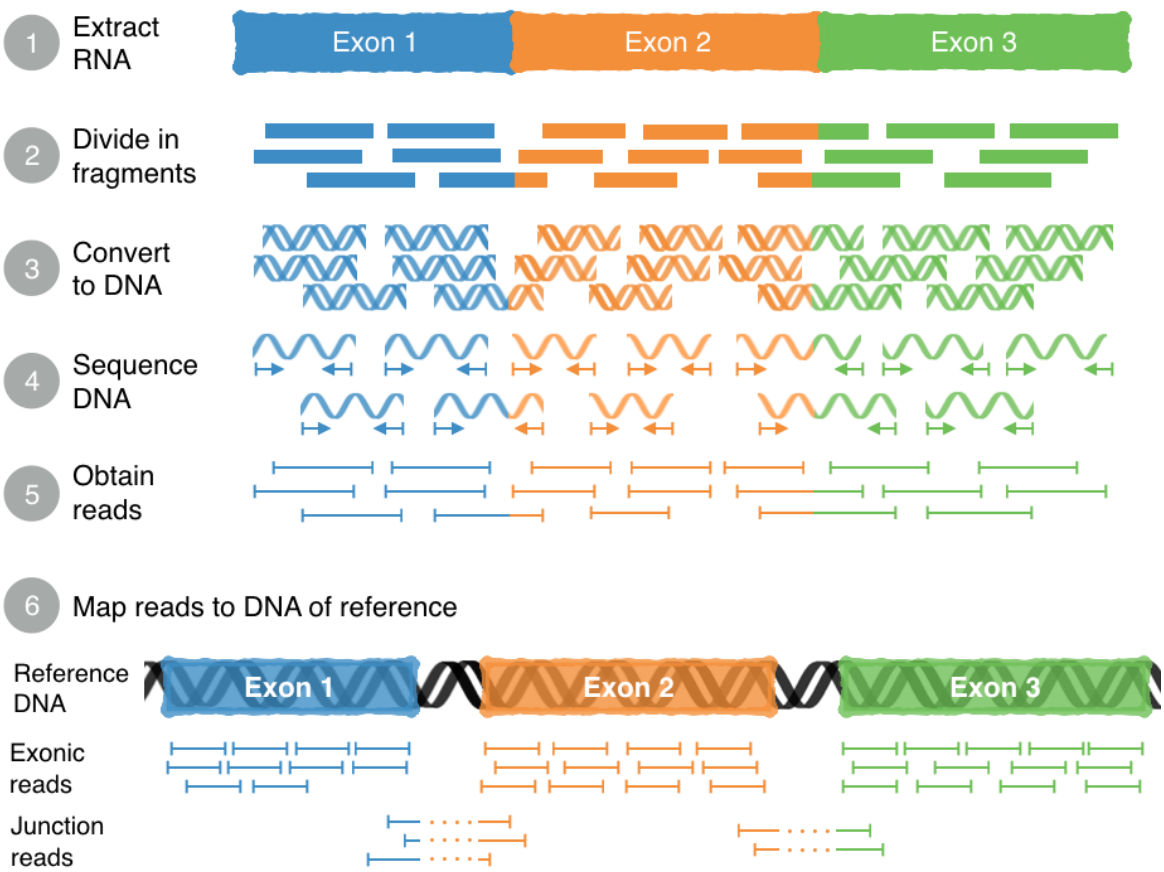

Figure 1.5: RNA-seq.

genes, for instance, requires a larger number of reads compared to highly-expressed genes, in order to detect more gene and with higher precision. To assess the sequencing depth required for a certain experiment, we can look at saturation curves.

Using 3 or more technical replicates allows to reduce external variability. Technical biases may occur, specially when performing multiple batches, as usually done for big sample sizes. To minimise such biases, it is important to use proper controls, randomise sample processing and manage sequencing reads. It is also possible to use batch-correction algorithms (like ComBat).

Quality control is a major step in RNA-seq data analysis. Low quality reads, duplicated sequences and overrepresented k-mers are some metrics used by FastQC to identify issues with reads from a particular sample that may be resolved by trimming reads or even discarding samples.

To reconstruct the moment at which the RNA-seq was sequenced, fragmented reads are compared against a reference genome or transcriptome[6], allowing to understand where the sequences come from and which features (genes, transcripts, exons, etc.)

[6]Alternatively, the transcriptome may be reconstructed *de novo* from available transcripts, as usually employed for organisms without any available reference. Given that short reads may be insufficient for this operation, newer technologies based on longer reads are preferentially used.

are more expressed. Unfortunately, some of those pieces fit in multiple places, such as reads from repetitive regions, and can be distributed based on empiric evidence and/or randomness. Shorter reads are more prone to multi-mapping (less chances of sequencing unique regions) and may be difficult to known for sure where to align them. Such issues can be mitigated using some techniques like paired-end reads and higher read coverage.

Based on the number of reads that were aligned on each region of the genome or transcriptome, it is possible to quantify features of interest, e.g. estimate gene, transcript or exon expression. To make these counts comparable across samples, they are normalised by transcript length, number of total reads and/or sequencing biases. Afterwards, gene expression can be linearly modelled across multiple conditions to identify differentially expressed genes [85].

Specifically, the study of alternative splicing has been greatly enhanced with the advent of cheaper, high-throughput technologies, since higher read coverage greatly benefits alternative splicing analysis []. One approach to study alternative splicing changes is based on the differential expression of isoforms of a single gene (e.g. CuffDiff2 [86]). However, isoform expression estimation based on shorts reads can be inaccurate. Other methods based on exon and junction reads specifically identify alterations in events of alternative splicing [85]: comparing the expression of different exons (differential exon usage, e.g. DEXseq [87]), estimating the percentage of alternative sequence inclusion (percent spliced-in, e.g. VAST-TOOLS [88, 89] and rMATS [90]) or based on graph theory to identify alternative splicing modules (e.g. DiffSplice [91]).

Transcriptomic studies allow to identify altered phenotype across development stages and pathological subtypes (such as stages of a disease progression), explore the molecular mechanisms underlying a phenotype and pinpoint disease biomarkers. These type of studies can be enhanced by integrating multiple data, such as genetic variants, methylation, drug sensitivity, proteomics and other omics data [].

1.3.2 Publicly-available big data

The development and economic feasibility of next-generation RNA sequencing lead to a wealth of publicly-available sequencing data that can be integrated with available clinical, drug-associated, mutation annotation, methylation and proteomic data. The public availability of these datasets to the research community ensures not only more transparency and reproducibility, but also bigger opportunities to unravel molecular mechanisms, identify more accurate biomarkers and predict novel treatments without individuals spending fortunes in grants for previously performed experiments, enabling new advances in personalised medicine via data sharing [92]. It also means that data can be exploited for other purposes other than those initially intended.

However, the ever-increasing amounts of large-scale data – *big data* – require more and more computational resources for their processing [65], specially when used to train machine learning models. These analyses can be quite prohibitive for researches interested in using current data for quick and standard biological queries. To satisfy such needs, many projects provide open access to pre-processed data directly via download (e.g. sequence aligned and normalised data) and/or via data exploring apps:

- **The Cancer Genome Atlas (TCGA)** with molecular and clinical data for more than 30 cancer types, including breast cancer and glioma from human samples. Multiple web apps tap into the data from this behemoth, including TCGASpliceSeq [93].

- **The Genotype Tissue Expression (GTEx)** project, a repository of gene expression data for more than 40 tissues from human samples [94]. It includes a portal to check data for specific genes and isoforms.

- The **recount** project has processed RNA-seq data from Sequencing Read Archive (SRA) [95, 96].

Even with the increasing economic feasibility of RNA-seq, cheaper technologies exist to measure gene expression for bigger sample sizes, like in the case of L1000, an inexpensive assay that estimates the expression of around 12000 genes [97]. This particular experiment was crucial to establish the Connectivity Map (CMap), a collection of almost half a million gene-expression signatures derived from chemical and genetic perturbations across multiple cell lines, dosages and timepoints [97]. CMap data has been used in multiple contexts, including to identify inhibitors of the stemness signature in multiple TCGA cancer types [98], to train machine learning models for designing molecules inducing desired transcriptomic changes [99] and to predict clinically-approved drugs for repurposing as SARS-CoV-2 antiviral agents [100].

To find effective novel treatments, it is important to understand how different cells respond to specific compounds. Multiple drug sensitivity datasets contain compound toxicity data across diverse cell lines, including the NCI-60 [101], the Cancer Therapeutics Response Portal (CTRP) [102] and the Genomics of Drug Sensitivity in Cancer (GDSC) [103]. Alongside gene expression data for each cell line, these datasets integrate data that allows to, for instance, pinpoint drugs that selectively target malignant cells.

Although it is getting easier to find more data online, there are many obstacles in making data accessible, including lack of standardised formats for storing molecular data, difficulty to retrieve human raw data, and lack of communication between different platforms [92]. Another major hurdle of disseminating data is related with

privacy issues and insufficient anonymisation of clinical data sharing, that can scare away individuals from sharing their personal data in public platforms.

Together with open data, there has also been a push for open access to scientific articles, an important step in disseminating science to everyone, including non-scientists. The publication of preprints has also been increasing, allowing for faster research dissemination and for early feedback from peers.

Open source is also important to easily allow reproducing published data analysis. Still, code alone may not suffice and the environment changes (e.g. different software versions and operative systems) may lead to unexpected results. To overcome those difficulties, standard tools can be used like Docker to share reproducible environments and Nextflow for scalable computational pipelines.

1.3.3 Software development

The nature of software has changed throughout the years from simple instructions that calculate Bernoulli numbers, the first published computer program by Ada Lovelace in 1843, to the "foundation for ultra-reliable software design", the on-board flight system that assisted the first moon landing with a crewed mission in 1969, supervised by Margaret Hamilton.

Software plays an important role in society nowadays and scientific research is no exception. For instance, the analysis of RNA-seq data requires specialised tools for quality control, sequence alignment, feature quantification, statistical analyses and visualisation techniques to assist researchers and clinicians in the biological interpretation of their results [85]. However, many of these specialised tools are developed by scientists with little programming knowledge and may lead to software with structural issues, e.g. non-user-friendly interfaces, unreproducible results, poorly documented systems, and reliance on deprecated technologies [104, 105].

To mitigate such issues, scientific software developers can adopt iterative approaches (like agile methodology) that fits the ever-evolving nature of scientific software development. Iterative development facilitates incremental improvement and delivery of stable software iterations by continuously planning and performing small tasks that are evaluated and prioritised based on the project context [104, 105, 106]. These development approaches go through a continuous cycle of several steps, including:

- **Requirement analysis** where we identify stakeholders and their requirements of what the system should do (functional) and that should characterise the system (non-functional; e.g. modular, reliable, secure, easy-to-use and scalable) [105, 107]. The initial requirements of a system tend to (and should) be of higher-level, but increase in complexity as the project advances: in simpler terms, a first version of the software should be simple and improved upon to get more features

over time [105, 108].

- **Design** concerns with the technologies to use during software development (e.g. frameworks for web app development and cloud hosting solutions for distribution), as well as the proper implementation of new features in the current software iteration or the integration with other programs via standard application programming interfaces (APIs), for instance. Good software design should allow to extend current functionality with as few changes to the core of the program as possible.

- **Development**, such as code structuring, feature implementation, bug fixing and code optimisation.

  - Version control systems (e.g. git) track changes to files in a project, allows to easily integrate code from different developers and compare files across diffterent versions [105, 109].
  - Kanban-like boards allow to visualise and manage the project workflow, where features and bugs can be commented on and tracked [105, 107].
  - Comprehensive documentation (tutorials, manuals, wikis, inline source code comments, etc.) is crucial to showcase the program and explain how features work via functions' description and examples to end-users and developers alike and can be written in plain text, Markdown and other common file formats [104, 105, 108, 109]. For development purposes, good documentation facilitates the maintenance and reusability of the program. Multiple tools automatically generate documentation from inline source code comments for different programming languages that can be automated, including roxygen2 for R [110] and Sphinx for Python [105, 107].

- **Testing** via unit tests, usability tests, performance tests, integration tests and security tests. Testing should be performed in multiple environments (e.g. different versions of programming languages, dependencies, operating systems and web browsers) [105, 108, 109]. The portion of the code covered by unit tests (written to check particular parts of code return the expected output when run) can be evaluated using code coverage tools such as CodeCov, allowing to understand how much of the code is being tested [107].

- **Software distribution (deployment)** is related with the release of program iterations to the hands of end-users: the easier it is to deploy, the faster new iterations can be released with new features and bug fixes [109].

  - Software can be distributed as web apps or desktop apps. Containerisation (e.g. via Docker) also helps distributing complex software and its dependen-

cies in an easier way. There are also many available repositories that allow to store code and/or compiled programs (GitHub to store git projects, DockerHub for Docker images, Bioconductor for bioinformatic R packages [111] and CRAN for generic R packages).

- When distributing code and/or programs, licensing must be defined since the first general release to avoid code misuse by third parties [105]. Different types of licenses can be chosen, allowing to decide whether the program can be modified, redistributed, used for commercial purposes and whether there are special conditions (e.g. any code derivation must be open-source and modifications must retain same license) [105].
- User feedback via bug reports and feature requests should be collected and considered for future iterations.

To facilitate continuous iterations, some steps can be automated using continuous integration (CI) tools, allowing to compile, run, test, deploy and document software. Those steps can then run in a multitude of environments whenever new changes are integrated in the code or at a regular interval (e.g. weekly) to ensure compatibility of the same software version with newer versions of external dependencies. Automation tools are crucial to quickly detect issues in a multitude of reproducible contexts and to promote code quality, testability, integration and continuous feedback [105, 107, 109, 104].

Many popular CI tools, like Travis-CI, GitHub Actions and AppVeyor, can be used for free in open-source projects. This approach promotes software quality by allowing peer-reviewing the code, reusing parts, fixing bugs, extending features and collaborating with external developers [105, 107], as well as it facilitates analysis transparency and reproducibility [105]. Moreover, it has educational value, allowing others to learn from the developed code, implemented solutions and project organisation [107].

Open-source biological R packages are fully supported by Bioconductor and its community [111], facilitating the distribution of R packages. Bioconductor also provides Docker images to facilitate access to their most popular R packages. Bioconductor also hosts packages that make use of the Shiny framework to create web apps from R [112].

The advent and popularisation of the Internet made software easier to distribute by simply allowing to download apps or use web browsers to directly present web apps []. Web apps follow a client-server architecture usually composed by three layers:

- **Presentation layer** is the user interface rendered by the user's local computer in their web browser based on standard web technologies: HTML, CSS and JavaScript.
- **Database layer** contains app-associated data such as user login details. The

database may be stored remotely (most commonly using a MySQL, PostgreSQL, MariaDB or MongoDB server) or in the user's computer.

- **Application layer**, the core logic of the application. The application layer can be run in its own remote server or in the same server as the database layer based on Python, Java, PHP or Perl. However, simpler web apps may opt for running the application layer as part of the presentation layer, thus using local resources.

Business intelligence dashboards are a common type of web app are that act as interactive data analysis tools to explore key performance indicators (KPI). Dashboards have increasingly been the subject of health care research, from monitoring medical equipment performance [113] and assessing surgical performance [114] to predicting high-risk patients for primary palliative care [115]. Inspired by such dashboards, tools to explore and analyse bioinformatic data are also available, including for gene expression analysis.

To provide effective dashboards, data visualisation is paramount for intuitive communication of complex results [116]. By introducing interactivity, the user can explore and manipulate the represented data, allowing greater flexibility to analyse and scrutinise the results relative to a comparable static plot and to obtain greater information insight by scrolling, zooming and drilling down into specific points [116].

Another vital aspect of any app is its interface, no matter whether graphical or command-line based. User-centered interfaces assist users achieving their goals by considering the program's audience, including their goals while using the program, the expectations on how to achieve them, familiarity with the vocabulary employed, computing skills and experience using similar software. Research on user interface design focuses beyond computing systems and covers human cognition, behaviour analysis and psychology [105, 107, 116, 117]. Interfaces can be evaluated and improved by asking for and listening to user's feedback or by performing usability testing [116].

Unfortunately, there is a lack of documentation on creating proper visual interfaces for bioinformatic apps. Even so, combining the concepts behind proper software development with dashboard design, big data visualisation and state-of-the-art transcriptomic analysis in freely available, open-source tools allows us to create potentially useful and (hopefully) popular web apps for sharing data insights with collaborators and even the whole scientific community.

Chapter 2

Objectives

The objectives of my work during the PhD thesis were to provide useful transcriptomic web apps to the scientific community.

First, as a continuation of the work I developed during my Master's thesis, I continued working on psichomics, an alternative splicing quantification, analysis and visualisation for TCGA data. We extended psichomics to support more data sources (including GTEx, recount2 and user-provided data), analyse gene expression, support alternative splicing quantification for 14 species, among other features.

- cTRAP: identification of candidate causal perturbations from differential gene expression data

Finally, we also developed an app server to deploy the previously mentioned tools as web apps, as well as the apps of other lab colleagues. This allows our apps to be readily available in a web browser and easily accessible to any user that visits our website.

Chapter 3

psichomics

After finishing the first year of my Masters in Informatics, I was looking for a challenging thesis where I could apply all that I learned into a bioinformatics project. While looking for computational biology groups, I found out about Nuno Morais lab, a research group interested in studying transcriptomics in disease.

Nuno made me aware of the need for graphical, interactive tools to allow non-experts to analyse and visualise splicing from processed big datasets. I loved the idea and started exploring ways of going from concept to reality. After toying with multiple frameworks and programming languages, I decided to stick with the R statistical language and the Shiny web app framework [112]. Shiny helped kick-start what would later be known as psichomics (Figure 3.1).

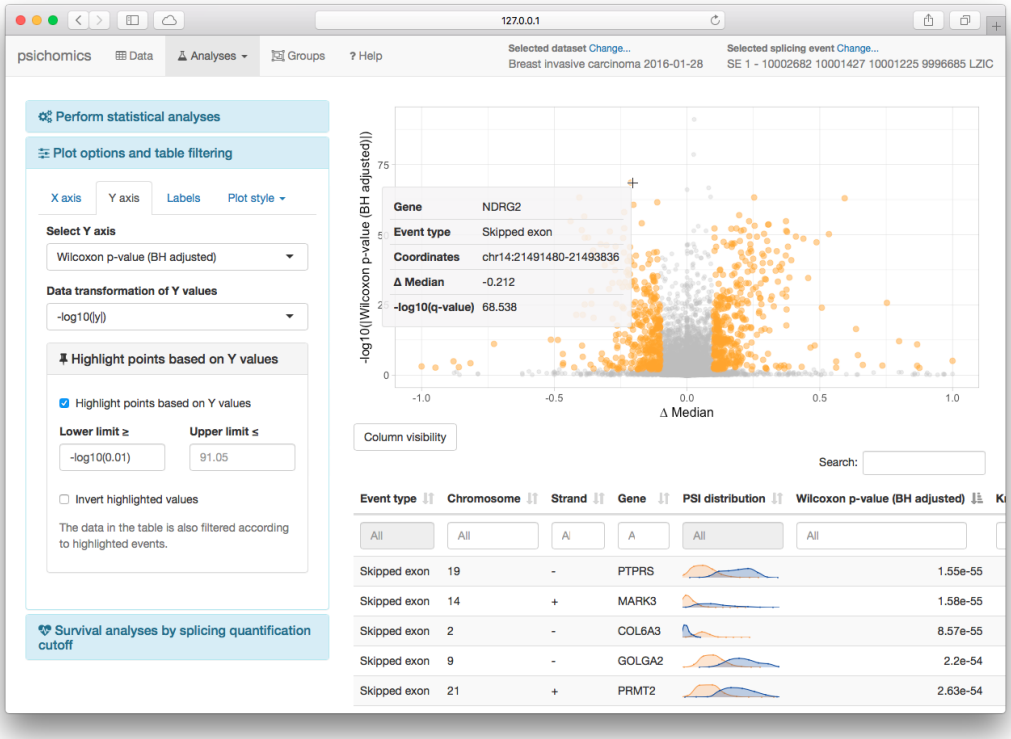

Figure 3.1: psichomics screenshot. TCGA breast cancer splicing analysis (28 Jan 2020).

psichomics was first made available in 2016 via Bioconductor to quantify, analyse and visualise alternative splicing using TCGA data [118]. Later on, I started my PhD project in the same lab and continued my work on psichomics. Nowadays, the tool allows to analyse gene expression and alternative splicing based on user-provided or public transcriptomic data, including those from TCGA [118], GTEx [94] and recount2 [95] (Table 3.1).

Table 3.1: Major milestones of psichomics.

