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

Durante o meu doutoramento no Instituto de Medicina Molecular João Lobo Antunes (iMM), desenvolvi aplicações *web* na área da transcriptómica, estabelecendo-as como recursos gratuitos e de código aberto disponíveis para a comunidade científica. As aplicações foram construídas a partir da linguagem de programação R com elementos gráficos baseados em *Shiny*, um pacote de R para criar aplicações *web*, e podem ser encontradas no Bioconductor, um repositório de pacotes de R.

psichomics

A relevância das alterações de *splicing* alternativo em condições fisiológicas e de doença, assim como a viabilidade económica crescente da sequenciação de RNA, têm progressivamente conduzido a um maior interesse no estudo global de *splicing* alternativo [1, 2, 3, 4, 5] e promovido grandes consórcios a disponibilizar publicamente dados de *splicing* alternativo. Tais consórcios incluem o The Cancer Genome Atlas (TCGA), que cataloga dados clínicos e moleculares de múltiplos tumores humanos [6]; o projecto the Genotype-Tissue Expression (GTEx), que se foca em dados de múltiplos tecidos humanos normais [7]; e o projecto recount2, um recurso com dados pré-processados de mais de 2000 estudos oriundos do Sequence Read Archive (SRA) [8].

A utilização dos dados pré-processados provenientes destes consórcios garante que os utilizadores não necessitam de armazenar e processar os ficheiros de dados iniciais, requerendo apenas recursos computacionais modestos para analisar expressão génica e *splicing* alternativo. Também é importante notar que os dados humanos não processados podem ser de acesso limitado e requerer um processo formal por razões de privacidade. No entanto, nenhuma ferramenta existente permitia realizar uma análise diferencial de *splicing* alternativo ao nível do transcriptoma com base em dados pré-processados descarregados automaticamente destas fontes públicas e com agrupamento das amostras a partir dos seus metadados.

Assim, desenvolvi o psichomics durante a minha tese de mestrado, uma ferramenta para quantificação, análise e visualização de *splicing* alternativo a partir de dados pré-processados provenientes do TCGA [6], utilizando uma interface gráfica intuitiva ou a linha de comandos. O psichomics realiza redução de dimensionalidades (por exemplo,

através da análise de componentes principais), *splicing* e expressão diferencial e análise de sobrevivência com incorporação de características moleculares e clínicas subjacentes às amostras usadas.

Durante o meu doutoramento, o psichomics foi melhorado de forma a descarregar e processar dados de fontes adicionais – incluindo dados do GTEx [7], do recount2 [8] e provenientes do utilizador –, analisar expressão génica, quantificar *splicing* alternativo para 14 espécies diferentes e outras novidades.

Após a publicação do artigo relativo ao psichomics em 2018 [9], fomos convidados a escrever um capítulo metódico publicado no ano de 2020, em que descrevemos como usar o psichomics para analisar diferenças de *splicing* alternativo no contexto de células estaminais humanas [10]. Desde a sua primeira publicação, o psichomics tem sido usado em vários artigos científicos [11, 12, 13, 14]. Acreditamos que estas citações, juntamente com os comentários positivos que temos vindo a receber dos utilizadores, revelam que vários investigadores e médicos conseguiram usufruir do psichomics para os auxiliar na descoberta de factores de prognóstico e alvos terapêuticos associados a splicing, assim como no avanço do nosso conhecimento sobre como o *splicing* alternativo é regulado em contexto fisiológico e de doença.

cTRAP

O Connectivity Map (CMap) é um repositório público de assinaturas transcriptómicas para centenas de perturbações genéticas e farmacológicas em linhas celulares de cancro humanas [15]. Ao comparar perfis de expressão génica diferencial com os do CMap, podemos inferir as potenciais causas moleculares das diferenças observadas e compostos que possam promover ou reverter essas alterações.

O CMap and LINCS Unified Environment (`clue.io`) foi desenvolvido como uma colecção de ferramentas intuitivas para explorar os dados do CMap e integrá-los com dados provenientes pelo utilizador [15]. No entanto, o `clue.io` limita o máximo número de genes utilizados para comparar com o CMap, expressa os resultados através de um valor de significância fora do padrão, é difícil de automatizar para análises subsequentes e não permite utilizar recursos computacionais locais. Além disso, o `clue.io` não permite actualmente integrar dados de sensibilidade às drogas, que podem auxiliar na identificação de compostos que alvejam células específicas.