Version	Release date	Main features
1.0.0	18 Oct 2016	Quantify and analyse alternative splicing from TCGA data[a]
1.0.8	18 Feb 2017	Analyse GTEx data
1.4.0	31 Oct 2017	Analyse gene expression from TCGA and GTEx data
1.4.3	13 Jan 2018	Faster alternative splicing quantification using Rcpp/C++
1.6.1	5 Jul 2018	Analyse recount2 and user-provided data
	2 Oct 2018	psichomics' original article [119] is published online
1.8.2	27 Mar 2019	Add list of RNA-binding proteins [120]
	21 Jan 2020	psichomics' book chapter [121] is published online
1.12.1	29 Jan 2020	Display visual diagrams of alternative splicing events
1.14.2	11 Aug 2020	Load VAST-TOOLS output[b] and more data formats
1.18.6	4 Oct 2021	Add web server support (optimised to run in ShinyProxy)[c]
1.20.0	28 Oct 2021	Download alternative splicing annotations for 14 species[d]

[a] Bioconductor release. [b] First time supporting intron retention events (psichomics does not calculate intron retention). [c] First version available online. [d] *Homo sapiens* (hg19 and hg38), *Mus musculus* (mm9 and mm10), *Bos taurus*, *Gallus gallus* (galGal3 and galGal4), *Xenopus tropicalis*, *Danio rerio*, *Branchiostoma lanceolatum*, *Strongylocentrotus purpuratus*, *Drosophila melanogaster*, *Strigamia maritima*, *Caenorhabditis elegans*, *Schmidtea mediterranea*, *Nematostella vectensis* and *Arabidopsis thaliana* based on VAST-TOOLS annotation.

Following many user requests, support for non-human data analysis was added with alternative splicing annotations for 14 species (including mouse, fruit fly, frog, and Arabidopsis thaliana). These annotations were published in Bioconductor ([]) and are based on those provided by the alternative splicing quantification tool VAST-TOOLS [88, 89]. Other improvements include support for loading VAST-TOOLS output tables, thus allowing to analyse intron retention events. However, I feel like it took until 2021 for the tool to fully realise its potential: when it finally became available online[1].

The article describing psichomics was published in 2018 [119]. We later were invited to write a methodological book chapter published in 2020 [121]. Both publications were written by me as the first and a co-corresponding author, together with Dr. Nuno Morais. The content of those publications, along with some content from my MSc Thesis, greatly inspired this chapter. In subsection 3.3.4: **Impact**, I also explore the reception of psichomics by the scientific community.

[1] More information in chapter 5: **CompBio app server**.

3.1 Background

The relevance of alternative splicing changes in physiological and disease conditions, along with the increasing economic feasibility of next-generation RNA sequencing (RNA-seq), has progressively driven transcriptome-wide alternative splicing studies [122, 123, 124, 125, 126] and promoted large consortium efforts to assemble publicly accessible splicing data. Such efforts include The Cancer Genome Atlas (TCGA), that catalogues clinical and molecular profiling data from multiple human tumours [118]; the Genotype-Tissue Expression (GTEx) project, that focuses on profiling normal human multi-tissue data [94]; and the recount2 project, a resource of processed RNA-seq data for over 2000 studies, mostly from the Sequence Read Archive (SRA) [95].

Among the openly available processed data from those public projects, counts of RNA-seq reads aligned to exon-exon junctions may be exploited for alternative splicing quantification and further analysis. Indeed, the ability to couple proper differential splicing analysis with, for instance, gene expression, protein domain annotation, clinical information or literature-based evidence enables researchers to extract, from those comprehensive public datasets, valuable insights into the role of alternative splicing in physiological and pathological contexts, as well as putative splicing-associated prognostic factors and therapeutic targets [123, 124, 125, 126, 127].

Several tools are currently available to quantify, analyse and visualise alternative splicing data. Similarly to psichomics, some analyse alternative splicing based on the commonly-employed and intuitive proportion of reads aligned to splice junctions supporting the inclusion isoform, known as Percent Spliced-In or PSI [122]. Examples of such tools are AltAnalyze [128], MISO [129], SpliceSeq [130], VAST-TOOLS [88], rMATS [90], SUPPA [131] and Whippet [132]. However, current alternative splicing analysis tools, regardless of their quantification metric, suffer from at least one of the following shortcomings:

1. Lack of support for imputing pre-processed data (e.g. splice junction read counts), leading to redundant, time-consuming RNA-seq read alignment and exon-exon junction detection, preceding alternative splicing quantification when exon-exon junction quantification is already available (e.g. when analysing TCGA, GTEx or recount2 data).

2. Limited set of statistical options for differential splicing analysis, mostly relying on median-based non-parametric tests and restricted to pairwise comparisons.

3. No incorporation of molecular or clinical information enabling analyses that reflect factorial designs or test linear models, for example. This is particularly limiting in the exploration of clinical datasets where, for instance, survival analyses permit assessing the potential prognostic value of alternative splicing events.

4. No support for transcriptome-wide filtering and sub-setting of events, based on common features or the outcome of statistical analyses, for interactive exploration of individual events of interest.

5. No user-friendly interactive graphical interface neither support for customisable statistical plots.

Using available pre-processed splice junction read counts from big data repositories exempts researchers from storing and processing large raw files that require expensive computational resources. To our knowledge, no tool allows to perform transcriptome-wide alternative splicing analysis using splice junction read counts from publicly available RNA-seq datasets (e.g. from TCGA, GTEx and recount2) with the option to easily compare them with user-provided groups interactively created based on sample metadata. For instance, jSplice [133] and DIEGO [134] do quantify alternative splicing from junction read counts but the user needs to manually convert such counts into a file format accepted by those programs. Moreover, none of those tools support survival analysis, exploratory and differential analyses of gene expression, or tests for association between gene expression levels and/or alternative splicing quantification changes.

To offer a comprehensive pipeline that integrates all the aforementioned features through both a command-line and an easy-to-use graphical interface, we have developed psichomics, an R package to quantify, analyse and visualise alternative splicing and gene expression data using TCGA, GTEx, recount2 and/or user-provided data. Our tool interactively performs dimensionality reduction, differential splicing and gene expression and survival analyses with direct incorporation of molecular and clinical features. We successfully employed psichomics to analyse stage I breast cancer TCGA data and identified alternative splicing events with putative prognostic value.

psichomics can be installed via Bioconductor (`bioconductor.org/packages/psichomics`) or Docker (`nunoagostinho/psichomics`). The latest version of psichomics can also be accessed online at `compbio.imm.medicina.ulisboa.pt/psichomics`. The source code of psichomics is available at `github.com/nuno-agostinho/psichomics`.

3.2 Materials and methods

psichomics allows to automatically process data (provided by the user or automatically downloaded from TCGA, GTEX and recount2), quantify alternative splicing, normalise and filter gene expression data and perform downstream analysis, including dimensionality reduction, differential expression/splicing analysis, correlation analysis, survival analysis and annotation of genes, transcripts and proteins (Figure 3.2).

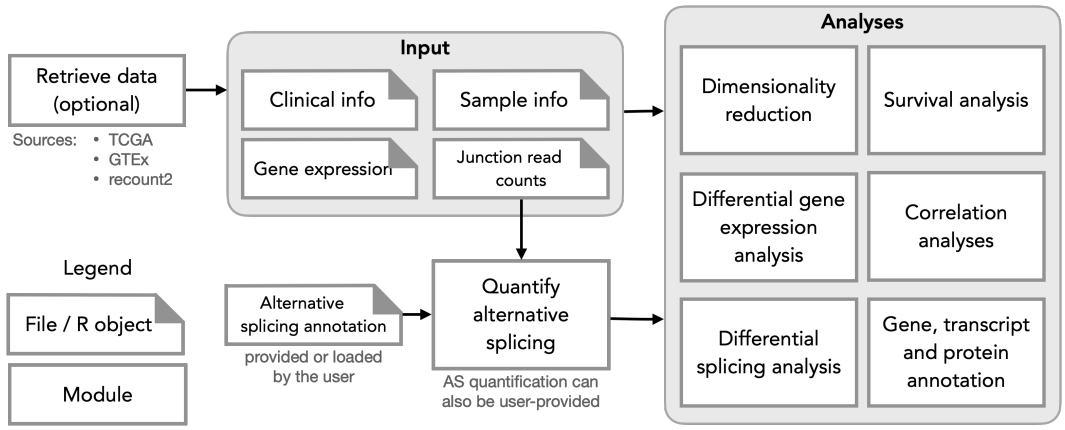

Figure 3.2: psichomics workflow. The user can provide their own input data or load data from TCGA, GTEx or recount2 to normalise gene expression data and quantify alternative splicing for downstream analyses.

psichomics was designed as a modular R package to easily modify and extend its components (Figure 3.3). These include support for multiple file formats and automatic data retrieval from multiple sources, parsing and standardisation of alternative splicing event identifiers from different programs and a variety of data analysis methodologies.

Using psichomics for alternative splicing analysis begins with the loading of splice junction read count data provided by the user or from external sources, followed by the quantification of alternative splicing (in case no pre-computed quantification is loaded) and subsequent analyses. Alternative splicing quantification is computed based on RNA-seq reads that align to exon-exon junctions and the genomic coordinates (annotation) of alternative splicing events. The proportion of reads aligned to junctions that support the inclusion isoform, known as the Percent Spliced-In or PSI [122], was the chosen quantification metric.

3.2.1 Data retrieval

Exon-exon junction and gene expression quantifications (obtained from pre-processed RNA-seq data), clinical data and sample metadata are accessible through FireBrowse's web application program interface (API) for TCGA data retrieval (`firebrowse.org/api-docs`). The FireBrowse API is used in psichomics to automatically download TCGA data according to the user-selected tumour type(s) as tab-delimited files within compressed folders, whose contents are subsequently loaded with minimal user interaction. GTEx data are automatically downloaded via the GTEx data portal (`gtexportal.org`) and select SRA project data via recount2 [95].

Other SRA projects and user-provided files may also be loaded in appropriate formats (Table 3.2), allowing for subsequent alternative splicing analysis, as explained in

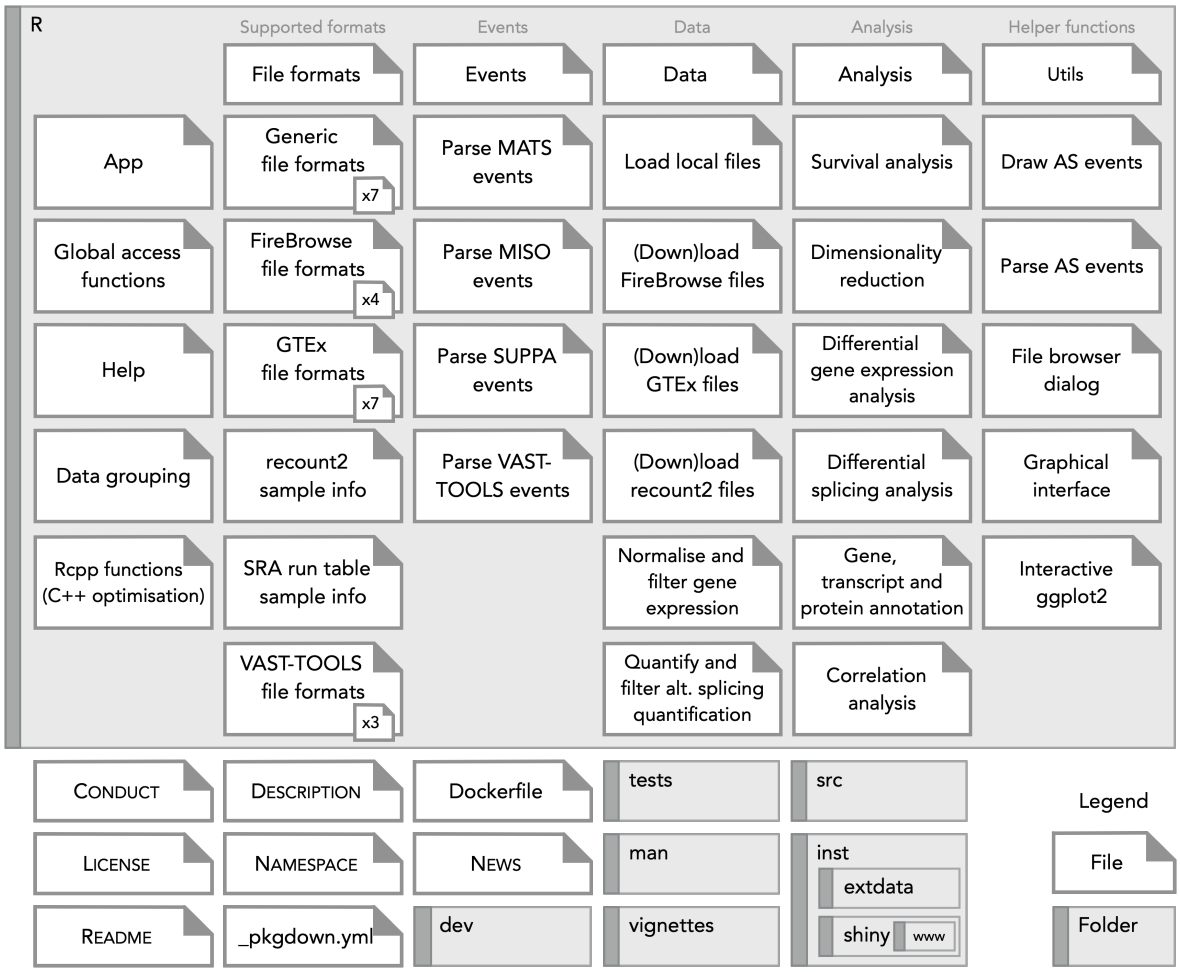

Figure 3.3: Visual representation of psichomics' file structure. As usual in an R package, the `R` folder contains the scripts with psichomics functions and data. `dev` is a custom folder that stores supporting scripts (e.g. test workflows); its contents are not included when building the R package.

the tutorial available at `nuno-agostinho.github.io/psichomics/articles/custom_data`).

3.2.2 Gene expression pre-processing

Gene expression quantifications can be filtered based on user-provided parameters (for instance, to account solely for genes supported by 10 or more reads in 10 or more samples, as performed by default) and normalised by raw library size scaling using function `calcNormFactors()` from R package `edgeR` [135]. Afterwards, counts per million reads (CPM) are computed and $\log_2$-transformed (if desired) using the function `cpm()` from `edgeR`. Log_2-transformation is performed by default.

3.2.3 Alternative splicing annotation

Annotations of alternative splicing events are available on-demand in psichomics for 14 species (Table 3.3). Human annotations were first available as a mix from multiple sources (as described in the following paragraphs). To support multiple species,

Table 3.2: Supported file formats in psichomics by source.

Source	Sample information	Subject information	Gene expression	Exon junction quantification	Alternative splicing quantification
SRA Run Selector	Yes				
STAR			Yes	Yes	
VAST-TOOLS			Yes		Yes
TCGA/FireBrowse	Yes	Yes	Yes	Yes	
SRA/recount2	Yes	Yes	Yes	Yes	
GTEx	Yes	Yes	Yes	Yes	
Other files	Yes	Yes	Yes	Yes	Limited[a]

[a] psichomics cannot fully parse alternative splicing events (e.g. it may not identify the cognate gene and coordinates) based on tables from these sources.

annotations were created based on VAST-TOOLS 23.06.20 (including for human, thus the redundancy). Custom annotation files can also be created by following the appropriate tutorial available at `nuno-agostinho.github.io/psichomics/articles/AS_events_preparation`.

Table 3.3: Available alternative splicing annotations.

Species	Assembly	Source
Homo sapiens	*hg19 and hg38*	VAST-TOOLS, SUPPA, MISO and rMATS
		VAST-TOOLS
Mus musculus	mm9 and mm10	VAST-TOOLS
Bos taurus	bosTau6	VAST-TOOLS
Gallus gallus	galGal3 and galGal4	VAST-TOOLS
Xenopus tropicalis	xenTro3	VAST-TOOLS
Danio rerio	danRer10	VAST-TOOLS
Branchiostoma lanceolatum	braLan2	VAST-TOOLS
Strongylocentrotus purpuratus	strPur4	VAST-TOOLS
Drosophila melanogaster	dm6	VAST-TOOLS
Strigamia maritima	strMar1	VAST-TOOLS
Caenorhabditis elegans	ce11	VAST-TOOLS
Schmidtea mediterranea	schMed31	VAST-TOOLS
Nematostella vectensis	nemVec1	VAST-TOOLS
Arabidopsis thaliana	araTha10	VAST-TOOLS

The hg19 annotation of human alternative splicing events was based on files used as input by MISO [129], VAST-TOOLS [88], rMATS [90] and SUPPA [131]. Annotation files from MISO and VAST-TOOLS are provided in their respective websites, whereas rMATS and SUPPA identify alternative splicing events and generate such annotation

files based on a given isoform-centered transcript annotation. As such, the human transcript annotation was retrieved from the UCSC Table Browser [136] in GTF and TXT formats, so that gene identifiers in the GTF file (misleadingly identical to transcript identifiers) were replaced with proper ones from the TXT version.

The collected hg19 annotation files were non-redundantly merged according to the genomic coordinates and orientation of each alternative splicing event and contain the following event types: skipped exon (SE), mutually exclusive exons (MXE), alternative first exon (AFE), alternative last exon (ALE), alternative 5′ splice site (A5SS), alternative 3′ splice site (A3SS), alternative 5′ UTR length (A5UTR), alternative 3′ UTR length (A3UTR), and intron retention (IR). The resulting hg19 annotation is available as an R annotation package in Bioconductor at `bioconductor.org/packages/alternativeSplicingEvents.hg19`, whereas the hg38 annotation (whose coordinates were converted from those of the hg19 annotation through function `liftOver()` from package `rtracklayer` [137], based on the hg19 to hg38 chain file from UCSC) is also available as an R annotation package in Bioconductor at `bioconductor.org/packages/alternativeSplicingEvents.hg38`.

3.2.4 Alternative splicing quantification

For each alternative splicing event in a given sample, its PSI value is estimated by the proportion of exon–exon junction read counts supporting the inclusion isoform therein [122]. The junction reads required for alternative splicing quantification depend on the type of event (Figure 3.4). Alternative splicing events involving a sum of junction read counts supporting inclusion and exclusion of the alternative sequence below a user-defined threshold (10 by default) are discarded to avoid imprecise quantifications based on insufficient evidence.

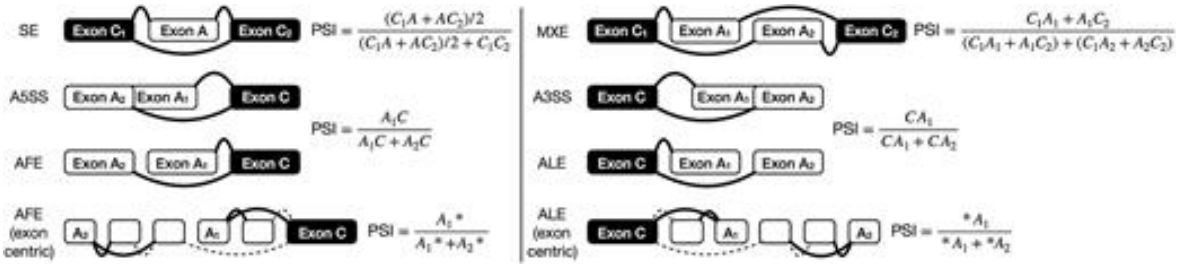

Figure 3.4: Alternative splicing quantification. Splice junctions required to quantify alternative splicing based on event type. C1A and AC2 represent read counts supporting junctions between a constitutive (C1 or C2, respectively) and an alternative (A) exon and therefore alternative exon A inclusion, while C1C2 represents read counts supporting the junction between the two constitutive exons and therefore alternative exon A exclusion. A1* and A2* represent the sum of read counts supporting junctions spanning the alternative first (A1) and second (A2) exon, respectively. Legend: skipped exon (SE), mutually exclusive exons (MXE), alternative 5′ splice site (A5SS), alternative 3′ splice site (A3SS), alternative first exon (AFE) and alternative last exon (ALE).

Alternative splicing quantification in psichomics is currently based on exon-exon junction read counts, yet intron retention events require intron-exon junction read counts for their quantification [138], whereas alternative 5′- and 3′-UTR require exon body read counts. psichomics does not currently quantify those types of alternative splicing events.

By default, psichomics quantifies all skipped exon events. However, the user can select to measure other types of alternative splicing events (Figure 3.4) and may hand in the list of genes whose alternative splicing events are to be specifically quantified. Furthermore, the step of alternative splicing quantification may be avoided if previously performed. psichomics allows the user to save the quantification of alternative splicing in a file to be loaded in a future session.

3.2.5 Data grouping

psichomics allows to group subjects and their samples or genes and their alternative splicing events for subsequent analysis. Subject and sample grouping can be performed based on available phenotypic (e.g. tissue type and histology) and clinical (e.g. disease stage, smoking history and ethnicity) features. Gene and splicing event grouping relies on respective user-provided identifiers. Moreover, the association between subject/sample groups specified by the user and those defined by the outcome of gene expression and alternative splicing analyses or by other clinical categorical variables can be statistically tested with Fisher's exact tests, implemented through function `fisher.test()` from `stats`.

3.2.6 Dimensionality reduction

Dimensionality reduction techniques can be performed on tables containing alternative splicing and gene expression quantifications, with the samples of interest as rows and the selected (if not all) splicing events or genes as columns, after centering and/or scaling the respective distributions (by default, they are only centered).

Principal component analysis (PCA) identifies the combinations of variables that contribute the most to data variance [139] and it is implemented through the singular value decomposition (SVD) algorithm provided by the `prcomp()` function from R package `stats`. The total contribution of each variable (splicing event or gene) towards data variance along selected principal components is measured based on the implementation of `fviz_contrib` from `factoextra`.

Independent component analysis (ICA), a method used for decomposing data into statistically independent components [140], can also be performed through the `fastICA()` function from the eponymous R package, preceded by data centering and/or scaling with the `scale()` function.

As many of the aforementioned functions cannot handle missing data, a user-defined threshold for the accepted number of missing values per alternative splicing event or gene (5%, by default) is used to discard variables before performing dimensionality reduction, whereas the remaining missing values are imputed for each variable as the median from non-missing data samples.

Moreover, samples can be clustered using k-means, partitioning around medoids (PAM) or clustering large applications (CLARA) methods, with the latter being optimised for large datasets and thus preferred by default. The implementation of these methods is based on the `kmeans()` function from `stats` and `pam()` and `clara()` functions from `cluster`, respectively.

3.2.7 Survival analysis

Kaplan-Meier estimators (and illustrating curves) [141] and proportional hazard (PH) models [142] may be applied to groups of patients defined by the user based on clinical features derived, for instance, from TCGA and user-owned data, with survival distributions being compared using the log-rank test. Survival analyses are implemented in psichomics using functions `Surv()`, `survfit()`, `survdiff()` and `coxph()` from R package `survival` [143].

To evaluate the prognostic value of a given alternative splicing event, survival analysis can be performed on groups of patients separated based on a given alternative splicing quantification (i.e. PSI) cut-off. Patients with multiple samples are assigned the average PSI value of their respective samples after sample filtering (e.g. when using TCGA data, only tumour samples are used for survival analysis by default). When survival differences are estimated for multiple PSI cut-offs for a single alternative splicing event, psichomics suggests the optimal cut-off that minimises the P-value of the log-rank test used to compare survival distributions, graphically supporting the suggestion with a PSI cut-off versus P-value scatter plot. Survival analysis can also be performed on groups defined by an expression cut-off for a selected gene.

3.2.8 Differential splicing and gene expression analyses

In psichomics, analysis of differential splicing between user-defined groups of samples can be performed on all or selected alternative splicing events. Given the non-normal distribution of PSI values [144, 145], median- and variance-based non-parametric tests, such as the Wilcoxon rank-sum (also known as Mann–Whitney U), Kruskal–Wallis rank-sum and Fligner–Killeen tests, are available and recommended [**?**]. Levene's and unpaired t-tests can nonetheless be performed as well. All these tests are available through the `stats` package with their default settings, except for Levene's test that was implemented based on the `leveneTest.default()` function from the `car` package.

To correct for multiple testing where applicable, P-value adjustment methods for the family-wise error rate (Bonferroni, Holm, Hochberg and Hommel corrections) and the false discovery rate (Benjamini–Hochberg and Benjamini–Yekutieli methods) are available through function `p.adjust()` from package `stats`. By default, multiple testing correction is performed using the Benjamini-Hochberg method.

Although the aforementioned statistical tests are also available to analyse the expression of single genes, genome-wide differential gene expression analysis is implemented based on gene-wise linear model fitting (using `lmFit` from R package `limma` [146]) for two selected groups, followed by moderated t-tests and the calculation of log-odds of differential expression, using empirical Bayes moderation of standard errors (function `eBayes()` from `limma`) and gene-wise variance modelling (`limma-trend`).

Statistical results can be subsequently explored through density and volcano plots with customisable axes to assist in the identification of the most significant changes when analyzing distributions across single or multiple events, respectively. A corresponding table with the results of all statistical analyses is also available and can be retrieved as a tab-delimited plain text file.

3.2.9 Correlation between gene expression and alternative splicing quantifications

The Pearson product-moment correlation coefficient, Spearman's rho (default) and Kendall's tau, all available with `cor.test()` from `stats`, can be used to correlate gene expression levels with alternative splicing quantifications. Such analyses allow, for instance, to test the association between the expression levels of RNA-binding proteins (RBPs) and PSI levels of interesting splicing events to identify which of these may undergo RBP-mediated regulation. As such, a list of RBPs is provided in-app [120], but the user can also define their own group of genes of interest for the test.

3.2.10 Gene, transcript and protein annotation and literature support

The representational state transfer (REST) web services provided by Ensembl [147], UniProt [148], the Proteins API [149] and PubMed [150] are used in order to annotate genes of interest with relevant biomolecular information (e.g. genomic location, associated transcript isoforms and protein domains, etc.) and related research articles. psichomics also provides the direct link to the cognate entries of relevant external databases, namely Ensembl [151], GeneCards [152], the Human Protein Atlas [153], the UCSC Genome Browser [154], UniProt [148] and VAST-DB [89].

3.2.11 Performance benchmarking

To measure the time taken by psichomics to load data, normalise gene expression, quantify PSIs for skipped exon events and perform global differential expression and splicing analyses between pairs of GTEx v7 tissues and between normal and primary solid tumour samples from multiple TCGA cohorts (data version 2016_01_28 from FireBrowse), the program was run 10 times with the same settings for different combinations of normal human tissues and tumour types in a machine running OS X 10.13.1 with 4 cores and 8GB of RAM, using Safari 11.0.1, RStudio Desktop 1.1.383 and R 3.4.1. The median duration of the 10 runs was used as the performance indicator.

To determine the approximate time complexity of the aforementioned steps in psichomics, gene expression and exon-exon junction quantification datasets were prepared based on approximate distributions obtained from the respective TCGA datasets: negative binomial distributions with a dispersion parameter of 0.25 and 0.2 reads and a mean parameter of 2000 and 100 reads for raw gene expression and exon-exon junction quantification, respectively. Each run was performed on datasets with numbers of samples ranging from 100 to 2500 in intervals of 100 (i.e. 100, 200, 300, ..., 2500) and 20 000 genes or 200 000 splice junctions (gene expression or exon-exon junction quantification, respectively). Splice junction identifiers (required for alternative splicing quantification) were randomly retrieved from the TCGA reference annotation. Based on their respective read counts, around 9000 alternative splicing events (i.e. those for which all involved inclusion and exclusion junctions were retrieved) were quantified across selected samples per run. For differential gene expression and splicing analyses, samples were randomly divided into two groups based on the emitted values of a Bernoulli distribution with a probability of success of 50%.

Polynomials of orders 1–6 were fitted to the relation between running time and the number of samples. As the running time is assumed to always increase with an increasing number of analysed samples, fitted polynomials were constrained to be monotone for 0 or more samples, using function `monpol()` from R package `MonoPoly` [155]. The best polynomial fits (Figure 3.5) were selected based on analyses of variance (ANOVA) between fitted polynomials of consecutive orders, starting with the comparison between polynomials of orders 1 and 2. A polynomial with higher order is only selected if exhibiting a significantly better fit (p-value < 0.05).

3.2.12 Alternative splicing quantification benchmarking

The publicly available RNA-seq data from multiple human, mouse and chicken tissue and cell line samples used in the development of VastDB [89] were aligned with splice-aware STAR [156] against the respective transcript-annotated genomes: UCSC hg19 genome assembly and GENCODE v19 annotation for human, UCSC mm10 genome as-

sembly and GENCODE vM14 annotation for mouse, and Ensembl 70 genome assembly and annotation for chicken. In total, 120/706/34 (human/mouse/chicken) exon skipping events quantified by psichomics (using function `quantifySplicing()` with default settings) were compared with the respective RT-PCR- and VAST-TOOLS-derived PSI values, available from VastDB [89].

Different numbers of junction reads were simulated for different given PSI values to test the impact of read coverage on the accuracy and precision of PSI estimation by psichomics. For each given PSI, junction reads supporting the exon inclusion were simulated as the number of successes obtained from a Bernoulli distribution with the event's junction read coverage (i.e. reads supporting inclusion plus reads supporting exclusion) as the number of observations and the PSI value as the probability of success. Those inclusion reads were then divided by the event's junction read coverage to estimate an 'observed' PSI value (as performed by psichomics) that was compared to the given 'real' PSI value. These simulations were performed for PSI values from 0 to 1 in 0.1 intervals and event coverages of 10, 20, 50, 100, 500 and 1000 junction read counts, with each combination being tested 10000 times.

TCGASpliceSeq [93] provides pre-computed alternative splicing quantifications across TCGA cohorts. As those quantifications are performed similarly by TCGASpliceSeq and psichomics, PSI estimates for each matching (based on genomic coordinates) alternative splicing event and sample from both tools were correlated across the entire TCGA dataset.

3.2.13 Continuous integration

Continuous integration (CI) tools ensure the automatic testing of software in multiple environments (different versions of operating systems, R, BioConductor, etc.). Currently popular CI tools include Travis CI (macOS and Linux, limited support for Windows), AppVeyor (Windows only) and GitHub Actions (Windows, macOS and Linux). Although psichomics was initially set up with Travis CI and AppVeyor, the flexibility of GitHub Actions in running the three main operating systems and the easiness of adding complex routines led me to replace Travis CI and AppVeyor with GitHub Actions.

psichomics has three GitHub Actions scripts. The first one creates Docker images and stores them in GitHub and Docker Hub for every psichomics release or change in the dev branch. The second one updates the package documentation website via `roxygen` and `pkgdown`. The last one builds and checks the R package using `rcmdcheck::rcmdcheck()` and `BiocCheck::BiocCheck()` for every change that is committed to the GitHub repository. psichomics is tested in Windows, macOS and Ubuntu, allowing to automatically check if the package builds correctly and if it passes

all unit tests (created using testthat) in multiple platforms, among other checks. The code coverage of the package is then tested via Codecov. All of these tools are free for open-source projects.

3.3 Results

psichomics offers both a graphical and a command-line interface. Although most features are common to both interfaces, we recommend less experienced users to opt for the Shiny-based graphical interface [112]. To start the graphical interface, the user is required to load the psichomics package in R and run function `psichomics()`, resulting in the automatic launch of the user's default web browser and of the program's graphical interface as a local web app. p

3.3.1 Case study

3.3.2 Time benchmarking

The times required to load, quantify and analyse data from different TCGA (data version 2016_01_28 from FireBrowse) and GTEx v7 cohorts were benchmarked. The breast cancer cohort contains the highest number of RNA-seq samples available in TCGA, thus being that for which it takes more time to load, quantify and analyse alternative splicing and gene expression data. Contrastingly, processed data from GTEx come bundled in files containing all tissues. Although only data from specified tissues are loaded, scanning though the large GTEx file still delays data loading. Tissues from GTEx were loaded in pairs for subsequent differential splicing analyses (Figure 3.5A).

Synthetic datasets for gene expression and exon-exon junction quantification of multiple sample sizes were generated, based on TCGA data distributions, to determine the time complexity of each step in psichomics as a function of the number of input samples s (Figure 3.5B). Assuming a constant number of genes (20 000 in the benchmark) or exon-exon junctions (200 000), the time taken to load data grows quadratically with s. Gene expression normalisation and differential expression are based on commonly-used, time-efficient bioinformatics tools and the times taken for each also grow quadratically with s. Alternative splicing quantification is associated with element-wise operations on matrices of dimensions s by the number of alternative splicing events and takes a runtime approximately proportional to the square of s, for a given number of alternative splicing events (around 9000 for each benchmarked run). Finally, differential splicing is based on multiple, distinct statistical analyses of alternative splicing quantification data and grows linearly with s.

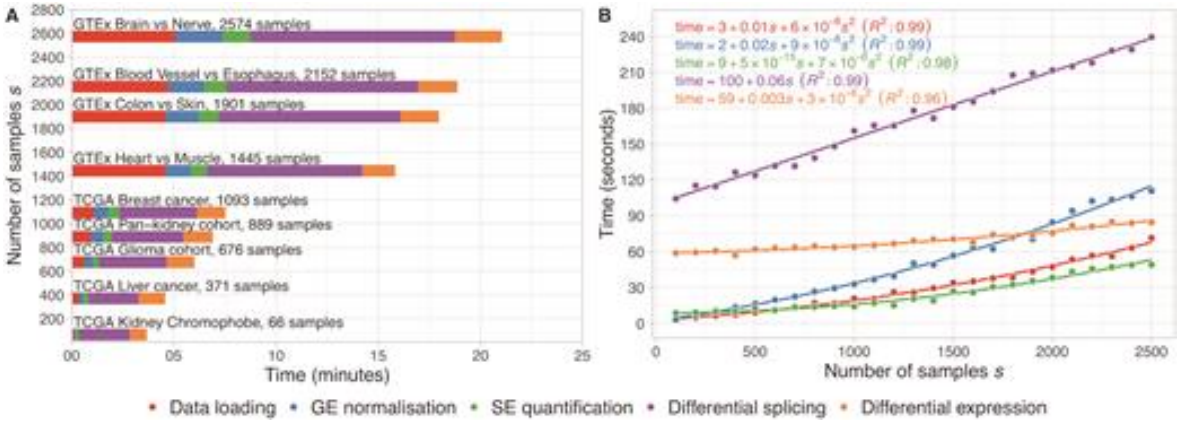

Figure 3.5: Performance benchmark for alternative splicing analysis using RNA-seq data from multiple TCGA and GTEx sample types. (A) Median times of 10 runs of data loading, gene expression (GE) normalisation, skipped exon (SE) event quantification and differential expression and splicing analysis (normal versus tumour for TCGA data or pairwise tissue comparison for GTEx data) using psichomics. The default settings were used during the runs. (B) Estimation of the time complexity of each of the aforementioned steps in psichomics. Randomly generated synthetic datasets of different sample size s were used as input. Equations and coefficient of determination (R^2) for the best fits are displayed.

3.3.3 Alternative splicing quantification benchmarking

Although jSplice's [133] and DIEGO's [134] splicing quantifications rely on junction read counts, their alternative splicing module expression and junction usage metrics, respectively, are not directly comparable with psichomics' PSI values. To evaluate their accuracy in the absence of any known tool with the same input (junction read counts) and output metric (PSI) as psichomics, psichomics-estimated PSI values were compared to those estimated by RT-PCR and using VAST-TOOLS [88] across multiple tissue and cell line samples from human, mouse and chicken [89]. VAST-TOOLS follows an analogous, and therefore more directly comparable, procedure for computing PSI values and there is a substantial overlap between the alternative splicing event annotations used by the two tools. psichomics estimates highly correlate with both others, particularly for mouse and human (Supplementary Figure S7), suggesting robustness and reproducibility in alternative splicing quantification by psichomics. Of note, the lower correlation for chicken samples is attributable to a single outlier, as its removal increases the correlation coefficients between psichomics and RT-PCR estimates (Pearson's r = 0.87, p-value < 0.01; Spearman's rho = 0.87, p-value < 0.01) and psichomics and VAST-TOOLS estimates (Pearson's r = 0.93, p-value < 0.01; Spearman's rho = 0.94, p-value < 0.01).

To assess the influence of RNA-seq read coverage on psichomics PSI estimates, different numbers of junction reads per event were simulated for different given PSI values (10000 times for each combination). Supplementary Figure S8 shows that the accuracy of PSI estimation by psichomics is expectedly sensitive to junction read coverage, par-

ticularly for intermediate PSI values, with 90% prediction intervals < 0.1 for coverage higher than a few hundred reads.

Alternative splicing events annotated by TCGASpliceSeq [93], an online tool that displays pre-computed PSI values across multiple TCGA tumour types, were matched to those from psichomics based on their genomic coordinates. In total, 321 183 of 757 749 (42%) skipped exon, 70 837 of 126 725 (56%) alternative $5'$ splice site and 90 940 of 155 799 (58%) alternative $3'$ splice site events were successfully matched. When available from both programs, PSI estimates for each of the 482 960 alternative splicing events in each of the 9 913 matched samples were compared between TCGASpliceSeq and psichomics, being highly correlated (N = 92 444 302; Pearson's r = 0.97, p-value $< 10^{15}$; Spearman's rho = 0.94, p-value $< 10^{15}$; Supplementary Figure S9).

3.3.4 Impact

- Invitation to write a book chapter

Although there is no direct way to measure user engagement with psichomics, we can track:

- Bioconductor downloads
- Docker Hub image downloads
- visitors to the official psichomics documentation
- visitors to the website
- feedback via GitHub issues and emails
- citations

3.4 Conclusion

Alternative splicing is a regulated molecular mechanism involved in multiple cellular processes and its dysregulation has been associated with diverse pathologies [157, 158, 122, 159]. The advent of next-generation sequencing technologies has allowed the investigation of transcriptomes of human biological samples to be expanded to alternative splicing. RNA-seq data, like those yielded by the GTEx and TCGA projects, are indeed playing crucial role in the improvement of our insights into the role of alternative splicing in both physiological and pathological contexts [158, 122, 160, 123, 124]).

However, the most commonly used tools for alternative splicing analyses currently do not allow researchers to fully benefit from the wealth of pre-processed RNA-seq data made publicly available by the aforementioned projects. For instance, they lack support for estimating PSIs based on splice junction read counts. Such functionality would allow users to overcome the difficulties caused by the raw RNA-seq data from GTEx and TCGA being under controlled access and, more importantly, their processing requiring

computational resources inaccessible to the majority of research labs. psichomics thus exploits pre-processed alternative splicing annotation and exon–exon junction read count data from TCGA and GTEx, two of the richest sources of molecular information on human tissues in physiological and pathological conditions, as well as recount2 and user-owned data, allowing researchers to hasten alternative splicing quantification and subsequent analyses by avoiding the time-consuming alignment of RNA-seq data to a genome or transcriptome of reference followed by splice junction detection.

Together with support for the integration of molecular and sample-associated clinical information, the group creation functionalities featured in psichomics ensure full customisability of data grouping for downstream analyses. Interesting groups to compare in TCGA, for instance, may range from the simple contrast between reformed and current smokers in lung cancer to complex combinations of gender, race, age, country and other subject attributes across multiple cancers. When survival data are available, survival analyses can be performed on samples by PSI or gene expression levels, thereby assessing the putative prognostic value of a respective molecular feature.

The integrative analysis of publicly available TCGA data by psichomics allowed us to identify multiple exons differentially spliced between breast tumour stage I and normal samples, therefore deeming them potential diagnostic biomarkers, and to assess their putative prognostic value. The output of psichomics is validated by identified alternative splicing alterations that have been previously linked to the disease, including events in RPS24, NUMB, FBLN2 and AP2B1. Previously understudied, yet intriguing, events were also identified, such as the skipping of SLMAP exon 23 and UHRF2 exon 10. These may provide novel insights into the early stages of breast cancer development. Indeed, it is of utmost importance to foster alternative splicing analyses of clinical samples as a crucial complement to more conventional research focused on total gene expression.

To ensure researchers with different skills can take the most out of psichomics, users lacking a computational background may feel more comfortable using the intuitive and more accessible graphical interface, whereas advanced users may opt for the command-line view. Should the demanding computational resources for hosting psichomics in a web server become available, we also envisage its web deployment, so that the program's latest version is publicly available on-demand with no installation required, levering the intuitive graphical interface to make alternative splicing analyses more enticing to less computationally-inclined biomedical researchers.

Notwithstanding its merits, a current limitation of psichomics is the current support only for events quantified based on exon–exon junction read counts, as not all types of alternative splicing events can be profiled using splice junction reads alone. For instance, exon–intron junction, exon body and intron body quantifications are vital to confirm intron retention and alternative $5'$ and $3'$ UTR events over further transcrip-

tional variations [138]. However, although GTEx (but neither TCGA nor recount2) readily provides intron and exon body read quantification for retrieval, none provides exon–intron junction quantification.

Another limitation of psichomics is its reliance on existing alternative splicing event annotations and an on the pre-processing of RNA-seq data by third-party pipelines (as is the case for GTEx, TCGA and recount2), depriving the user of the flexibility to identify de novo alternative splicing events. However, as we detail in `nuno-agostinho.github.io/psichomics/articles/AS_events_preparation`, when FASTQ or BAM files are accessible, psichomics supports the loading of alternative splicing annotations generated by different programs that take those files as input, namely rMATS [90], which is able to generate de novo annotations.

Using psichomics, we are able not only to identify novel exons differentially spliced between tumour stage I and normal breast samples but also to pinpoint potentially clinically relevant splicing events by embracing clinical data and evaluating their prognostic value. We expect that fellow researchers and clinicians will be able to intuitively employ psichomics to assist them in uncovering novel splicing-associated prognostic factors and therapeutic targets, as well as in advancing our understanding of how alternative splicing is regulated in physiological and disease contexts.