Assim, desenvolvemos o cTRAP para identificar potencias perturbações moleculares causais ao comparar resultados completos de expressão génica diferencial providenciados pelo utilizador com aqueles do CMap, assim como prever compostos que promovam ou revertam as diferenças observadas. O cTRAP também permite comparar com associações de expressão génica e sensibilidade às drogas derivadas do NCI-60 [16], do Cancer Therapeutics Response Portal (CTRP) [17] e do Genomics of Drug Sensitiv-

ity in Cancer (GDSC) [18], para identificar compostos que podem alvejar os fenótipos associados com os perfis de expressão diferencial do utilizador e inclui uma análise similar à do Gene-Set Enrichment Analysis (GSEA) para identificar o enriquecimento de descriptores moleculares para compostos do NCI60 e do CMap. No cTRAP, a similaridade entre resultados de expressão diferencial é medida por *gene set enrichment* [15, 19] e valores de correlação.

O manucripto associado ao cTRAP (do qual eu sou co-primeiro autor e autor co-correspondente) encontra-se em preparação para submissão a uma revista científica *peer-reviewed* internacional. Esperamos que o cTRAP permita aos seus utilizadores identificar potenciais perturbações moleculares causais e compostos para melhor compreender mecanismos biológicos, tal como prever agentes terapêuticos relevantes.

CompBio: servidor de aplicações *web*

Ambos o psichomics e o cTRAP apresentam interfaces gráficas para auxiliar os utilizadores a explorar a maioria das funções dos respectivos programas de forma interactiva através de passos simples, devidamente explicados em tutoriais online. Tanto os tutoriais como as próprias ferramentas são constantemente actualizados conforme comentários e pareceres dos próprios utilizadores.

No entanto, apesar das suas interfaces gráficas fáceis de usar, os tradicionais canais de distribuição de pacotes de R, que usamos para distribuir o psichomics e o cTRAP (como o CRAN e o Bioconductor), podem ser dissuasores para quem não se sentir confortável a usar a linha de comandos do R, afastando potenciais utilizadores. Assim, procurei formas alternativas de disponibilizar as nossas aplicações, incluindo conversores de aplicações Shiny em programas normais Electron de computador (infelizmente, as soluções existentes não estão finalizadas), até encontrar o ShinyProxy, um programa de código aberto para hospedar aplicações R/Shiny e Python como aplicações *web* através de instâncias de Docker (*containers*).

Para correr o ShinyProxy, é necessário um servidor para correr o programa constantemente. Dessa forma, para distribuir as ferramentas de forma gratuita e mais acessível requerendo somente a utilização de navegadores de *Internet* modernos, criámos o servidor CompBio para disponibilizar o psichomics, o cTRAP e diversos outros pacotes de R como aplicações *web* – incluindo programas construídos pelos meus colegas de laboratório (Ageing Atlas, betAS e scStudio). O projecto CompBio está actualmente a correr numa máquina virtual Linux no *cluster* de computação do iMM.

O coração deste projecto é o Docker Compose, um programa que permite gerir múltiplos serviços em simultâneo que correm a partir de imagens Docker e que interagem entre si, incluindo o ShinyProxy, um proxy reverso para as configurações do servidor (Nginx) e programas para correr tarefas em segundo plano (Celery, Redis e

Flower), análise de dados do website (Plausible, PostgreSQL e ClickHouse), monitorização de recursos (Prometheus e Grafana) e desenvolvimento (RStudio Web, usado apenas para testar e desenvolver soluções directamente no servidor).

Todo o código deste projecto é de fácil configuração, pode ser facilmente portado para qualquer máquina e está publicamente disponível em `github.com/nuno-agostinho/compbio-app-server`, sendo a plataforma Docker o único requisito de instalação. Para adicionar novas aplicações ao projecto, basta editar as definições do ShinyProxy num ficheiro de texto e disponibilizar a respectiva imagem Docker localmente no servidor ou no repositório Docker Hub. A manutenção do projecto também é fácil, dado que basta actualizar a versão das imagens Docker que se quer usar no projecto e recomeçar quaisquer serviços afectados com o Docker Compose.