Chapter 4

cTRAP

Our lab had a brainstorm session during a stormy day in our 2017 Madeira retreat, where we discussed the unique propositions that the lab could provide to the scientific community.

One idea that emerged was to make it easier to identify putative causal perturbations by comparing a custom differential gene expression against the large-scale database of differential expression profiles from CMap [97], a repository of transcriptomic signatures for thousands of genetic (gene overexpression or knockout) and pharmacological perturbations in human cancer cell lines. This kind of analysis is available in CMap's web apps at `clue.io` [97], but their issues include difficulty in automating their workflow to use their results in downstream analyses, poor API documentation for proper programatic access and lack of basic visualisation tools.

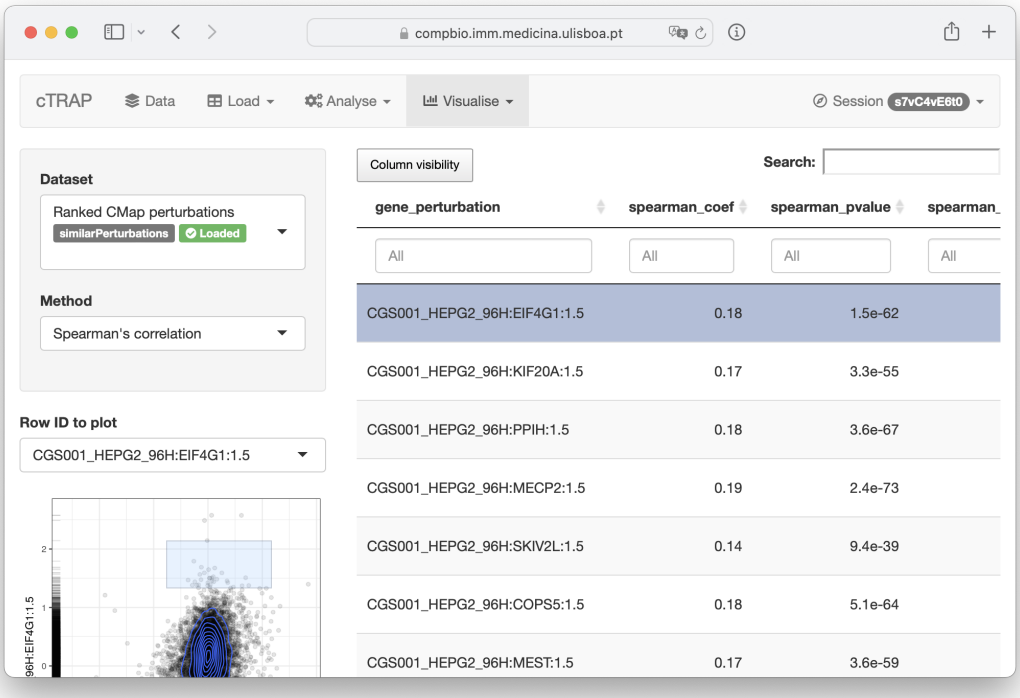

Figure 4.1: cTRAP screenshot (21 Dec 2021).

To overcome those limitations, we developed cTRAP, an R package to compare user-provided differential gene expression profiles with those from CMap, allowing to infer putative candidate molecular causes for the observed differences, as well as compounds that may promote or revert them.

After releasing the first version in Bioconductor, multiple features were added (Table 4.1). We ended up including a way to predict targeting drugs by using drug sensitivity datasets that featured gene expression for multiple genes in many cell lines. Moreover, we added a GSEA-based analysis of the enrichment of molecular descriptors for compounds from NCI60 and CMap. More recently, we developed a visual interface to be hosted online, with support for user sessions and background tasks.

Table 4.1: Major cTRAP milestones.

Version	Release date	Main features
1.0.0	2 Nov 2018	Compare differential expression profiles against CMap data[a]
1.4.0	12 Nov 2019	Predict targeting drugs using NCI60, CTRP and GDSC data Analyse drug set enrichment using molecular descriptors
1.8.0	30 Oct 2020	Include graphical functions to load data and analyse results
1.10.0	20 May 2021	Improve speed and memory usage when comparing data
1.12.0	28 Oct 2021	Add web server support (optimised to run in ShinyProxy)[b]

[a] First Bioconductor release. [b] First version available online.

The associated cTRAP manuscript (of which I am a co-first and co-corresponding author) is in preparation for submission to an international peer-reviewed scientific journal and shares some similarities with this chapter.

4.1 Background

The Connectivity Map (CMap) is a repository of transcriptomic signatures of thousands of genetic and pharmacological perturbations in human cancer cell lines [97]. Comparing differential gene expression profiles with those from CMap allows to infer putative molecular causes for the observed differences, as well as compounds that may promote or revert those changes.

The CMap and LINCS Unified Environment (`clue.io`) was developed as a collection of user-friendly tools for the manipulation of CMap data and their integration with user-provided data [97]. However, `clue.io` limits the maximum number of input genes for CMap queries, expresses results' significance in a non-standard significance score, is difficult to automate for downstream analyses and does not support using local computing resources. Furthermore, `clue.io` does not currently integrate with drug sensitivity datasets, which could further assist in pinpointing compounds that selectively target cells.

We thus developed cTRAP, an R package and web app that identifies potentially causal molecular perturbations by seamlessly comparing full user-provided differential gene expression results with those available from CMap. cTRAP also supports comparisons with gene expression/drug sensitivity associations derived from the NCI-60 [101], the Cancer Therapeutics Response Portal (CTRP) [102] and the Genomics of Drug Sensitivity in Cancer (GDSC) [103], to identify compounds that could target the phenotypes associated with the user-provided differential expression profiles. In cTRAP, similarity is measured by gene set enrichment [97, 161] and correlation scores.

cTRAP can be installed using Bioconductor (`bioconductor.org/packages/cTRAP`) or Docker (`nunoagostinho/ctrap`). The latest version of cTRAP is also available online at `compbio.imm.medicina.ulisboa.pt/cTRAP`. The source code of cTRAP is available at `github.com/nuno-agostinho/cTRAP`.

4.2 Materials and methods

From a vector of user-provided differential expression results (e.g. t-statistic values) with respective gene symbols, cTRAP can return a ranked list of similar CMap perturbations or predict targeting drugs. Moreover, cTRAP can also analyse the enrichment of drug sets in an ordered vector of compounds to identify common compound characteristics (Figure 4.2).

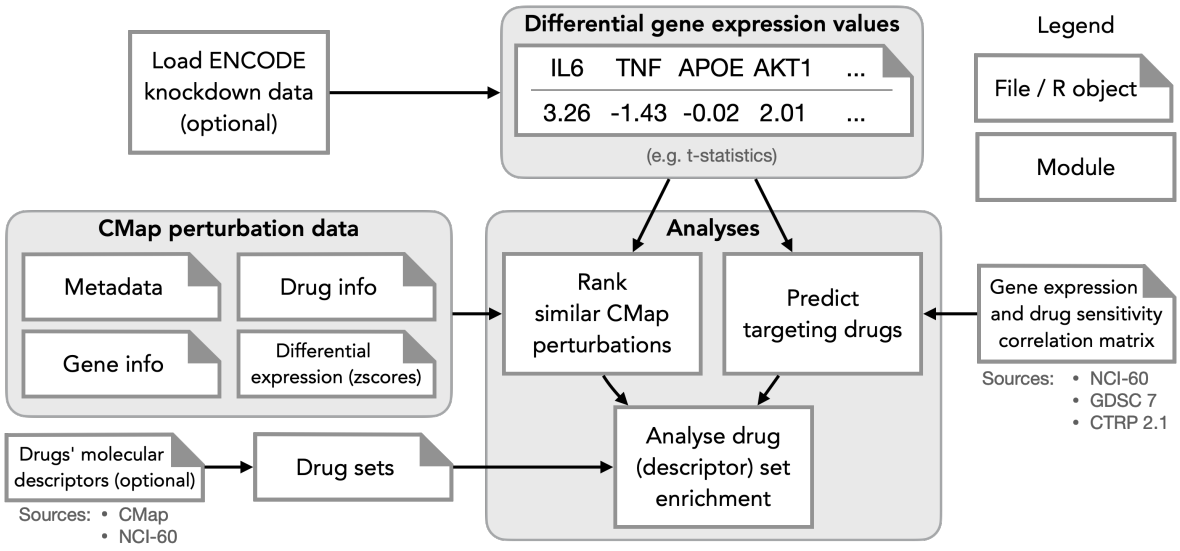

Figure 4.2: cTRAP workflow. cTRAP allows to perform three analyses: **(1) rank similar CMap perturbations** by comparing user-provided differential gene expression values against CMap perturbation data, **(2) predict targeting drugs** by comparing user-provided differential gene expression values against correlation matrices of gene expression and drug sensitivity data and **(3) analyse drug (descriptor) set enrichment** using drug sets and the results from either the first or second analysis. CMap perturbation data, gene expression/drug sensitivity correlation matrices and drug molecular descriptors for drug sets can be automatically downloaded.

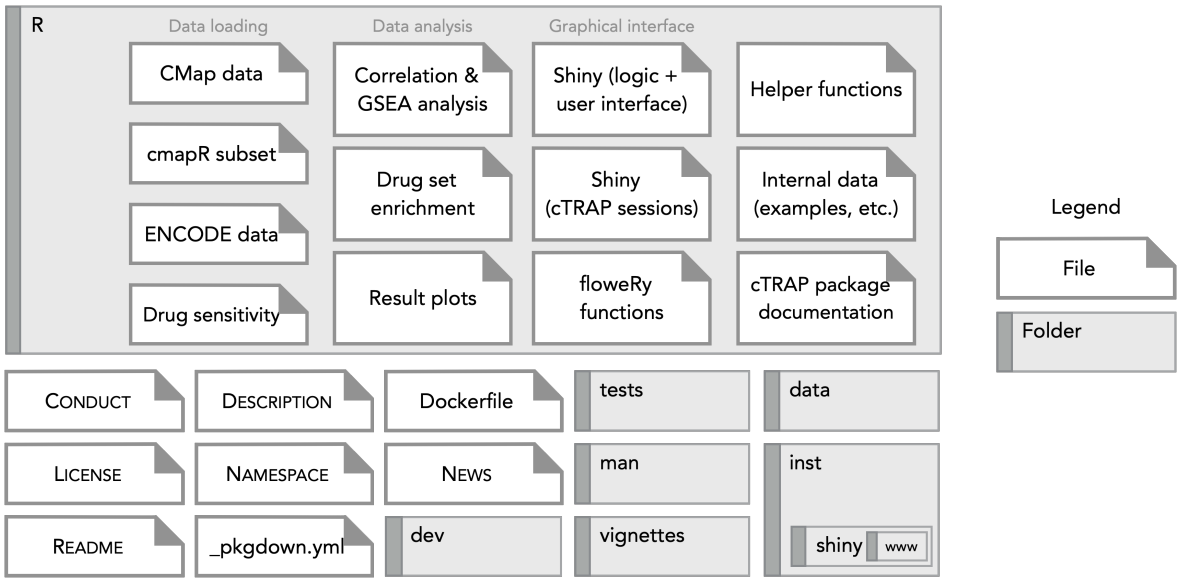

Figure 4.3: Visual representation of cTRAP's file structure. As usual in an R package, the `R` folder contains the scripts with cTRAP functions and data. `dev` is a custom folder that stores supporting scripts (e.g. test workflows and benchmarks); its contents are not included when building the R package.

4.2.1 ENCODE knockdown data

Using cTRAP, we can query and download ENCODE knockdown (and respective control) samples for multiple cell lines, filter low coverage counts from gene expression data, convert from ENSEMBL identifiers to gene symbol, and perform differential gene expression using `voom()`, `lmFit()` and `eBayes()` from the `limma` R package [146]. First, `voom()` is used with the *quantile* normalisation to transform count data to $\log_2$ CPM (counts per million) and estimate the mean-variance relationship to compute weights used in linear modelling. Gene-wise linear models are then fitted using `lmFit()` between the knockdown and the control samples, followed by moderated t-tests and the calculation of log-odds of differential expression, using `eBayes()` for empirical Bayes moderation of standard errors.

cTRAP includes an example dataset (`diffExprStat`) with the differential gene expression results (t-statistic values) between the EIF4G1 knockdown in HepG2 versus control (Listing 4.1).

Listing 4.1: Code to obtain example dataset `diffExprStat`.

```
library(cTRAP)
ENCODEmetadata <- downloadENCODEknockdownMetadata(cellLine="HepG2",
                                                  gene="EIF4G1")
ENCODEsamples  <- loadENCODEsamples(ENCODEmetadata)[[1]]
counts         <- prepareENCODEgeneExpression(ENCODEsamples)

# Remove low coverage genes (>= 10 counts shared by >= 2 samples)
```

```
minReads   <- 10
minSamples <- 2
filter     <- rowSums(counts[ , -c(1, 2)] >= minReads) >= minSamples
counts     <- counts[filter, ]

# Convert ENSEMBL identifiers to gene symbols
counts$gene_id <- convertGeneIdentifiers(counts$gene_id)

# Perform differential gene expression (DGE) analysis
diffExpr <- performDifferentialExpression(counts)

# Get t-statistic values of DGE and respective gene names
diffExprStat <- diffExpr$t
names(diffExprStat) <- diffExpr$Gene_symbol
```

4.2.2 Ranking of similar CMap perturbations

CMap perturbations can be categorised into gene knockdown, gene over-expression and compounds. In cTRAP, available perturbation types and respective conditions can be enquired using the function `getCMapConditions()` that will download CMap perturbation metadata. Afterwards, the function `filterCMapMetadata()` allows to filter the metadata based on selected perturbations types, cell lines, dosages and time points, allowing to specifically load only the desired data in downstream analyses. This information is passed to `prepareCMapPerturbations()` to download (if file is not found) and process CMap differential expression profiles z-scores (GCTX file) and gene and compound information. Given that the GCTX file size is around 21GB, we recommend to download the file directly from GEO GSE92742's Level 5 data link (`ftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE92nnn/GSE92742/suppl/GSE92742_Broad_LINCS_Level5_COMPZ.MODZ_n473647x12328.gctx.gz`).

After comparing differential expression z-scores from select CMap perturbations against user-provided differential expression results, `rankSimilarPerturbations()` returns a table with ranked CMap perturbations and their respective correlation coefficients and GSEA scores. Lower ranks indicate perturbations whose differential expression profiles are more similar to the user-provided data, i.e. CMap perturbations that potentially mimic the user-provided transcriptomic changes, whereas higher ranks define perturbations that may revert those changes.

To rank CMap perturbations, cTRAP performs Spearman's and Pearson's correlations between the user-provided statistics for differential expression and values from CMap perturbations, and calculates a GSEA-based score (all three methods are run by default). For each method, the similarity scores are averaged across multiple cell lines for the same conditions (when available) and those averages are then used to rank

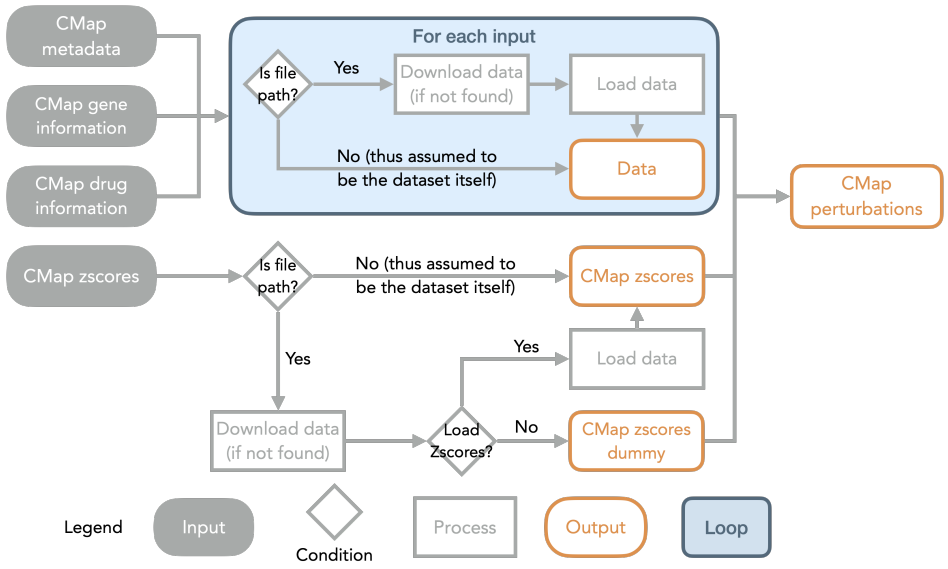

Figure 4.4: Loading data from CMap perturbations. Input arguments support either the data itself (as data frames) or their respective file path. If the file path directs to a non-existing file, data is first downloaded and then saved to the given file path. To avoid high memory usage, CMap perturbations' z-scores of differential expression profiles (CMap zscores) are not loaded into memory when a file path is given. Instead, only metadata are loaded into a *dummy* object that can be subset as a normal R object for downstream analyses.

CMap perturbations. By default, results for individual cell lines are provided for informative purposes (e.g. to check the heterogeneity of response across cell lines) but not used when ranking. The different ranking scores are combined via the rank product, ultimately used to sort the CMap perturbations.

The GSEA-based score is calculated via the following steps:

1. Sort genes from the user-provided differential expression statistics;
2. Define the top 150 (by default) and bottom 150 (by default) genes as two sets
3. For each CMap perturbation, sort genes by their differential expression z-scores and calculate the Weighted Connectivity Score (WTCS) [97] based on the GSEA enrichment scores for the two sets.

As an example, for a CMap perturbation with a similar differential expression profile to user's input, we expect to find higher enrichment of the top gene set in the most up-regulated genes and higher enrichment of the bottom gene set in the most down-regulated genes.

To minimise peak RAM usage, `prepareCMapPerturbations()` downloads the GCTX file (technically, a customised HDF5 file) for the CMap's perturbation differential expression z-scores (if not previously downloaded) and returns its path without loading the file content itself, creating a *dummy* object that only stores its file path, perturbation names, gene symbols and other associated metadata (Figure 4.4). Based

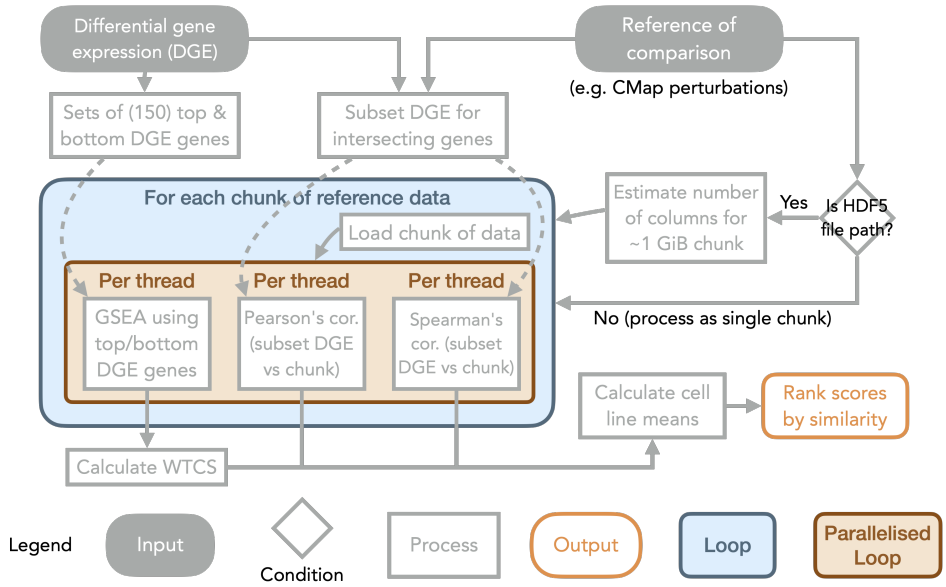

Figure 4.5: cTRAP similarity analysis. User-provided differential gene expression is compared against a reference (e.g. differential expression z-scores of CMap perturbations) and ranked by similarity. If the reference of comparison is a HDF5 file path, the file is processed in 1GiB chunks to minimise peak memory usage. These analyses support multiple threads in Linux and macOS.

on the file path of this *dummy* object (that can be subset like a normal R object), `rankSimilarPerturbations()` loads a $\leq$ 1 GiB chunk[1], compares its differential expression z-score values against user-provided data and repeats the analysis for the next chunk (Figure 4.5). For each chunk, multithreaded support for Linux and macOS can be enabled for each comparison method via the `mclapply()`[2] function from R package `parallel` that can enabled by setting the number of threads to 2 or higher.

The ranked list from `rankSimilarPerturbations()` can be plotted using `plot()`, showing a list of all results ordered by a given score or either a scatterplot or GSEA plot for a predicted targeting drug.