Quero aproveitar este momento para pedir uma pausa para sentar e relaxar, para deambular pelo nosso *website* em `compbio.imm.medicina.ulisboa.pt`, para acordar as aplicações *web* que ali pernoitam e para desfrutar da viagem. A página principal é uma galeria de trabalho do nosso laboratório, trabalho o qual tenho um tremendo orgulho em apoiar.

Palavras-chave: bioinformática, aplicações *web*, *splicing* alternativo, expressão génica, drogas específicas.

Summary

During my PhD at Instituto de Medicina Molecular João Lobo Antunes (iMM), I developed transcriptomic web apps as free, open-source resources and an app server to deploy those apps.

psichomics

Transcriptome-wide studies of alternative splicing have demonstrated its involvement in multiple cellular processes and linked its dysregulation to diverse pathologies. To exploit such publicly available data, we developed psichomics with an intuitive graphical interface to quantify, analyse and visualise alternative splicing using pre-processed data from The Cancer Genome Atlas, the Genotype-Tissue Expression project and the recount2 project, as well as user-provided data. psichomics assists the user integrating molecular and clinical sample-associated features to perform survival, dimensionality reduction, and differential alternative splicing and gene expression analyses. Since its publication in 2018, psichomics has been used to discover splicing-associated prognostic factors and therapeutic targets, along with studying alternative splicing regulation in physiological and disease contexts.

cTRAP

The Connectivity Map (CMap) hosts thousands of genetic and pharmacological perturbations. We developed cTRAP to tap into CMap and identify potentially causal molecular perturbations by comparing user-provided differential gene expression results with those from CMap, using correlation and gene set enrichment scores. cTRAP can also compare against gene expression/drug sensitivity associations derived from the NCI-60 cancer cell line panel, the Cancer Therapeutics Response Portal and the Genomics of Drug Sensitivity in Cancer project, to pinpoint compounds that may target the phenotypes associated with the user-provided differential expression profiles. We hope that cTRAP allows users to identify putative causal perturbations to better understand biological mechanisms, as well as to predict therapeutic targets.

CompBio app server

Both psichomics and cTRAP feature graphical interfaces to assist users explore most of their functionality. We set up the CompBio app server that uses Docker Compose, ShinyProxy, Nginx and other web services to deploy our lab's Shiny web apps, publicly available at `compbio.imm.medicina.ulisboa.pt`.

Keywords: bioinformatics, web apps, alternative splicing, gene expression, selective drugs.

Acknowledgements

My PhD has been an incredible 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.

TO BE CONTINUED.

List of Figures

List of Tables

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 put 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 stepped 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 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. Millions and millions 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 [20]. 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, we know that 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 are known as biomolecules and are crucial for the survival of living organisms [21].

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) [21]. 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 [21].

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 predates DNA and proteins in evolution [20, 21, 22]. After all, those RNA molecules could store genetic information like DNA and catalyse chemical reactions akin to enzymes, making RNA a prime candidate for life to take its first steps. As life evolved, these specific catalysing and storing 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 [20, 21].

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. In his publication almost two centuries apart from today, Mulder reported the chemical compounds of *les substances les plus essentielles du règne animal: la fibrine, l'albumine et la gélatine* [23]. To refer to these substances, Mulder named these words *protein* based on a Greek adjective that means of the *first rank or position*, reflecting the perceived importance of those molecules [23, 24].

Some decades later in 1869, Friedrich Miescher isolated 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 [25]. Albrecht Kossel (a former professor of Miescher) and colleagues described 5 organic compounds from nuclein: adenine, cytosine, guanine, thymine, and uracil [26, 27, 28, 29]. Nuclein was eventually renamed *nucleic acid*, but its importance was not recognised at the time [25].

In the first decades of the 20th century, the scientific consensus was that proteins carried genetic information, but Boveri and Sutton theorised otherwise [25, 30]:

> 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. ([30])

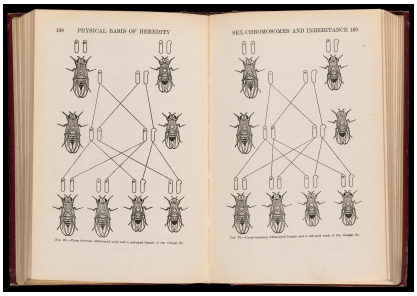


Figure 1.1: Thomas Morgan's fruit fly experiments.

The Boveri-Sutton chromosome theory of genetic inheritance followed Mendel's controversial [] work from 1865 [30] and was later supported by fruit fly experiments from an initially skeptical [] Thomas Morgan [31]. 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* [31]. 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 [32] [1].

[1] Muller's 1927 Science paper [32] that earned him the Nobel prize was not peer-reviewed, cited no references and lacked the methods section [33]. This may have happened not only because Muller 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 [33]. It would take until 1930 for Muller to publish results addressing Altenburg's criticism, although Muller was aware of the issues before 1927 [33].

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) [37], 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. ([38])

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 [39]. Together with Joshua Lederberg, Tatum demonstrated in 1946 that bacteria can exchange genetic material in a process called genetic recombination [40].

In 1952, Alfred Hershey and Martha Chase demonstrated that during viral infection by bacteriophage T2, its DNA, but not any viral protein, enters inside the bacteria [41]. The viral DNA is enough to produce the DNA molecules found in progeny virus particles. Amid the contemporary belief that proteins were the carriers of hereditary information, the Hershey-Chase experiment complemented previous publications suggesting that that role belonged to DNA [41].

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 [42]. DNA is composed by two phosphate-sugar chains linked together via hydrogen bonds by pairs of nucleotides: adenine pairs with thymine and cytosine with guanine. Crick and Watson also proposed that this strand complementarity could be important for DNA replication [42]. Afterwards,

[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.) [34]. In 1959, Robert Whitakker suggested a fungi kingdom amid three others [35], a proposal that later blossomed into his popular five-kingdom classification system published in 1969 [36]. In his 1969 article, Whittaker explains why fungi should not be considered plants to his fellow peers.

Arthur Kornberg observed the proposed nucleotide pairing in DNA synthesised by an enzyme that replicates DNA using one of its strands as a template: the DNA polymerase [43].

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 [44, 45]. 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 [46, 45]. Multiple ribosomes were found to bind to a single RNA molecule (polysomes), allowing for parallelised protein synthesis [47].

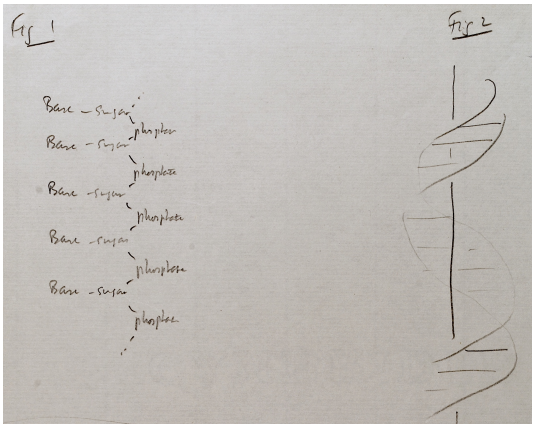


Figure 1.2: Figure drafts for a manuscript on DNA structure from Francis Crick and James Watson.

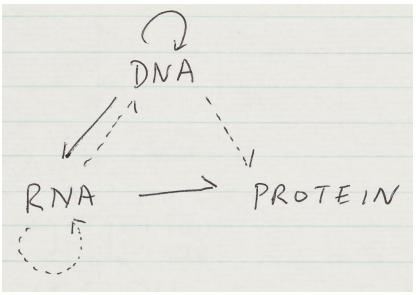


Figure 1.3: Central dogma of molecular biology as drafted by Francis Crick.

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*[48, 49]. 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 [48, 50] and that those amino acids would be responsible for the protein's three-dimensional structure – and consequently, its functionality [48]. 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 [51, 52]. 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 [53, 54, 55] [3].