4.2.3 Prediction of targeting drugs

Gene expression and drug activity data across multiple cell lines are available from NCI-60 [101], Cancer Therapeutics Response Portal (CTRP) 2.1 [102] and Genomics of Drug Sensitivity in Cancer (GDSC) 7 [103]. For each source, the internal function `prepareExpressionDrugSensitivityAssociation()` performs the following steps:

1. Download all the necessary data depending on given source;

[1]The default 1 GiB (1024^3 bytes) allows loading chunks of around 10000 columns and 14000 rows ($10000 \times 14000 \times 8$ bytes$/1024^3 = 1.04$ GiB). CMap's GCTX file has around 14000 rows (genes).

[2]`mclapply()` allows parallel computing via forking where child processes are spawned from a parent process and share their memory. Forking is unavailable in Windows and current multithreaded alternatives need multiple copies of the same data chunk for each thread, requiring multiple slow copy operations for 1GiB chunks that slowed down the runtime of the analyses when prototyping.

2. Perform Spearman's correlation (by default) between the expression of each gene against the sensitivity of intersecting cell lines to each drug;

3. Generate a matrix with the correlation coefficients per gene and drug; and

4. Prepare metadata for downstream analyses, including gene, compound and cell line information from each source.

A higher correlation coefficient for a given gene and drug suggests a gene whose higher expression is associated with higher drug sensitivity across multiple cell lines. As this process can take multiple hours to finish for all sources, the resulting objects were stored online for each aforementioned source and can be listed with `listExpressionDrugSensitivityAssociation()` and downloaded and loaded into R using `loadExpressionDrugSensitivityAssociation()`.

To identify compounds that could target the phenotype associated with specific differential expression profiles, we use `predictTargetingDrugs()` with those profiles and a correlation matrix of gene expression and drug sensitivity as input. The correlation coefficients between gene expression and drug sensitivity for each drug are compared against user-provided differential expression results by Spearman's and Pearson's correlation and GSEA-based scores (as performed when ranking CMap perturbations, results from comparison methods are ranked and then those rankings are finally used to calculate the rank product's rank). `predictTargetingDrugs()` returns a table with ranked predicted targeting drugs and their respective correlation coefficients and GSEA scores (Figure 4.5). A lower rank comprise drugs that may target phenotypes similar to the user-provided differential expression profile.

The resulting object can be plotted with `plot()`, showing a list of all results ordered by a given score or either a scatterplot or GSEA plot for a predicted targeting drug.

The function `plotTargetingDrugsVSsimilarPerturbations()` compares the results from predicted targeting drugs and CMap perturbations that may mimic or revert the observed phenotype. For the available compound identifiers in the metadata pertaining from the different datasets (e.g. compound name, Broad ID, PubChem CID and SMILES), the function will automatically select the identifiers with higher number of matching values between the two datasets, unless the identifiers are explicitly defined by the user. A scatterplot is then returned using, by default, the rank product's rank of targeting drugs in one axis and the rank product's rank of similar perturbations in the other.

4.2.4 Drug descriptor set enrichment analysis

Juan Carlos, a former member of the lab, computed drug descriptors (e.g. molecular weight and number of aromatic rings) for compounds from CMap and NCI-

60 based on their three-dimensional (3D) and two-dimensional (2D) characteristics. These descriptors were uploaded to `compbio.imm.medicina.ulisboa.pt/public/cTRAP/` and the resulting files can be automatically downloaded and processed to R using `loadDrugDescriptors()`.

`prepareDrugSets()` allows to create sets of descriptors. By default, the function creates a maximum of 15 sets per drug descriptor. For each alphanumeric descriptor, one set is created per unique value of that descriptor. Alphanumeric descriptors containing more than 15 unique values (by default) will be discarded. For numerical descriptors, `prepareDrugSets()` internally uses the `binr::bins()` function to create evenly-distributed bins of drug descriptors, where each set contains a minimum number of points equal to the number of non-missing values divided by the number of maximum sets (15 by default) divided by a constant (5 by default).

The function `analyseDrugSetEnrichment()` analyses the enrichment of the created drug descriptor sets in a named numeric vector or an object returned from `rankSimilarPerturbations()` – only if run against CMap compound perturbations – or `predictTargetingDrugs()`. The GSEA-based enrichment analysis is internally performed using `fgsea::fgsea()`.

The resulting object can be plotted with `plot()`, showing a list of all results ordered by a given score or either a scatterplot or GSEA plot for a predicted targeting drug.

4.2.5 Time and memory benchmarking

We measured elapsed time using R's `Sys.time()` immediately before and after ranking similar CMap perturbations, predicting targeting drugs (using NCI60 expression and drug sensitivity association, the most time-consuming option) and performing drug set enrichment analysis using cTRAP 1.8.1 (296f9b21). As input, we used the t-statistics for the differential expression between EIF4G1 knockdown versus control based on ENCODE gene expression data from cell line HepG2 (the `diffExprStat` object in the cTRAP package).

We measured the heap memory usage of cTRAP 1.8.1 (296f9b21) across time by running R 4.0.3 in debug mode with the heaptrack 1.0.0 profiler. heaptrack tracks and logs all calls to the core memory allocation functions via `LD_PRELOAD` and respective backtraces. For R to work properly with heaptrack, the file `/usr/bin/R` was edited – all lines of the last *if* statement were commented out, except for:

```
exec ${debugger} ${debugger_args} "${R_binary}" ${args} "${@}"
```

Afterwards, we benchmarked R scripts running cTRAP with:

```
R -d heaptrack -f ${cTRAP_Rscript} --args ${cTRAP_Rscript_args}
```

All benchmarks were run in a workstation running Ubuntu 18.04.5 LTS with 768 GB of RAM memory and 72 cores (Intel Xeon Gold 6254 CPU @ 3.10GHz). The

benchmarked cTRAP scripts are publicly available in cTRAP's GitHub repository: `github.com/nuno-agostinho/cTRAP/tree/master/dev/benchmark`

4.2.6 Continuous integration

Similarly to psichomics (subsection 3.2.13: **Continuous integration**), cTRAP uses GitHub Actions to:

- Update Docker images in Docker Hub (`nunoagostinho/ctrap`) and GitHub (`github.com/nuno-agostinho/cTRAP`);
- Update website documentation using `roxygen` [110] and `pkgdown` [162];
- Build and test cTRAP in Windows, macOS and Linux.

4.3 Results

4.3.1 Case study

4.3.2 Time and memory optimisation

Show benchmarks.

4.3.3 Graphical interface

One of the most recent features is the support for visual interfaces that interactively perform most cTRAP features via the web browser. The graphical interface was modularly built and exposed via 5 functions that work harmoniously with R code:

- `launchDiffExprLoader()` to load differential expression data. Returns a differential expression object that can be used in cTRAP analyses.
- `launchCMapDataLoader()` to explore and load CMap data by type of perturbation, cell types, time points and dosages. Returns filtered CMap data based on the user's selection.
- `launchMetadataViewer()` to check metadata of given cTRAP objects.
- `launchResultPlotter()` to view and plot cTRAP results given as input.
- `launchDrugSetEnrichmentAnalyser()` to analyse drug set enrichment and visualize respective results.

Like usual R functions, these graphical interfaces functions accept input and may return output. Thus, they can be intertwined with R code, allowing to easily reproduce cTRAP analyses (Listing 4.2).

Listing 4.2: Calling CTRAP's graphical interface functions in an R script.

```
# Launch differential expression loading interface to select knockdown
# data from ENCODE (pre-filtered for HepG2 cell line and EIF4G1 gene)
diffExpr <- launchDiffExprLoader(cellLine="HepG2", gene="EIF4G1")

# After filter selection, launchDiffExprLoader() does the following:
# 1. Download ENCODE's HepG2 data for EIF4G1 knockdown and controls
# 2. Perform DGE between EIF4G1 knockdown vs. control
# 3. Return resulting t-statistics by gene

# Load CMap knockdown data in HepG2
cmapKD <- launchCMapDataLoader(
    cellLine="HepG2",
    perturbationType="Consensus signature from shRNAs targeting the
   same gene")
# Load CMap compound data in HepG2
cmapCompounds <- launchCMapDataLoader(cellLine="HepG2",
                                      perturbationType="Compound")
# Load all CMap data in HepG2
cmapPerts <- launchCMapDataLoader(cellLine="HepG2")

# View metadata of all resulting CMap data objects
launchMetadataViewer(cmapKD, cmapCompounds, cmapPerts)

# Rank similar perturbations ----------------------------------------
compareKD        <- rankSimilarPerturbations(diffExpr, cmapKD)
compareCompounds <- rankSimilarPerturbations(diffExpr, cmapCompounds)
comparePerts     <- rankSimilarPerturbations(diffExpr, cmapPerts)

launchResultPlotter(compareCompounds, compareKD, comparePerts)

# Predict targeting drugs -------------------------------------------
listExpressionDrugSensitivityAssociation()
assocMatrix <- listExpressionDrugSensitivityAssociation()[[1]]
assoc       <- loadExpressionDrugSensitivityAssociation(assocMatrix)
predicted   <- predictTargetingDrugs(diffExpr, assoc)
launchResultPlotter(predicted)

# Plot targeting drugs vs similar perturbations ---------------------
launchResultPlotter(predicted, compareCompounds)

# Analyse drug set enrichment ---------------------------------------
descriptors <- loadDrugDescriptors("NCI60", "3D")
```

```
42 drugSets     <- prepareDrugSets(descriptors)
43
44 launchDrugSetEnrichmentAnalyser(drugSets, compareCompounds)
45 launchDrugSetEnrichmentAnalyser(drugSets, predicted)
```

In order to increase its usefulness to the scientific community, cTRAP is available online[3] with a comprehensive interface that provides most aforementioned features in a single app via a sixth function: `cTRAP()`. A clear question arrived with such a strategy: how to deal with long-running tasks? The way R/Shiny is built, an entire cTRAP session would be consuming useful resources during the cTRAP analyses, but this would not properly scale for multiple users using heavy memory resources simultaneously. To avoid this, long-running tasks are put in a queue depending on available resources and performed in the background. But this also meant that the users would need get their results back once finished calculating. And thus the idea of user sessions was born.

User sessions

When visiting cTRAP, a user is greeted with a welcome screen that allows to create a new session or restore a previous one (Figure 4.6). If the user clicks to create a new session, a random alphanumeric string (token) is created. The token will be the name of the folder storing user data and cTRAP ensures that the token is unique and no other folder is currently using it (Figure 4.7).

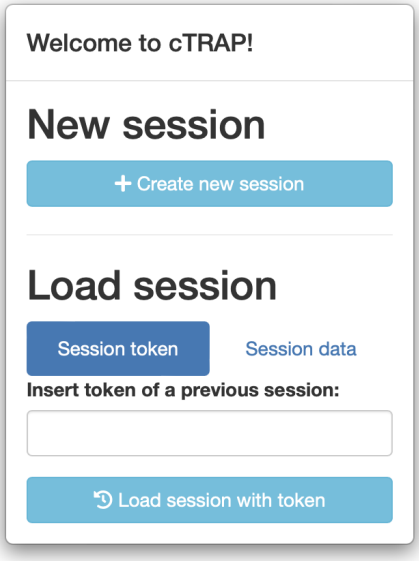

Figure 4.6: Welcome modal.

If the user loads data to their session, a new folder is named after the session token (Figure 4.7). All updates to the session data are immediately saved to the session folder. To avoid downloading commonly-used files across sessions (e.g. the big 21GB CMap perturbations z-scores file), those files can be made available in a folder accessible to all sessions, thus skipping the step of downloading and preparing the data.[4]

While using cTRAP, the user can create a new session and load a previous session via a token or a RDS file at any time. When using the token, cTRAP loads the contents of the folder named after the token – if no such folder exists, it will warn the user (Figure 4.7). In case the user uploads a RDS file, cTRAP will create a new session and load the contents of the RDS file as the data session (Figure 4.7). Using an RDS file ensures the user can open the data in an R session in their local computer given

[3] More information in chapter 5: **CompBio app server**
[4] This is how cTRAP is configured in our web server.

that this RDS file simply contains a list of all the datasets available in the session or even use this object in a local version of cTRAP. Moreover, as sessions start to accumulate in our server, they may be removed from the system and thus may become inaccessible[5].

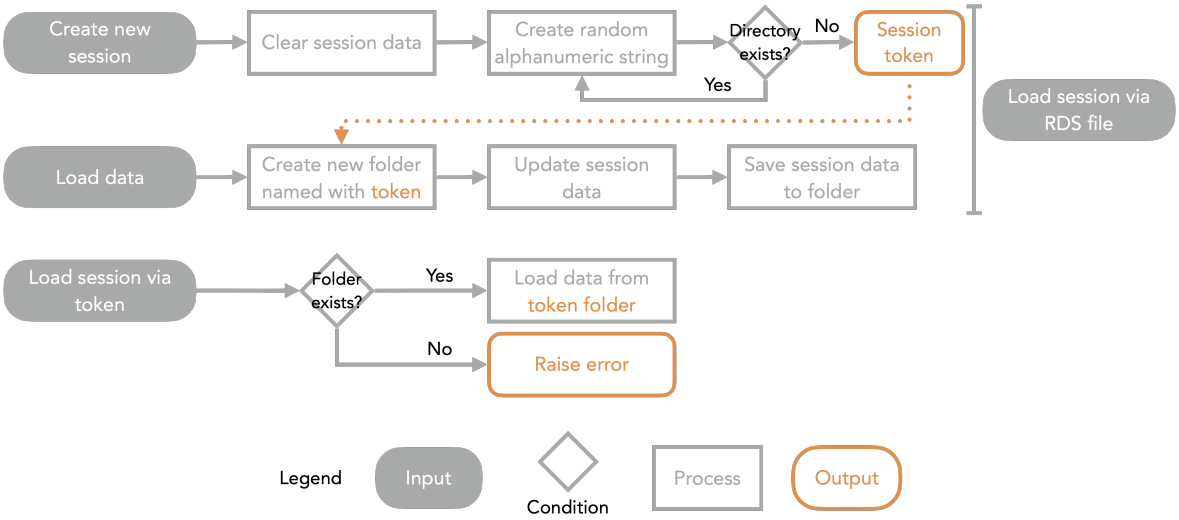

Figure 4.7: User session workflow. cTRAP allows to create new sessions or load a previous one via a given token or RDS file.

Background tasks

As usual in an R/Shiny app, cTRAP runs tasks in the same R process as the app itself. The user has to wait for the long-running tasks to finish before being able to interact again with the app. Critically, if cTRAP is shut down, all running processes will stop. These issues can be overcomed by running tasks in the background using:

- Celery, a task queue manager written in Python; and
- Flower, a monitoring app that provides an HTTP API for Celery.

In order to run cTRAP analyses via Celery, both cTRAP and Celery must be installed in the same system. Celery requires a `tasks.py` file with its tasks defined. For cTRAP, one of these tasks must be set up to run R commands via Python's `subprocess` module to spawn new processes and using the `Rscript` front-end (Listing 4.3).

Listing 4.3: An example `tasks.py` file to run R commands or Rscript files via Celery.

```
1 import os, time
2 from datetime import datetime
3 from subprocess import run, PIPE
4
```

[5]We have considered implementing a 14-day expiration period for user sessions. This is not currently in place.

```
# Celery configuration
from celery import Celery
os.environ.setdefault('C_FORCE_ROOT', 'true')
app = Celery(
    "tasks",
    broker=os.environ.get('CELERY_BROKER_URL', 'redis://redis'),
    backend=os.environ.get('CELERY_RESULT_BACKEND', 'redis://redis'))
app.conf.CELERY_WORKER_SEND_TASK_EVENTS = True

# Runs R command and returns output
#   - Use cat(), e.g. 'cat(2+2)', to capture output as a job result
#   - Errors will result in a task state of FAILURE
def execR(cmd):
    return run(cmd, check=True, stdout=PIPE, text=True).stdout

# Run a given R expression as a Celery job
@app.task
def R(cmd): return execR(["Rscript", "-e", cmd])

# Run a given Rscript file as a Celery job
@app.task
def Rscript(cmd): return execR(["Rscript", cmd])

if __name__ == "__main__": app.start()
```

Celery can receive its jobs via HTTP methods from Flower, facilitating the communication between cTRAP and Celery. To assist using Flower's HTTP API, I created the R package floweRy (`github.com/nuno-agostinho/floweRy`) to simplify sending jobs to Celery using R, as briefly demonstrated in Listing 4.4 and Listing 4.5.

Listing 4.4: Using plain `httr`.

```
library(httr)
flower <- function(...) paste0(
    "http://localhost:5555", ...)
# Run R command '3 + 4' in Celery
POST(flower("/api/task/apply/
    tasks.R")), body="3 + 4",
    encode="json")
# Get status of all Celery tasks
GET(flower("/api/tasks"))
```

Listing 4.5: Using `floweRy`.

```
library(floweRy)
options(flowerURL=
    "http://localhost:5555")
# Run R command '3 + 4' in Celery
taskApply("tasks.R", "3 + 4")


# Get status of all Celery tasks
taskList()
```

cTRAP currently supports ranking similar CMap perturbation and predicting targeting drugs as background processes. The user can monitor the status of their background tasks in cTRAP. When the background processes finish, Celery output files are saved into the session folder. The data is then automatically loaded and the user is informed of such via a notification in cTRAP (Figure 4.8). If the user has closed the

session, this process will occur when the cTRAP session is loaded via its token.

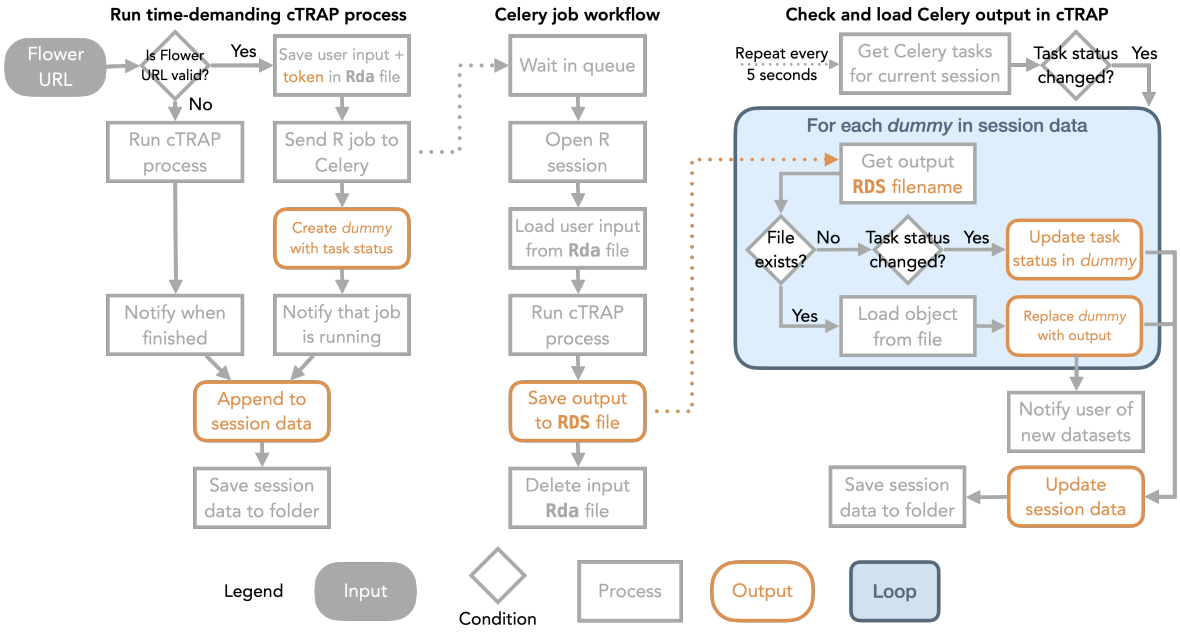

Figure 4.8: cTRAP process running in Celery. Time-demanding cTRAP processes can be run in the background using Celery/Flower. While running in Celery, the output of the cTRAP process is saved to the folder associated with the token of the user's session. When that specific session is active, all finished files are automatically loaded as part of the data session and the user is notified.

Why saving *dummy* objects as part of user session data?

4.4 Conclusion

It would also be interesting to send automatic emails to users when a job finishes or raises an error. This would require to set up an email address to which to send emails from.

For future cTRAP iterations, it would interesting to support CMap LINCS 2020, an expansion of the current CMap dataset that is described as a "3-fold expansion on the previous resource, and notable new subsets of data include CRISPSR knockout of >5k genes and hematopoietic and non-cancer cell models" (`clue.io/data/CMap2020#LINCS2020`).

Also, newever versions of drug sensitivity datasets: GDSC 8?

Chapter 5

CompBio app server

Since I started building psichomics, I wanted my work to be publicly available as an online web app, providing users the most up-to-date version at their fingerprints, without having to install, update and manage different versions of R, Bioconductor, psichomics and all their dependencies. Five years after the first Bioconductor release of psichomics in 2016, that vision finally came true.

One of our lab's ambitious goals is to develop interactive visual tools to assist in exploring biological data, either provided by users or available from big datasets. We want our tools to be used by anyone, no matter their computational background. To turn that dream into reality, I set up the CompBio app server, a Linux virtual machine running in iMM computing cluster that hosts psichomics, cTRAP and other Shiny apps from my lab colleagues. The server is accessible at `compbio.imm.medicina.ulisboa.pt` (Figure 5.1) and its code at `github.com/nuno-agostinho/compbio-app-server`.

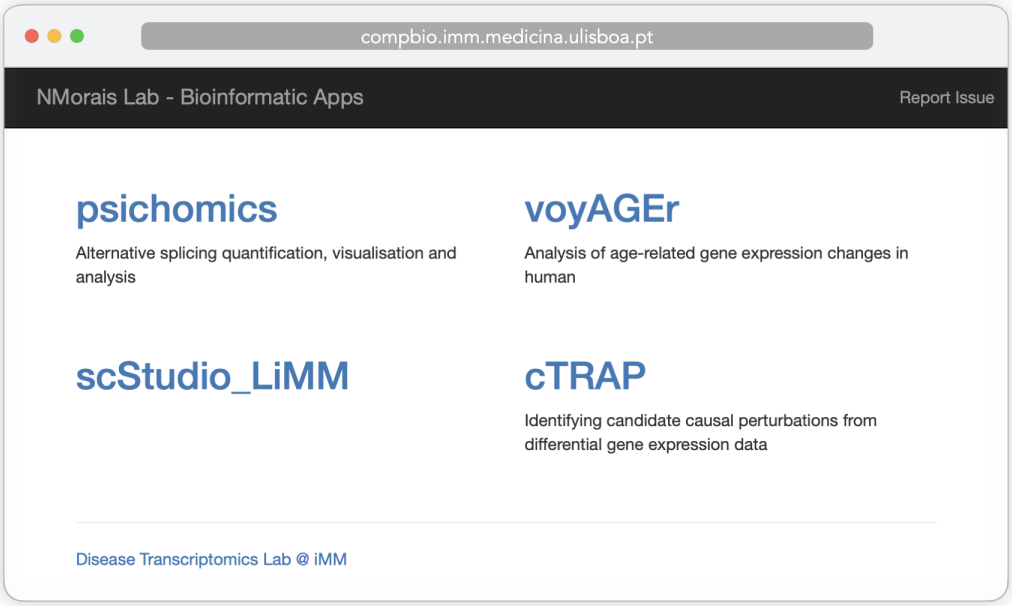

Figure 5.1: CompBio's homepage screenshot. List of hosted web apps (11 Nov 2021).

5.1 Background

Our lab uses the R statistical language to analyse clinical and molecular data from public sources and collaborators. In order to share data insights with our collaborators or even the whole scientific community, we have been increasingly creating exploratory dashboards using the Shiny R package [112]. While developing interactive Shiny web apps, it is natural to wonder: what is the best way to share them?

5.1.1 Desktop apps

Shiny apps are written in R, an interpreted programming language whose source code can run in multiple platforms [163, 112]. When run locally, the Shiny app starts running in the device itself (`localhost`) and is accessible via a web browser. Shiny apps can be part of an R package and be provided in CRAN or Bioconductor (such as in the case of psichomics and cTRAP). Nonetheless, this requires the user to install multiple programs in their computer: R, Shiny, the Shiny app, and all their dependencies. This can take up some time if the user does not have R and many of the required libraries installed. For instance, installing psichomics in a new system can take up to 1 hour. Moreover, it still requires opening an R session to start the visual interface, which may discourage technically-challenged users to try out psichomics.

One way to reduce the number of dependencies installed is by using Docker (`docker.com`), allowing to run isolated Linux virtual environments (containers) that already contain programs and all their dependencies set up. This approach simply requires end-users to install Docker and to download the desired Docker images online. Still, Docker is a program that needs administrator privileges for installation that (1) not all users may have and (2) may not feel comfortable to give to a software they would not otherwise install.

An alternative is Electron (`electronjs.org`), a software framework that allows to develop cross-platform graphical user interface apps using web technologies by combining a web browser rendering engine (Chromium, used in Chrome and other web browsers to convert HTML and CSS code into an interactive web page) and a JavaScript environment. The app itself runs the web app as if it were a usual desktop app. Some open-source projects like electricShine (`github.com/chasemc/electricShine`) and photon (`github.com/COVAIL/photon`) allow to convert Shiny apps to Electron apps, but they are still not fully developed and lack important features (like support for some operative systems). Regardless, compared to native apps, Electron apps are slower, have a significant overhead, take more space and consume more RAM, making Electron less attractive for intensive data-processing apps.

5.1.2 Web apps

Web apps are cross-platform, always up-to-date and can be accessed by any (modern) web browser, making access to such apps easier for end-users [105]. However, a constantly online web server needs to be running and share its computing resources (e.g. amount of RAM, storage and CPU threads) across multiple users. The resources allocated to a web server depend on the resources consumed per app, the number of simultaneous users and the data stored per user. The price of components and their maintenance is specially relevant if anticipating a large number of end-users.

Multiple web app hosting services support Shiny apps or Docker containers of Shiny apps, including Heroku (`heroku.com`) and `shinyapps.io`. Both of these app hosting services offer subscription plans depending on allocated system resources, including a free plan useful to run basic apps: Heroku's free plan offers 2 threads, 512 MB of RAM and 500 MB of storage per app[1], whereas `shinyapps.io`'s free plan allows for 5 apps with 25 computing hours per month using 1024 MB of RAM and 1 GB of storage per app[2]. Such services take care of deploying the web apps and we can select a different plan to scale up the required resources to run the apps, depending on their usage. They also allow to monitor app resource usage and understand how the apps are being used and if the resources employed are sufficient or not without much effort to the developer.

Besides third-party server hosting, Shiny apps can also be deployed in local web servers. This requires server maintenance and may be harder to scale resources because of higher up-front costs. The following programs allow to host Shiny apps in a local machine:

- **Shiny Server** (`rstudio.com/products/shiny/shiny-server`) is a bare-featured open-source program with only the essential features to host Shiny apps.
- **RStudio Connect** (`rstudio.com/products/connect`) is a paid program[3] with many more features than Shiny Server, including user authentication, Python-based app support and resource usage metrics.
- **ShinyProxy** (`shinyproxy.io`) is an open-source program to host Shiny apps in Docker containers with many of the features found in RStudio Connect, including user authentication, Python-based app support and resource usage metrics.

Given that we have sufficient computing resources at our lab's disposal, we decided to build an app server – a web server dedicated to deploy our web apps. We decided

[1] According to Heroku (`heroku.com/pricing` and `devcenter.heroku.com/articles/limits`) as of 24 November 2021. Unverified accounts (i.e. not associated with a valid credit card) are limited to 5 apps.

[2] According to official `shinyapps.io` documentation (`docs.rstudio.com/shinyapps.io/applications.html` and `shinyapps.io#pricing`) as of 24 November 2021.

[3] According to RStudio (`rstudio.com/pricing`), all RStudio commercial products are free for teaching purposes and 50% discounted for academic research from their regular bundle pricing starting at 22000$ per year as of 24 November 2021.

to use ShinyProxy as it has many of the advantages of using the proprietary RStudio Connect for free. Following this choice, we had to think how to properly develop the web server so it is easy to maintain, update and add new apps.

In this chapter, I describe CompBio, our app server built with Docker Compose, a program to simultaneously manage multiple interacting Docker containers to allow for R/Shiny and Python app deployment (ShinyProxy) over a reverse proxy (Nginx), background tasks (Celery, Redis and Flower), website analytics (Plausible, PostgreSQL and ClickHouse), resource monitoring (Prometheus and Graphana), and feature testing (RStudio Web, only used to develop features and R scripts). CompBio is currently running in a virtual machine in a Linux computing cluster and hosts Shiny apps from NMorais lab, including the tools previously mentioned in this document: psichomics and cTRAP. CompBio is so named because it powers **Comp**utational **Bio**logy apps.

5.2 Materials and methods

CompBio is built using Docker Compose to manage the Docker images of multiple services: ShinyProxy, Nginx, Celery, Redis, Flower, Plausible, PostgreSQL, ClickHouse, Prometheus, Grafana and RStudio Web (Table 5.1). RStudio Web is only available in the development profile. The services communicate between each other via a single network created by Docker (Figure 5.2).

Table 5.1: CompBio web services.

Role	Service	Docker image[a]
Web app deployment	ShinyProxy	`openanalytics/shinyproxy`
Reverse proxy	Nginx	`nginx`
Background tasks	Celery + cTRAP	Based on `nunoagostinho/ctrap`[b]
	Redis	`redis`
	Flower	`mher/flower`
Website analytics (i.e. track visitor metrics)	Plausible	`plausible/analytics`
	PostgreSQL	`postgres`
	ClickHouse	`yandex/clickhouse-server`
Register and monitor server resources	Prometheus	`prom/prometheus`
	Grafana	`grafana/grafana`
Run R sessions and test features	RStudio Web[c]	Based on `rocker/rstudio`

[a] Available in Docker Hub, unless stated otherwise. [b] Python and Celery are installed on top of cTRAP Docker image, allowing Celery to run cTRAP analyses: see file `celery/Dockerfile`. [c] Only available in the development profile.

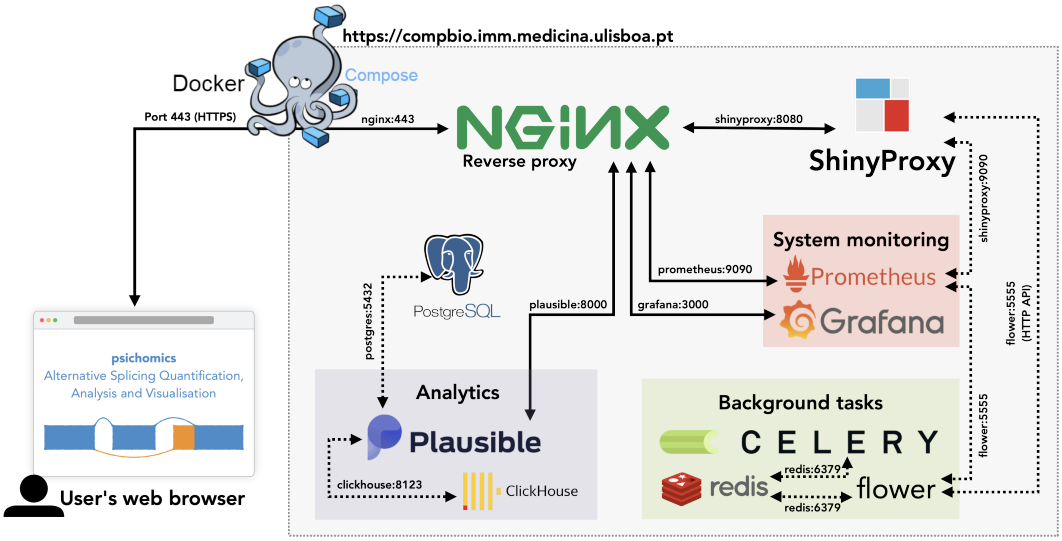

Figure 5.2: App server architecture is based on Docker Compose. All services are provided via Docker images and communicate with each other via a Docker-created network using the name of the service and a specific port (e.g. Nginx communicates with ShinyProxy via `shinyproxy:8080`). The groups (analytics, system monitoring and background tasks) are strictly conceptual.

5.3 Results

The CompBio project makes use of a two-tiered architecture as the user interface is displayed using the user's web browser to render the HTML, CSS and JavaScript code, whereas the application and database layers are all run in the app server.

The code to run the CompBio app server supports Linux[4] machines with only Docker and Docker Compose installed, thus making the setup easily portable across different Linux computers and requiring minimal user setup.

5.3.1 Design decisions

The functional requirements for the app server were the following:

- Add new and update existing Shiny apps
- Easy to maintain and update
- Allow to run background processes (support for cTRAP long-running tasks)
- Track simple visitor metrics (e.g. apps used, elapsed time, geographical location of users, etc.)
- Monitor system usage

Noted non-functional requirements (quality attributes) for the app server include:

- Modular and maintainable (system components should be easy to update and replace without affecting other components)
- Extendable (easy to test new system components and deploy new apps)

[4]CompBio may run in other operating systems. However, this is beyond the scope of this project.

- Scalable (support increasing/decreasing availability of system resources)
- Easy-to-use (easy to navigate and descriptive in case of errors)
- Open-source and free
- Portable

There are a lot of programs that can go into a web server. Experimenting different programs while managing their manifold dependencies to develop an healthy web server is like an intricate ballet where all finely-coordinated dancers interplay for an astounding performance. This can be as difficult as it sounds: a wrong move can affect the whole show. After all, each program/dependency has its own requirements and some may be a distress to (un)install. Moreover, when the server is online, errors may arise due to configuration changes (such as new app updates), requiring a fast rollback to minimise server downtime. A solution is to use self-contained and modular programs, such as in the case of Docker containers. But how to coordinate several Docker containers to beautifully perform the Swan Lake?

With Docker Compose, multiple applications are run isolated from others in their own Docker containers, allowing to easily update or replace them without affecting other system components. All services spawned in Docker Compose are Docker images, either pre-created (e.g. Docker images from Docker Hub) or built before starting up all services (in this case, a Dockerfile is required to create the Docker images). Docker Compose allows to quickly play and swap programs: the services present in the server were selected after trying out many other combinations of alternative apps. The modularity of Docker Compose allows to easily test new system components and update software versions.

5.3.2 Docker Compose

All the code required to run the app server is available at `github.com/nuno-agostinho/compbio-app-server`. A single file (`docker-compose.yml`) contains the main configuration of each application in the server, and extra configuration files are available in the local directory. For organisation purposes, the project is organised by folders named after each service, where each folder stores files (e.g. Dockerfile, configuration and data) associated with the respective application (Figure 5.3).

The command `docker-compose up -d --build` downloads Docker images mentioned in `docker-compose.yml`, builds Docker images from Dockerfiles (argument `--build`) and starts up the services in detached mode (argument `-d`).

Although data from Docker containers are only available temporary after the container is stopped, files can be preserved in Docker volumes to avoid data loss when restarting services. When starting the `docker-compose.yml` project, Docker volumes for specific directories (e.g. database data) are mounted.

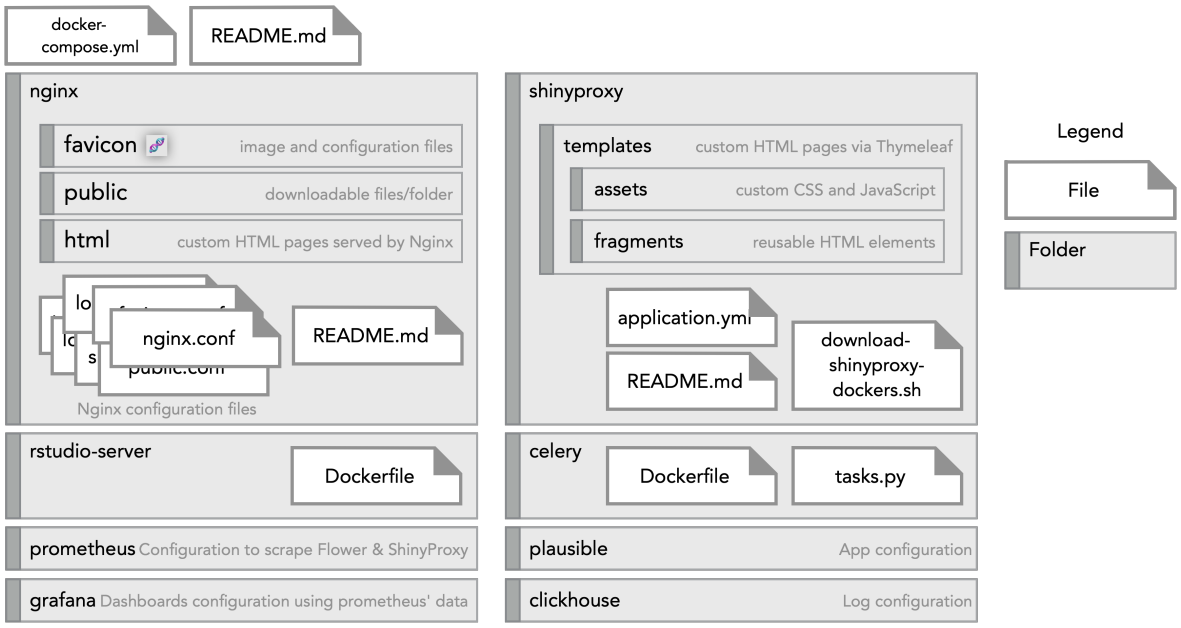

Figure 5.3: Visual representation of the file structure of the CompBio app server. Each folder contains files associated with a specific service. Folders `rstudio-server` and `celery` contain Dockerfiles for building custom Docker images of the respective services. Multiple `README.md` document the usage of the services and the app server itself. The root file `docker-compose.yml` contains the main configuration of each service.

Docker Compose has multiple commands to manage the services. For instance, it is possible to restart single services without affecting other programs, like `docker-compose restart shinyproxy`. This is useful after modifying the configuration of a service. However, changes to `docker-compose.yml` are only applied when shutting down all services with `docker-compose down`.

5.3.3 ShinyProxy

ShinyProxy is an open-source program that deploys R/Shiny and Python apps via Docker. When a user starts an app, ShinyProxy creates a new Docker container exclusively for that user. The containers are automatically terminated 30 minutes (by default) after the last user interaction.

Deploying new Shiny apps in the app server is as simple as adding the name of the Docker image in the ShinyProxy configuration file (`shinyproxy/application.yml`), downloading the respective Docker image and restarting ShinyProxy. Alternatively, running the script `download-shinyproxy-dockers.sh` automatically downloads or updates Docker images used in ShinyProxy.

ShinyProxy offers multiple built-in features, including:

- **App recovery:** when restarting ShinyProxy, ShinyProxy-initiated Docker containers continue running in the background. The apps will be unavailable while

ShinyProxy is not running, but will be attached to ShinyProxy once it is running again, allowing for quick server maintenance tasks[5].

- **User authentication:** authentication with multiple methods, including social login via GitHub, LinkedIn, Google, etc. However, user authentication requires all visitors to login before continuing. As we prefer users to be able to anonymously access our apps, this feature is currently disabled.
- **Multiple app instances:** users can open and manage multiple app instances simultaneously (not currently enabled in the app server)[6].

Progress bar when loading ShinyProxy apps

When ShinyProxy is loading an app, a spinning wheel is shown as a loading indicator. For apps that take more than 10 seconds to load (e.g. psichomics and cTRAP), the user may think the website is not working and close the window before the app is loaded. To avoid that, the spinning wheel was replaced with a progress bar to provide a time estimate for app loading (Figure 5.4), making wait times more tolerable [**?**, 164].

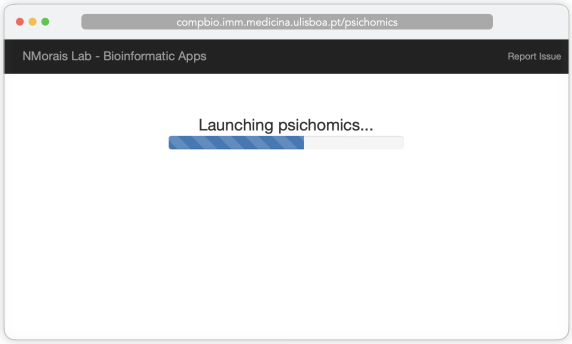

Figure 5.4: Progress bar displayed while psichomics loads (11 Nov 2021).

By default, the progress bar takes 5 seconds to fill (as sample Shiny apps take that much to launch in ShinyProxy), but the time is customisable for specific apps by editing the ShinyProxy configuration file (`shinyproxy/application.yml`) and adding a `template-properties.start-up` parameter to a specific app. For instance, psichomics takes 20 seconds to fully load the progress bar (i.e. `template-properties.start-up: 20s`), whereas cTRAP takes 15 seconds. When the app finishes loading, the progress bar is replaced by the app regardless of the progress displayed to the user. The accuracy of the progress bar does not have to be perfect to serve its purpose [**?**, 164].

Custom HTML pages

5.3.4 Nginx

Nginx is a reverse proxy, i.e. an intermediary that decides what is shown to the user depending on the URL visited (akin to those switchboard operators seen in the old movies). In CompBio, Nginx is responsible to fulfil user requests, to ensure HTTPS

[5] More information in `shinyproxy.io/documentation/app-recovery`.
[6] More information in `shinyproxy.io/documentation/ui/#using-multiple-instances-of-an-app`.

traffic is encrypted via SSL certificates, serve publicly available files and show a custom error page if ShinyProxy is not responding (e.g. temporarily down or overloaded).