François Jacob and Jacques Monod speculated in 1961 that ribosomal protein synthesis required an intermediate molecule with the template message to convert from

[3] In 1955, Severo Ochoa and Marianne Grunberg-Manago discovered the polynucleotide phosphorylase (PNPase) enzyme that they thought to synthesise RNA polymers from DNA [56]. Ochoa was erroneously awarded a Nobel prize in 1959 for discovering the biological mechanism of RNA synthesis [].

DNA to protein and that would act as the *messenger* [45, 57]. Unlike many of their 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 [45]. Based on contemporary experiments, Jacob and Monod proposed unstable RNA molecules as relevant candidates and named them messenger RNA (mRNA) [45, 57]. 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 [45].

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. ([58])

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 from another gene [58].

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* [59]. *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 [59, 60]. 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 [59, 60]. Later in 1974, the addition of a 5′-methylated cap was found in hnRNA and cytoplasmic mRNA and was proposed as an eukaryotic post-transcriptional RNA modification that protects the 5′-end of RNAs from degradation enzymes [61, 62].

1.2.1 Alternative splicing

First reported in mammalian cells infected with human adenovirus 2 [63, 64] and later observed in endogenous mammalian and eukaryotic genes [65, 66], mRNA-DNA hy-

bridisation 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 or after transcription of the precursor mRNA (pre-mRNA), non-coding sequences (introns) are excised from the transcript, remaining only the expressed segments (exons) in a process called *splicing* [63, 64, 67]. Moreover, multiple different transcripts may be produced from the same primary transcript by *alternative splicing* of segments of their sequence, thus promoting transcriptome diversity [63, 64, 68, 69, 70].

In 1985, an RNA-protein complex composed by U1, U2, U4, U5 and U6 small nuclear ribonucleoproteins (snRNPs) was reported central for RNA splicing: the spliceosome [71]. 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) [72, 73]. 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 [71, 74, 75]. The intron lariat is debranched (i.e. converted to a linear form) before its degradation [74, 76].

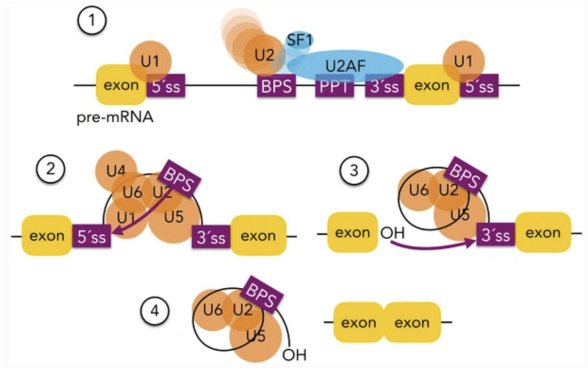


Figure 1.4: Spliceosome assembly and splicing reactions. (1) U1 snRNP binds to the 5′ splice site (5′ss), whereas the splicing factor 1 (SF1) and U2AF proteins bind to the branch point site (BPS), the polypyrimidine tract (PPT), and 3′ splice site (3′ss). The interaction between U1 and U2 snRNPs results 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 out. During the 1980s, Thomas Cech identified rRNAs that underwent self-splicing by breaking and forming covalent bonds with no associated proteins [77], 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 [78]. 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, one of the pillars that originated the RNA world hypothesis [77, 20]. 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 [22]. Primary transcripts from yeast can be spliced with the mammalian splicing machinery [22].

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 [79]. 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, promoting or repressing the inclusion of alternative sequences [79]. 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 [79].

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) [80, **?**]. 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 [80, **?**]. Notably, the fruit fly has 38 000 alternative transcripts generated from a single gene (Dscam) [70]. 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 [81, 82]. For instance, changes in splicing factors and subsequent perturbations to splicing can affect multiple hallmarks of cancer [83]. Therefore, multiple potential splicing-targeting therapies have been developed based on antisense oligonucleotides, small molecules and novel techniques [82].

1.3 Bioinformatics

Bioinformatics is a multidisciplinary field based on the usage and development of computer programs to analyse large-scale biological data [84]. The first bioinformatic analyses were performed on proteins. Following Sanger's work in 1959 on identifying

the amino-acid composition of the protein insulin in multiple species, many proteins started being sequenced [85, 86, 87, 84]. 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. ([48])

For that to come to fruition, there was a need to sequence much more proteins from different species. 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. Afterwards, the full protein sequence is reconstructed based on its overlapping fragments, a long and tedious process performed by hand [84, 88]. All these limitations meant that only 6 different proteins were fully sequenced by 1962 [89]. To overcome those difficulties in the reconstruction step, Margaret Dayhoff and Robert Ledley developed COMPROTEIN, the first bioinformatics program [84, 89]. COMPROTEIN allows to compare a high number of small peptide fragments and suggests possible full protein sequences [89]. 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*[90].

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 [91]. 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 [84]. Since 1970, computer-assisted phylogenetics started being a reality with the introduction of the Needleman-Wunsch algorithm [92] and variants, such as the Smith-Waterman algorithm [93]. These programs computationally measure the distance between two sequences by pair-wise comparison of amino acids between protein sequences [92, 93].

Years after automatic protein sequencing machines being available based on Edman degradation – *protein sequenators* as first called in 1967 [94] –, DNA sequencing methods were presented based on electrophoresis: the enzymatic Sanger method (also known as dideoxy method) [95] and the chemical degradation method Maxam-Gilbert sequencing [96]. These tedious and slow processes required manual intervention *at both the experimental and interpretative levels* [97].

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 [97]. 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 [97].

Later, the advent of Next-Generation Sequencers (NGS) allowed the massive parallel sequencing of nucleic acids. These techniques use alternative methods to Sanger sequencing to efficiently sequence DNA or RNA fragments simultaneously.

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 [98] and GenBank [99]. 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 [100]. 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 [101].

In 1995, the first complete sequence of a free-living organism was published for the bacterium *H. influenzae* [102]. From 1996 to 2000, the whole genomes of multiple other organisms were sequenced, including for the yeast *S. cerevisiae* [103], the nematode *C. elegans* [104], the fruit fly *D. melanogaster* [105, 106], and the plant *A. thaliana* [107]. 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 [108, 109]. 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. ([108])

More recently, the Telomere-to-Telomere (T2T) consortium exploited current long-read sequencing (third-generation sequencing), along with other sequencing technologies, in order to fully unravel the gapless assembly of the human genome for use in biomedical research [109] [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 [110]. 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 [110, 111]. The genomics field is concerned with the study and cross-species comparison of genomes [111].

In the same vein, transcriptomics is the field that studies the transcriptome: the set of RNA transcripts[5], as first defined in 1996 [112]. Transcriptomics is usually performed based on data generated from high-throughput technologies that allow to simultaneously analyse the expression of multiple RNAs. Diverse technologies have been proposed for the large-scale study of transcripts since the 1990s, including:

- **Expressed Sequence Tags (EST)**: proposed in 1991 as a pilot experiment to focus on the expressed genes via the Human Genome project [113]. EST allows to identify random sequences of complementary DNA (cDNA), i.e. reverse transcribed mRNA sequences.

- **Serial Analysis of Gene Expression (SAGE)**: developed in 1995 for the *quantitative and simultaneous analysis of a large number of transcripts*. [114]

- **Microarrays**: first mentioned in a 1995 study as a method to measure simultaneously the expression of multiple genes in an high-density array with small wells via cDNA hybridisation [115]. According to the article: *The large and expanding database of complementary DNA (cDNA) sequences from many organisms presents the opportunity of defining these patterns at the level of the whole genome* [115]. The first genome-wide microarray study was later conducted in

[4] Although the most recent T2T pre-print manuscript from 2021 describes ongoing work for the missing chromosome Y [109], 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`.

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

yeast during 1997 [116], followed by the whole human transcriptome using infant human brains in 1999 [112].

- **Short-read RNA sequencing (RNA-seq)**: first mentioned in 2008 as a novel quantitative technique based on the Illumina platform to sequence cDNA fragments in a massively parallel method [117]. This method is followed by computational mapping of resulting short reads (spanning 50 base pairs at the time, currently along the lines of 150-200 base pairs) to a genome of reference, allowing to unravel the transcriptional regions of the yeast. More accurate and sensitive than microarray methods, RNA-seq can quantify more lowly-expressed transcripts than microarrays by avoiding cross-hybridisation issues. RNA-seq data also allows to accurately identify exon boundaries and therefore introns, crucial for alternative splicing analysis [117].