Nginx is also used for ensuring encrypted HTTPS traffic via SSL certificates. SSL certificates are handed by the IT team at iMM and we only need to point Nginx to the correct location of those certificates. SSL certificates include three separate parts: the site certificate, intermediate certificates, and the private key.

A public folder is available via Nginx.

In case ShinyProxy is down, Nginx will serve a custom error page stating that the server is down probably because of ShinyProxy. This is informative enough to end-users that know they should wait to refresh the page in a moment and also to admins that will understand that ShinyProxy is temporarily down (this can happen because of multiple reasons, such as a restart of the service or overloading).

5.3.5 Background tasks

The main goal of our app server is to deploy Shiny apps, i.e. apps that were developed in R. The main disadvantage of using R is that it was designed for running single-thread processes. When running a Shiny app, that is specifically hard, as users cannot interact with the app while analysing data. This has been solved in multiple ways that extend R:

- by using promises/future
- by running R in the background

However, using any of those solutions implies that the Docker container needs to be active during the whole process, which can be especially egregious if the user session is big and is consuming many resources, disallowing other users to use those resources.

But how to make large tasks run in the background? In Lobo, iMM's computing cluster, we use a task manager for that function: SLURM. However, SLURM is too complex to install – I was not able to make a working prototype of SLURM with Docker Compose. So, what about a light-weight SLURM? That is what Celery is, a Python package that

5.3.6 Website analytics

Plausible is an open-source, privacy-focused web analytics tool that collects traffic metrics for multiple websites and provides them via an interactive dashboard. CompBio runs the self-hosted version of Plausible. All of Plausible metrics (e.g., visitor numbers, total page views and session duration) are anonymously aggregated without cookies, thus avoiding individual tracing.

Using the self-hosted version of Plausible guarantees that the user data tracked is done locally in the server. Plausible also protects user privacy by making their data

hard to individually trace and by complying with current privacy laws (GDPR, CCPA and PECR). This is in stark contrast with Google Analytics.

5.3.7 Resource monitoring

Prometheus monitors server resources. Graphana is used to visualise the metrics collected by Prometheus.

Celery

Many ShinyProxy metrics (including app usage time, app failures and user numbers) are collected with Prometheus and visualised using Grafana.

Nginx requires further configuration to be monitored. Currently, it is not monitored.

System

5.3.8 Server maintenance

CompBio is a web server that hosts Shiny applications and is publicly accessible by everyone online. This makes our server a target for potential security attacks. In order to mitigate such vulnerabilities, it is crucial to update user-facing programs (Docker, Docker Compose, Nginx and ShinyProxy), while components that are not directly available to end-users should be updated when possible. As updates may contain breaking changes that hamper website functionality, it is recommended to read change logs related to new software versions to pinpoint potential issues before updating.

Updates to Docker and Docker Compose need to be performed by an administrator using Linux's `apt-get` command[7]. Docker images of the server (including Nginx and ShinyProxy), on the other hand, require a user in the `docker` group to edit the versions of the Docker images used in `docker-compose.yml` and restart the Docker Compose project[8]. The advantage of using Docker Compose: if something goes wrong with the updated Docker images, simply revert `docker-compose.yml` to a previous working state and restart.

5.4 Conclusion

I think that the requirements for the app server were met with the current implementation.

CompBio currently runs in a virtual machine in Lobo, iMM computing cluster. All the hardware-related issues are taken care by the iMM IT team and they also support us with issues regarding SSL certificates, WebSocket connections and resource allocation. Moreover, I expect the server components to be easy to maintain and

[7] sudo apt-get update && sudo apt-get upgrade

[8] While inside the project folder: `docker-compose down && docker-compose up -d --build`

update. Components need to be manually updated but, in case of issues, it is easy to rollback to a stable, working version of the app server with previously used Docker images.

In the future, we can increase resources of our virtual machine if needed. In case we prefer to port the app server to a new machine, as the project was built on Docker Compose, relocating the app server is as easy as moving the project data to the new machine, installing Docker and Docker Compose, downloading required Docker images and starting the app server as previously indicated.

Chapter 6

Discussion

One thing I struggled during my path since starting building psichomics was the lack of guidelines on how to properly design and test interactive bioinformatic web apps. The resources are there, but too scattered.

Also, the lack of a systematic approach to app design is notable in multiple programs in the wild. My Master's was

6.1 PanAShé

In a collaborative lab effort, we are also developing a Nextflow pipeline to process raw RNA sequencing data from TCGA (Cancer Genome Atlas Research Network et al., 2013) and GTEx (The GTEx Consortium, 2013) in order to provide processed gene expression and alternative splicing data from samples from multiple normal and diseased tissues. The aims of this project extend those of recount2 (Collado-Torres, 2017) and include alternative splicing analysis, as well as a complementary dashboard to help users explore the data in these data sources. We are also considering integrating the data from this project in psichomics in lieu of the limited processed data from the public sources for TCGA and GTEx.

The Nextflow pipeline we are working on is based on Docker images for portability and reproducibility. This means that only Docker and Nextflow are required to be installed in the computer running the pipeline. We intend to write a peer-reviewed article regarding this project, as well as share our scripts and processed data with the scientific community as soon as possible.

6.2 Conclusion

Bibliography

[1] Gilbert W. Origin of life: The RNA world. Nature. 1986;319(6055):618–618. doi:10.1038/319618a0.

[2] Alberts B, Johnson A, Lewis J, Raff M, Roberts K, Walter P. Molecular Biology of the Cell. 5th ed. New York, NY: Garland Science; 2008.

[3] Sharp PA. On the origin of RNA splicing and introns. Cell. 1985;42(2):397–400. doi:10.1016/0092-8674(85)90092-3.

[4] Mulder GJ. Sur La Composition De Quelques Substances Animales. Bulletin des Sciences Physiques et Naturelles en Néerlande. 1838; p. 104–119.

[5] VICKERY HB. The origin of the word protein. The Yale journal of biology and medicine. 1950;22(5):387–393.

[6] Dahm R. Friedrich Miescher and the discovery of DNA. Developmental Biology. 2005;278(2):274–288. doi:https://doi.org/10.1016/j.ydbio.2004.11.028.

[7] Kossel A. Ueber eine neue Base aus dem Thierkörper. Berichte der Deutschen Chemischen Gesellschaft zu Berlin. 1885;18(79-81).

[8] Kossel A, Neumann A. Ueber das Thymin, ein Spaltungsproduct der Nucleïnsäure. Berichte der Deutschen Chemischen Gesellschaft. 1893;26(3):2753–2756.

[9] Kossel A, Neumann A. Darstellung und Spaltungsprodukte der Nucleïnsäure (Adenylsäure). Berichte der Deutschen Chemischen Gesellschaft zu Berlin. 1894;27:2215–2222.

[10] Ascoli A. Über ein neues Spaltungsprodukt des Hefenucleins. Hoppe-Seyler's Zeitschrift für physiologische Chemie. 1901;31:161–5.

[11] Sutton WS. On the Morphology of the Chromosome Group in Brachystola Magna. vol. 14. Biological Bulletin; 1902.

[12] Morgan TH, Sturtevant AH, Bridges CB, Muller HJ. The Mechanism of Mendelian Heredity. Nineteenth Century Collections Online (NCCO): Science, Technology, and Medicine: 1780-1925. H. Holt; 1915. Available from: `https://books.google.pt/books?id=GZEEAAAAYAAJ`.

[13] Muller HJ. Artificial Transmutation of the Gene. Science. 1927;66(1699):84–87. doi:10.1126/science.66.1699.84.

[14] Calabrese EJ. Muller's nobel prize research and peer review. Philosophy, ethics, and humanities in medicine : PEHM. 2018;13(1):15–15. doi:10.1186/s13010-018-0066-z.

[15] Haeckel E. Das protistenreich. Leipzig: Ernst Günther; 1878. Available from: `https://www.biodiversitylibrary.org/item/119851`.

[16] Whittaker RH. On the broad classification of organisms. Q Rev Biol. 1959;34:210–226. doi:10.1086/402733.

[17] Whittaker RH. New Concepts of Kingdoms of Organisms. Science. 1969;163(3863):150–160. doi:10.1126/science.163.3863.150.

[18] Levene PA, London ES. The Structure Of Thymonucleic Acid. Journal of Biological Chemistry. 1929;83(3):793–802. doi:https://doi.org/10.1016/S0021-9258(18)77108-1.

[19] Demerec M. What is a gene? Journal of Heredity. 1933;24:368–378.

[20] Beadle GW, Tatum EL. Genetic Control of Biochemical Reactions in Neurospora. Proceedings of the National Academy of Sciences of the United States of America. 1941;27(11):499–506. doi:10.1073/pnas.27.11.499.

[21] HERSHEY AD, CHASE M. Independent functions of viral protein and nucleic acid in growth of bacteriophage. The Journal of general physiology. 1952;36(1):39–56. doi:10.1085/jgp.36.1.39.

[22] WATSON JD, CRICK FHC. Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. Nature. 1953;171(4356):737–738. doi:10.1038/171737a0.

[23] Kornberg A, Lehman IR, Bessman MJ, Simms ES. Enzymic synthesis of deoxyribonucleic acid. Biochimica et Biophysica Acta. 1956;21(1):197–198. doi:https://doi.org/10.1016/0006-3002(56)90127-5.

[24] Palade GE. A Small Particulate Component Of The Cytoplasm. The Journal of Biophysical and Biochemical Cytology. 1955;1(1):59–68. doi:10.1083/jcb.1.1.59.

[25] Jacob F, Monod J. Genetic regulatory mechanisms in the synthesis of proteins. Journal of Molecular Biology. 1961;3(3):318–356. doi:https://doi.org/10.1016/S0022-2836(61)80072-7.

[26] Hoagland MB, Stephenson ML, Scott JF, Hecht LI, Zamecnik PC. A Soluble Ribonucleic Acid Intermediate In Protein Synthesis. Journal of Biological Chemistry. 1958;231(1):241–257. doi:https://doi.org/10.1016/S0021-9258(19)77302-5.

[27] Warner JR, Knopf PM, Rich A. A Multiple Ribosomal Structure In Protein Synthesis. Proceedings of the National Academy of Sciences. 1963;49(1):122–129. doi:10.1073/pnas.49.1.122.

[28] CRICK FH. On protein synthesis. Symp Soc Exp Biol. 1958;12:138–163.

[29] CRICK F. Central Dogma of Molecular Biology. Nature. 1970;227(5258):561–563. doi:10.1038/227561a0.

[30] Crick FHC, Barnett L, Brenner S, Watts-Tobin RJ. General Nature of the Genetic Code for Proteins. Nature. 1961;192(4809):1227–1232. doi:10.1038/1921227a0.

[31] Khorana HG, Büchi H, Ghosh H, Gupta N, Jacob TM, Kössel H, et al. Polynucleotide synthesis and the genetic code. Cold Spring Harb Symp Quant Biol. 1966;31:39–49. doi:10.1101/sqb.1966.031.01.010.

[32] Crick FHC. The origin of the genetic code. Journal of Molecular Biology. 1968;38(3):367–379. doi:https://doi.org/10.1016/0022-2836(68)90392-6.

[33] Hurwitz J, Bresler A, Diringer R. The enzymic incorporation of ribonucleotides into polyribonucleotides and the effect of DNA. Biochemical and Biophysical Research Communications. 1960;3(1):15–19. doi:https://doi.org/10.1016/0006-291X(60)90094-2.

[34] Weiss SB, Gladstone L. A MAMMALIAN SYSTEM FOR THE INCORPORATION OF CYTIDINE TRIPHOSPHATE INTO RIBONUCLEIC ACID1. Journal of the American Chemical Society. 1959;81(15):4118–4119. doi:10.1021/ja01524a087.

[35] Stevens A. Incorporation of the adenine ribonucleotide into RNA by cell fractions from E. coli B. Biochemical and Biophysical Research Communications. 1960;3(1):92–96. doi:https://doi.org/10.1016/0006-291X(60)90110-8.

[36] Grunberg-Manago M, Ortiz PJ, Ochoa S. Enzymic synthesis of polynucleotides. I. Polynucleotide phosphorylase of azotobacter vinelandii. Biochim Biophys Acta. 1956;20(1):269–285. doi:10.1016/0006-3002(56)90286-4.

[37] Brenner S, Jacob F, Meselson M. An Unstable Intermediate Carrying Information from Genes to Ribosomes for Protein Synthesis. Nature. 1961;190(4776):576–581. doi:10.1038/190576a0.

[38] Britten RJ, Davidson EH. Gene Regulation for Higher Cells: A Theory. Science. 1969;165(3891):349–357. doi:10.1126/science.165.3891.349.

[39] Darnell JE, Philipson L, Wall R, Adesnik M. Polyadenylic Acid Sequences: Role in Conversion of Nuclear RNA into Messenger RNA. Science. 1971;174(4008):507–510. doi:10.1126/science.174.4008.507.

[40] Edmonds M, Vaughan MHJ, Nakazato H. Polyadenylic acid sequences in the heterogeneous nuclear RNA and rapidly-labeled polyribosomal RNA of HeLa cells: possible evidence for a precursor relationship. Proc Natl Acad Sci U S A. 1971;68(6):1336–1340. doi:10.1073/pnas.68.6.1336.

[41] Perry RP, Kelley DE. Existence of methylated messenger RNA in mouse L cells. Cell. 1974;1(1):37–42. doi:https://doi.org/10.1016/0092-8674(74)90153-6.

[42] Rottman F, Shatkin AJ, Perry RP. Sequences containing methylated nucleotides at the 5′ termini of messenger RNAs: Possible implications for processing. Cell. 1974;3(3):197–199. doi:10.1016/0092-8674(74)90131-7.

[43] Berget SM, Moore C, Sharp PA. Spliced segments at the 5′ terminus of adenovirus 2 late mRNA. Proceedings of the National Academy of Sciences. 1977;74(8):3171–3175. doi:10.1073/pnas.74.8.3171.

[44] Chow LT, Gelinas RE, Broker TR, Roberts RJ. An amazing sequence arrangement at the 5′ ends of adenovirus 2 messenger RNA. Cell. 1977;12(1):1–8. doi:10.1016/0092-8674(77)90180-5.

[45] Brack C, Tonegawa S. Variable and constant parts of the immunoglobulin light chain gene of a mouse myeloma cell are 1250 nontranslated bases apart. Proc Natl Acad Sci U S A. 1977;74(12):5652–5656. doi:10.1073/pnas.74.12.5652.

[46] Early P, Rogers J, Davis M, Calame K, Bond M, Wall R, et al. Two mRNAs can be produced from a single immunoglobulin μ gene by alternative RNA processing pathways. Cell. 1980;20(2):313–319. doi:10.1016/0092-8674(80)90617-0.

[47] Gilbert W. Why genes in pieces? Nature. 1978;271(5645):501–501. doi:10.1038/271501a0.

[48] Chow LT, Broker TR. The spliced structures of adenovirus 2 fiber message and the other late mRNAs. Cell. 1978;15(2):497–510. doi:10.1016/0092-8674(78)90019-3.

[49] Nevins JR, Darnell JEJ. Steps in the processing of Ad2 mRNA: poly(A)+ nuclear sequences are conserved and poly(A) addition precedes splicing. Cell. 1978;15(4):1477–1493. doi:10.1016/0092-8674(78)90071-5.

[50] Schmucker D, Clemens JC, Shu H, Worby CA, Xiao J, Muda M, et al. Drosophila Dscam is an axon guidance receptor exhibiting extraordinary molecular diversity. Cell. 2000;101(6):671–684. doi:10.1016/s0092-8674(00)80878-8.

[51] Grabowski P, Seiler S, Sharp P. A multicomponent complex is involved in the splicing of messenger RNA precursors. Cell. 1985;42(1):345–353. doi:10.1016/s0092-8674(85)80130-6.

[52] Mount SM. A catalogue of splice junction sequences. Nucleic Acids Research. 1982;10(2):459–472. doi:10.1093/nar/10.2.459.

[53] Black DL, Chabot B, Steitz JA. U2 as well as U1 small nuclear ribonucleoproteins are involved in premessenger RNA splicing. Cell. 1985;42(3):737–750. doi:10.1016/0092-8674(85)90270-3.

[54] Ruskin B, Green MR. An RNA Processing Activity That Debranches RNA Lariats. Science. 1985;229(4709):135–140. doi:10.1126/science.2990042.

[55] Horowitz DS, Abelson J. Stages in the second reaction of pre-mRNA splicing: the final step is ATP independent. Genes Dev. 1993;7(2):320–329. doi:10.1101/gad.7.2.320.

[56] Arenas J, Hurwitz J. Purification of a RNA debranching activity from HeLa cells. Journal of Biological Chemistry. 1987;262(9):4274–4279. doi:https://doi.org/10.1016/S0021-9258(18)61343-2.

[57] Kruger K, Grabowski PJ, Zaug AJ, Sands J, Gottschling DE, Cech TR. Self-splicing RNA: autoexcision and autocyclization of the ribosomal RNA intervening sequence of Tetrahymena. Cell. 1982;31(1):147–157. doi:10.1016/0092-8674(82)90414-7.

[58] Altman S, Baer M, Guerrier-Takada C, Vioque A. Enzymatic cleavage of RNA by RNA. Trends in Biochemical Sciences. 1986;11(12):515–518. doi:10.1016/0968-0004(86)90086-1.

[59] Smith CWJ, Patton JG, Nadal-Ginard B. Alternative Splicing In The Control Of Gene Expression. Annual Review of Genetics. 1989;23(1):527–577. doi:10.1146/annurev.ge.23.120189.002523.

[60] Meyer A, Schloissnig S, Franchini P, Du K, Woltering JM, Irisarri I, et al. Giant lungfish genome elucidates the conquest of land by vertebrates. Nature. 2021;590(7845):284–289. doi:10.1038/s41586-021-03198-8.

[61] Montes M, Sanford BL, Comiskey DF, Chandler DS. RNA Splicing and Disease: Animal Models to Therapies. Trends in genetics : TIG. 2019;35(1):68–87. doi:10.1016/j.tig.2018.10.002.

[62] SANGER F, THOMPSON EO, KITAI R. The amide groups of insulin. The Biochemical journal. 1955;59(3):509–518. doi:10.1042/bj0590509.

[63] RYLE AP, SANGER F, SMITH LF, KITAI R. The disulphide bonds of insulin. The Biochemical journal. 1955;60(4):541–556. doi:10.1042/bj0600541.

[64] Harris JI, Sanger F, Naughton MA. Species differences in insulin. Archives of Biochemistry and Biophysics. 1956;65(1):427–438. doi:https://doi.org/10.1016/0003-9861(56)90203-X.

[65] Gauthier J, Vincent AT, Charette SJ, Derome N. A brief history of bioinformatics. Briefings in Bioinformatics. 2018;20(6):1981–1996. doi:10.1093/bib/bby063.

[66] EDMAN P. A method for the determination of amino acid sequence in peptides. Arch Biochem. 1949;22(3):475.

[67] Dayhoff MO, Ledley RS. Comprotein: A Computer Program to Aid Primary Protein Structure Determination. In: Proceedings of the December 4-6, 1962, Fall Joint Computer Conference. AFIPS '62 (Fall). New York, NY, USA: Association for Computing Machinery; 1962. p. 262–274. Available from: `https://doi.org/10.1145/1461518.1461546`.

[68] Dayhoff MO, Hersh RT, Chang MA, Sochard MR. Atlas of Protein Sequence and Structure. 1st ed. National Biomedical Research Foundation; 1965.

[69] Pauling L, Zuckerkandl E. Chemical Paleogenetics: Molecular "Restoration Studies" of Extinct Forms of Life. Acta Chem Scand. 1963;17:S9–S16. doi:10.3891/acta.chem.scand.17s-0009.

[70] Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology. 1970;48(3):443–453. doi:https://doi.org/10.1016/0022-2836(70)90057-4.

[71] Smith TF, Waterman MS. Identification of common molecular subsequences. Journal of Molecular Biology. 1981;147(1):195–197. doi:https://doi.org/10.1016/0022-2836(81)90087-5.

[72] Edman P, Begg G. A Protein Sequenator. European Journal of Biochemistry. 1967;1(1):80–91. doi:https://doi.org/10.1111/j.1432-1033.1967.tb00047.x.

[73] Sanger F, Nicklen S, Coulson AR. DNA sequencing with chain-terminating inhibitors. Proceedings of the National Academy of Sciences of the United States of America. 1977;74(12):5463–5467. doi:10.1073/pnas.74.12.5463.

[74] Maxam AM, Gilbert W. A new method for sequencing DNA. Proc Natl Acad Sci U S A. 1977;74(2):560–564. doi:10.1073/pnas.74.2.560.

[75] Hood LE, Hunkapiller MW, Smith LM. Automated DNA sequencing and analysis of the human genome. Genomics. 1987;1(3):201–212. doi:https://doi.org/10.1016/0888-7543(87)90046-2.