- **Long-read RNA-seq**: RNA-seq methods that generate reads over 1000 base pairs and up to 10000 base pairs, allowing to sequence full transcripts in some cases.

- **Direct RNA-seq**: direct sequencing of RNA molecules without modification or reverse transcription.

RNA-seq data analysis

Before transcriptomic analyses, transcripts are isolated by first disrupting cell membranes and neutralising RNA-degrading enzymes (RNases). As over 90% of the extracted RNA is ribosomal, depletion of rRNAs and/or enrichment of the desired species are required for proper analysis. Most datasets currently enrich for polyadenylated RNAs (i.e. isolating mostly mature mRNAs).

After extraction, transcripts are sequenced. RNA-seq is a standard practice to better understand what features (genes, transcripts, exons, etc.) were 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 cDNA 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 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, such as ComBat [118, 119].

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 [120].

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

[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.

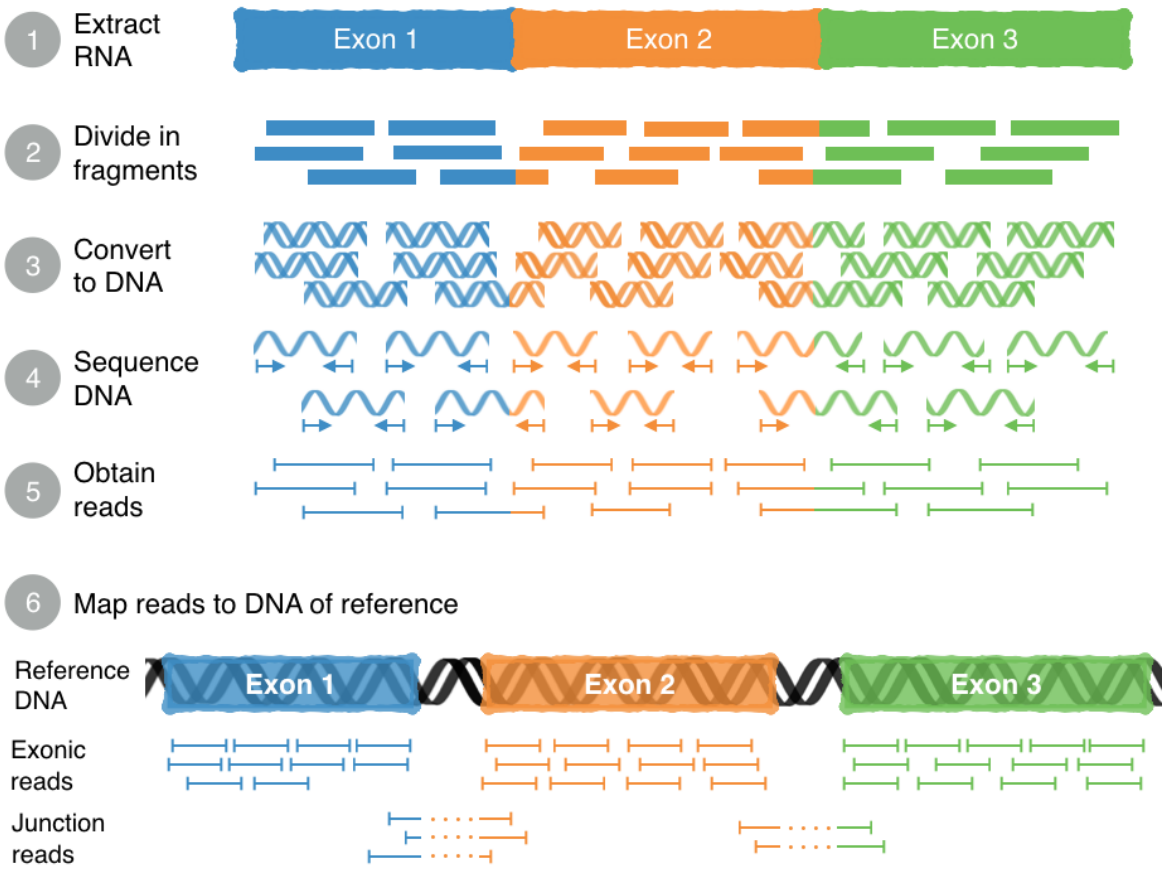


Figure 1.5: RNA sequencing and read mapping. RNA is first extracted from a sample (1) and divided into small fragments (2). These fragments are then converted to DNA (3) and sequenced, producing short text strings called reads (4). Finally, these reads are mapped to a DNA of reference (5) allowing to reconstruct the extracted mRNA and identify exon coordinates in the reference DNA.