[76] Protein Data Bank. Crystallography: Protein Data Bank. Nature New Biology. 1971;233(42):223–223. doi:10.1038/newbio233223b0.

[77] Burks C, Fickett JW, Goad WB, Kanehisa M, Lewitter FI, Rindone WP, et al. The GenBank nucleic acid sequence database. Comput Appl Biosci. 1985;1(4):225–233.

[78] Higgins DG, Sharp PM. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. Gene. 1988;73(1):237–244. doi:https://doi.org/10.1016/0378-1119(88)90330-7.

[79] Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. Journal of Molecular Biology. 1990;215(3):403–410. doi:https://doi.org/10.1016/S0022-2836(05)80360-2.

[80] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. Nature. 2004;431(7011):931–945. doi:10.1038/nature03001.

[81] Nurk S, Koren S, Rhie A, Rautiainen M, Bzikadze AV, Mikheenko A, et al. The complete sequence of a human genome. bioRxiv. 2021;doi:10.1101/2021.05.26.445798.

[82] Yadav SP. The wholeness in suffix -omics, -omes, and the word om. J Biomol Tech. 2007;18(5):277.

[83] Kuska B. Beer, Bethesda, and Biology: How Genomics Came Into Being. JNCI: Journal of the National Cancer Institute. 1998;90(2):93–93. doi:10.1093/jnci/90.2.93.

[84] Piétu G, Mariage-Samson R, Fayein NA, Matingou C, Eveno E, Houlgatte R, et al. The Genexpress IMAGE knowledge base of the human brain transcriptome: a prototype integrated resource for functional and computational genomics. Genome Res. 1999;9(2):195–209.

[85] Conesa A, Madrigal P, Tarazona S, Gomez-Cabrero D, Cervera A, McPherson A, et al. A survey of best practices for RNA-seq data analysis. Genome biology. 2016;17:13–13. doi:10.1186/s13059-016-0881-8.

[86] Trapnell C, Hendrickson DG, Sauvageau M, Goff L, Rinn JL, Pachter L. Differential analysis of gene regulation at transcript resolution with RNA-seq. Nature Biotechnology. 2013;31(1):46–53. doi:10.1038/nbt.2450.

[87] Li Y, Rao X, Mattox WW, Amos CI, Liu B. RNA-Seq Analysis of Differential Splice Junction Usage and Intron Retentions by DEXSeq. PLOS ONE. 2015;10(9):1–14. doi:10.1371/journal.pone.0136653.

[88] Irimia M, Weatheritt RJ, Ellis JD, Parikshak NN, Gonatopoulos-Pournatzis T, Babor M, et al. A Highly Conserved Program of Neuronal Microexons Is Misregulated in Autistic Brains. Cell. 2014;159(7):1511–1523. doi:10.1016/j.cell.2014.11.035.

[89] Tapial J, Ha KCH, Sterne-Weiler T, Gohr A, Braunschweig U, Hermoso-Pulido A, et al. An atlas of alternative splicing profiles and functional associations reveals new regulatory programs and genes that simultaneously express multiple major isoforms. Genome Res. 2017;27(10):1759–1768. doi:10.1101/gr.220962.117.

[90] Shen S, Park JW, Lu Zx, Lin L, Henry MD, Wu YN, et al. rMATS: robust and flexible detection of differential alternative splicing from replicate RNA-Seq data. Proc Natl Acad Sci U S A. 2014;111(51):E5593–601. doi:10.1073/pnas.1419161111.

[91] Hu Y, Huang Y, Du Y, Orellana CF, Singh D, Johnson AR, et al. DiffSplice: the genome-wide detection of differential splicing events with RNA-seq. Nucleic Acids Research. 2012;41(2):e39–e39. doi:10.1093/nar/gks1026.

[92] Rockhold F, Bromley C, Wagner EK, Buyse M. Open science: The open clinical trials data journey. Clin Trials. 2019;16(5):539–546. doi:10.1177/1740774519865512.

[93] Ryan M, Wong WC, Brown R, Akbani R, Su X, Broom B, et al. TCGASpliceSeq a compendium of alternative mRNA splicing in cancer. Nucleic Acids Res. 2016;44(D1):D1018–22. doi:10.1093/nar/gkv1288.

[94] Lonsdale J, Thomas J, Salvatore M, Phillips R, Lo E, Shad S, et al. The Genotype-Tissue Expression (GTEx) project. Nature Genetics. 2013;45(6):580–585. doi:10.1038/ng.2653.

[95] Collado-Torres L, Nellore A, Kammers K, Ellis SE, Taub MA, Hansen KD, et al. Reproducible RNA-seq analysis using recount2. Nature biotechnology. 2017;35(4):319–321. doi:10.1038/nbt.3838.

[96] Wilks C, Zheng SC, Chen FY, Charles R, Solomon B, Ling JP, et al. recount3: summaries and queries for large-scale RNA-seq expression and splicing. Genome Biology. 2021;22(1):323. doi:10.1186/s13059-021-02533-6.

[97] Subramanian A, Narayan R, Corsello SM, Peck DD, Natoli TE, Lu X, et al. A Next Generation Connectivity Map: L1000 Platform and the First 1,000,000 Profiles. Cell. 2017;171(6):1437–1452.e17. doi:10.1016/j.cell.2017.10.049.

[98] Malta TM, Sokolov A, Gentles AJ, Burzykowski T, Poisson L, Weinstein JN, et al. Machine Learning Identifies Stemness Features Associated with Oncogenic Dedifferentiation. Cell. 2018;173(2):338–354. doi:10.1016/j.cell.2018.03.034.

[99] Méndez-Lucio O, Baillif B, Clevert DA, Rouquié D, Wichard J. De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. Nature Communications. 2020;11(1):10. doi:10.1038/s41467-019-13807-w.

[100] Le BL, Andreoletti G, Oskotsky T, Vallejo-Gracia A, Rosales R, Yu K, et al. Transcriptomics-based drug repositioning pipeline identifies therapeutic candidates for COVID-19. Scientific Reports. 2021;11(1):12310. doi:10.1038/s41598-021-91625-1.

[101] Shoemaker RH. The NCI60 human tumour cell line anticancer drug screen. Nature Reviews Cancer. 2006;6(10):813–823. doi:10.1038/nrc1951.

[102] Seashore-Ludlow B, Rees MG, Cheah JH, Cokol M, Price EV, Coletti ME, et al. Harnessing Connectivity in a Large-Scale Small-Molecule Sensitivity Dataset. Cancer Discovery. 2015;5(11):1210–1223. doi:10.1158/2159-8290.CD-15-0235.

[103] Yang W, Soares J, Greninger P, Edelman EJ, Lightfoot H, Forbes S, et al. Genomics of Drug Sensitivity in Cancer (GDSC): a resource for therapeutic biomarker discovery in cancer cells. Nucleic Acids Research. 2012;41(D1):D955–D961. doi:10.1093/nar/gks1111.

[104] Storer T. Bridging the Chasm: A Survey of Software Engineering Practice in Scientific Programming. ACM Comput Surv. 2017;50(4). doi:10.1145/3084225.

[105] Silva L, Jimenez R, Blomberg N, Luis Oliveira J. General guidelines for biomedical software development [version 2; peer review: 2 approved]. F1000Research. 2017;6(273). doi:10.12688/f1000research.10750.2.

[106] Dyba T, Dingsoyr T. What Do We Know about Agile Software Development? IEEE Software. 2009;26(5):6–9. doi:10.1109/MS.2009.145.

[107] Hewitt E. Semantic Software Design: A New Theory and Practical Guide for Modern Architects. O'Reilly Media; 2019. Available from: `https://books.google.pt/books?id=TtWxDwAAQBAJ`.

[108] Kanat-Alexander M. Code Simplicity: The Fundamentals of Software. O'Reilly Media; 2012.

[109] Ford N, Richards M, Sadalage P, Dehghani Z. Software Architecture: the Hard Parts: Modern Trade-Off Analyses for Distributed Architectures. O'Reilly Media, Incorporated; 2021. Available from: `https://books.google.pt/books?id=sWNozgEACAAJ`.

[110] Wickham H, Danenberg P, Csárdi G, Eugster M. roxygen2: In-Line Documentation for R; 2021. Available from: `https://CRAN.R-project.org/package=roxygen2`.

[111] Huber W, Carey VJ, Gentleman R, Anders S, Carlson M, Carvalho BS, et al. Orchestrating high-throughput genomic analysis with Bioconductor. Nature Methods. 2015;12(2):115–121. doi:10.1038/nmeth.3252.

[112] Chang W, Cheng J, Allaire J, Sievert C, Schloerke B, Xie Y, et al.. shiny: Web Application Framework for R; 2021. Available from: `https://CRAN.R-project.org/package=shiny`.

[113] Iadanza E, Gonnelli V, Satta F, Gherardelli M. Evidence-based medical equipment management: a convenient implementation. Medical & Biological Engineering & Computing. 2019;57(10):2215–2230. doi:10.1007/s11517-019-02021-x.

[114] Baghdadi A, Lama S, Singh R, Hoshyarmanesh H, Razmi M, Sutherland GR. A data-driven performance dashboard for surgical dissection. Scientific Reports. 2021;11(1):15013. doi:10.1038/s41598-021-94487-9.

[115] Tan A, Durbin M, Chung FR, Rubin AL, Cuthel AM, McQuilkin JA, et al. Design and implementation of a clinical decision support tool for primary palliative Care for Emergency Medicine (PRIM-ER). BMC medical informatics and decision making. 2020;20(1):13–13. doi:10.1186/s12911-020-1021-7.

[116] Tidwell J, Brewer C, Valencia A. Designing Interfaces: Patterns for Effective Interaction Design. O'Reilly Media; 2019. Available from: `https://books.google.pt/books?id=rQXFDwAAQBAJ`.

[117] Alves T, Natálio J, Henriques-Calado J, Gama S. Incorporating personality in user interface design: A review. Personality and Individual Differences. 2020;155:109709. doi:https://doi.org/10.1016/j.paid.2019.109709.

[118] Chang K, Creighton CJ, Davis C, Donehower L, Drummond J, Wheeler D, et al. The Cancer Genome Atlas Pan-Cancer analysis project. Nature Genetics. 2013;45(10):1113–1120. doi:10.1038/ng.2764.

[119] Saraiva-Agostinho N, Barbosa-Morais NL. psichomics: graphical application for alternative splicing quantification and analysis. Nucleic Acids Research. 2018;47(2):e7–e7. doi:10.1093/nar/gky888.

[120] Sebestyen E, Singh B, Minana B, Pages A, Mateo F, Pujana MA, et al. Large-scale analysis of genome and transcriptome alterations in multiple tumors unveils novel cancer-relevant splicing networks. Genome Res. 2016;26(6):732–744. doi:10.1101/gr.199935.115.

[121] Saraiva-Agostinho N, Barbosa-Morais NL. In: Kidder BL, editor. Interactive Alternative Splicing Analysis of Human Stem Cells Using psichomics. New York, NY: Springer US; 2020. p. 179–205. Available from: `https://doi.org/10.1007/978-1-0716-0301-7_10`.

[122] Wang ET, Sandberg R, Luo S, Khrebtukova I, Zhang L, Mayr C, et al. Alternative isoform regulation in human tissue transcriptomes. Nature. 2008;456(7221):470–476. doi:10.1038/nature07509.

[123] Tsai YS, Dominguez D, Gomez SM, Wang Z. Transcriptome-wide identification and study of cancer-specific splicing events across multiple tumors. Oncotarget. 2015;6(9):6825–6839. doi:10.18632/oncotarget.3145.

[124] Danan-Gotthold M, Golan-Gerstl R, Eisenberg E, Meir K, Karni R, Levanon EY. Identification of recurrent regulated alternative splicing events across human solid tumors. Nucleic Acids Res. 2015;43(10):5130–5144. doi:10.1093/nar/gkv210.

[125] Chhibber A, French CE, Yee SW, Gamazon ER, Theusch E, Qin X, et al. Transcriptomic variation of pharmacogenes in multiple human tissues and lymphoblastoid cell lines. Pharmacogenomics J. 2017;17(2):137–145. doi:10.1038/tpj.2015.93.

[126] Climente-González H, Porta-Pardo E, Godzik A, Eyras E. The Functional Impact of Alternative Splicing in Cancer. Cell Rep. 2017;20(9):2215–2226. doi:10.1016/j.celrep.2017.08.012.

[127] Anczuków O, Akerman M, Cléry A, Wu J, Shen C, Shirole NH, et al. SRSF1-Regulated Alternative Splicing in Breast Cancer. Mol Cell. 2015;60(1):105–117. doi:10.1016/j.molcel.2015.09.005.

[128] Emig D, Salomonis N, Baumbach J, Lengauer T, Conklin BR, Albrecht M. AltAnalyze and DomainGraph: analyzing and visualizing exon expression data. Nucleic Acids Res. 2010;38(Web Server issue):W755–62. doi:10.1093/nar/gkq405.

[129] Katz Y, Wang ET, Airoldi EM, Burge CB. Analysis and design of RNA sequencing experiments for identifying isoform regulation. Nat Methods. 2010;7(12):1009–1015. doi:10.1038/nmeth.1528.

[130] Ryan MC, Cleland J, Kim R, Wong WC, Weinstein JN. SpliceSeq: a resource for analysis and visualization of RNA-Seq data on alternative splicing and its functional impacts. Bioinformatics. 2012;28(18):2385–2387. doi:10.1093/bioinformatics/bts452.

[131] Alamancos GP, Pagès A, Trincado JL, Bellora N, Eyras E. Leveraging transcript quantification for fast computation of alternative splicing profiles. RNA. 2015;21(9):1521–1531. doi:10.1261/rna.051557.115.

[132] Sterne-Weiler T, Weatheritt RJ, Best AJ, Ha KCH, Blencowe BJ. Efficient and Accurate Quantitative Profiling of Alternative Splicing Patterns of Any Complexity on a Laptop. Mol Cell. 2018;72(1):187–200. doi:10.1016/j.molcel.2018.08.018.

[133] Christinat Y, Pawłowski R, Krek W. jSplice: a high-performance method for accurate prediction of alternative splicing events and its application to large-scale renal cancer transcriptome data. Bioinformatics. 2016;32(14):2111–2119. doi:10.1093/bioinformatics/btw145.

[134] Doose G, Bernhart SH, Wagener R, Hoffmann S. DIEGO: detection of differential alternative splicing using Aitchison's geometry. Bioinformatics. 2018;34(6):1066–1068. doi:10.1093/bioinformatics/btx690.

[135] Robinson MD, McCarthy DJ, Smyth GK. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics. 2010;26(1):139–140. doi:10.1093/bioinformatics/btp616.

[136] Karolchik D, Hinrichs AS, Furey TS, Roskin KM, Sugnet CW, Haussler D, et al. The UCSC Table Browser data retrieval tool. Nucleic Acids Res. 2004;32(Database issue):D493–6. doi:10.1093/nar/gkh103.

[137] Lawrence M, Gentleman R, Carey V. rtracklayer: an R package for interfacing with genome browsers. Bioinformatics. 2009;25(14):1841–1842. doi:10.1093/bioinformatics/btp328.

[138] Braunschweig U, Barbosa-Morais NL, Pan Q, Nachman EN, Alipanahi B, Gonatopoulos-Pournatzis T, et al. Widespread intron retention in mammals functionally tunes transcriptomes. Genome Res. 2014;24(11):1774–1786. doi:10.1101/gr.177790.114.

[139] Ringner M. What is principal component analysis? Nat Biotechnol. 2008;26(3):303–304. doi:10.1038/nbt0308-303.

[140] Hyvarinen A, Oja E. Independent component analysis: algorithms and applications. Neural Netw. 2000;13(4-5):411–430. doi:10.1016/s0893-6080(00)00026-5.

[141] Rich JT, Neely JG, Paniello RC, Voelker CCJ, Nussenbaum B, Wang EW. A practical guide to understanding Kaplan-Meier curves. Otolaryngol Head Neck Surg. 2010;143(3):331–336. doi:10.1016/j.otohns.2010.05.007.

[142] Spruance SL, Reid JE, Grace M, Samore M. Hazard ratio in clinical trials. Antimicrob Agents Chemother. 2004;48(8):2787–2792. doi:10.1128/AAC.48.8.2787-2792.2004.

[143] Therneau T, Grambsch P. Modeling Survival Data: Extending The Cox Model. vol. 48; 2000.

[144] Kakaradov B, Xiong HY, Lee LJ, Jojic N, Frey BJ. Challenges in estimating percent inclusion of alternatively spliced junctions from RNA-seq data. BMC Bioinformatics. 2012;13 Suppl 6:S11. doi:10.1186/1471-2105-13-S6-S11.

[145] Jia C, Hu Y, Liu Y, Li M. Mapping Splicing Quantitative Trait Loci in RNA-Seq. Cancer Inform. 2015;14(Suppl 1):45–53. doi:10.4137/CIN.S24832.

[146] Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, et al. limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res. 2015;43(7):e47. doi:10.1093/nar/gkv007.

[147] Yates A, Beal K, Keenan S, McLaren W, Pignatelli M, Ritchie GRS, et al. The Ensembl REST API: Ensembl Data for Any Language. Bioinformatics. 2015;31(1):143–145. doi:10.1093/bioinformatics/btu613.

[148] Wu CH, Apweiler R, Bairoch A, Natale DA, Barker WC, Boeckmann B, et al. The Universal Protein Resource (UniProt): an expanding universe of protein information. Nucleic Acids Res. 2006;34(Database issue):D187–91. doi:10.1093/nar/gkj161.

[149] Nightingale A, Antunes R, Alpi E, Bursteinas B, Gonzales L, Liu W, et al. The Proteins API: accessing key integrated protein and genome information. Nucleic Acids Res. 2017;45(W1):W539–W544. doi:10.1093/nar/gkx237.

[150] Roberts RJ. PubMed Central: The GenBank of the published literature. Proc Natl Acad Sci U S A. 2001;98(2):381–382. doi:10.1073/pnas.98.2.381.

[151] Cunningham F, Amode MR, Barrell D, Beal K, Billis K, Brent S, et al. Ensembl 2015. Nucleic Acids Res. 2015;43(Database issue):D662–9. doi:10.1093/nar/gku1010.

[152] Fishilevich S, Zimmerman S, Kohn A, Iny Stein T, Olender T, Kolker E, et al. Genic insights from integrated human proteomics in GeneCards. Database (Oxford). 2016;2016. doi:10.1093/database/baw030.

[153] Uhlen M, Fagerberg L, Hallstrom BM, Lindskog C, Oksvold P, Mardinoglu A, et al. Proteomics. Tissue-based map of the human proteome. Science. 2015;347(6220):1260419. doi:10.1126/science.1260419.

[154] Goldman M, Craft B, Swatloski T, Cline M, Morozova O, Diekhans M, et al. The UCSC Cancer Genomics Browser: update 2015. Nucleic Acids Res. 2015;43(Database issue):D812–7. doi:10.1093/nar/gku1073.

[155] Murray K, Müller S, Turlach BA. Fast and flexible methods for monotone polynomial fitting. Journal of Statistical Computation and Simulation. 2016;86(15):2946–2966. doi:10.1080/00949655.2016.1139582.

[156] Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, et al. STAR: ultrafast universal RNA-seq aligner. Bioinformatics. 2013;29(1):15–21. doi:10.1093/bioinformatics/bts635.

[157] Kelemen O, Convertini P, Zhang Z, Wen Y, Shen M, Falaleeva M, et al. Function of alternative splicing. Gene. 2013;514(1):1–30. doi:10.1016/j.gene.2012.07.083.

[158] Paronetto MP, Passacantilli I, Sette C. Alternative splicing and cell survival: from tissue homeostasis to disease. Cell Death Differ. 2016;23(12):1919–1929. doi:10.1038/cdd.2016.91.

[159] Oltean S, Bates DO. Hallmarks of alternative splicing in cancer. Oncogene. 2014;33(46):5311–5318. doi:10.1038/onc.2013.533.

[160] Gallego-Paez LM, Bordone MC, Leote AC, Saraiva-Agostinho N, Ascensão-Ferreira M, Barbosa-Morais NL. Alternative splicing: the pledge, the turn, and the prestige : The key role of alternative splicing in human biological systems. Hum Genet. 2017;136(9):1015–1042. doi:10.1007/s00439-017-1790-y.

[161] Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, et al. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. PNAS; Proceedings of the National Academy of Sciences. 2005;102(43):15545–15550.

[162] Wickham H, Hesselberth J, Salmon M. pkgdown: Make Static HTML Documentation for a Package; 2021. Available from: `https://CRAN.R-project.org/package=pkgdown`.

[163] R Core Team. R: A Language and Environment for Statistical Computing; 2021. Available from: `https://www.R-project.org/`.

[164] Yablonski J. Doherty Threshold. In: Laws of UX: Using Psychology to Design Better Products & Services. 1st ed. O'Reilly Media, Inc.; 2020.