some of those sequences fit in multiple places, such as reads from repetitive regions and so are distributed based on empiric evidence and/or randomness. Shorter reads are more prone to multi-mapping (less chances of sequencing unique regions) and their alignment may prove ambiguous. Such issues can be mitigated by using 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 [121].

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 [122]). However, isoform expression estimation based on short reads can be inaccurate. Other methods based on exon and junction reads specifically identify alterations in events of alternative splicing [121]: comparing the expression of different exons (differential exon usage, e.g. DEXseq [123]), estimating the percentage of alternative sequence inclusion (percent spliced-in, e.g. VAST-TOOLS [124, 125] and rMATS [126]) or based on graph theory to identify alternative splicing modules (e.g. DiffSplice [127]).

Transcriptomic studies like those performed using RNA-seq data allow to identify altered phenotypes 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, proteomics and other omics data [6, 128].

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 spending fortunes in grants to repeat experiments already performed by others, enabling new advances in personalised medicine via data sharing [129]. 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 and analysis [84], specially when used to train machine learning models. These analyses can be quite prohibitive for non-specialised researches interested in quick biological queries. To satisfy such needs, some projects provide open access to pre-processed data via download (e.g. sequence aligned and normalised data) and apps for data exploration:

- **The Cancer Genome Atlas (TCGA)** with molecular and clinical data for more than 30 cancer types (e.g. breast cancer and glioma) from more than 10 000 human samples [6]. Multiple web apps tap into the data from this behemoth, including TCGASpliceSeq [130] and Xena Browser [131].
- **The Genotype Tissue Expression (GTEx)** project, a repository of gene expression data for more than 40 tissues, totalling more than 15 000 human samples [7]. GTEx Portal (`gtexportal.org`) allows to inspect, for instance, the expression values of specific genes and isoforms across multiple tissues.
- The **recount** project has processed RNA-seq data from Sequencing Read Archive (SRA) [8, 132].

Even with the increasing economic feasibility of RNA-seq, cheaper technologies allow measuring gene expression for larger sample sizes, like in the case of L1000, an inexpensive assay platform where the profiling of 978 transcripts allows to estimate the expression of around 12000 genes via computational inference [15]. The development of L1000 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 [15]. CMap data has been used in multiple contexts, including to identify inhibitors of the stemness signature in multiple TCGA cancer types [133], to train machine learning models for designing molecules inducing desired transcriptomic changes [134] and to predict clinically-approved drugs for repurposing as SARS-CoV-2 antiviral agents [135].

To find effective novel treatments, it is important to understand how different cell respond to specific compounds. Multiple drug sensitivity datasets contain compound toxicity data across diverse cell lines, including the NCI-60 [16], the Cancer Therapeutics Response Portal (CTRP) [17] and the Genomics of Drug Sensitivity in Cancer (GDSC) [18]. 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 the number of datasets is getting larger every day, many obstacles render data inaccessible, including lack of standardised formats for storing molecular data, difficulty in retrieving human raw data, and inefficient communication between different platforms [129]. Another major hurdle of data dissemination is 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 pre-prints 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 Nextflow to run scalable computational pipelines that may employ Docker containers for reproducibility in different machines [136].

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 [121]. 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 [137, 138].

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 [137, 138, 139]. 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) [138, 140]. 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 [138, 141].

- **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 [138, 142].

  - Kanban-like boards allow to visualise and manage the project workflow, where features and bugs can be commented on and tracked [138, 140].

  - 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 [137, 138, 141, 142]. 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 [143] and Sphinx for Python [138, 140].

- **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) [138, 141, 142]. 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 [140].

- **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 [142].

- Software can be distributed as web-browser-based applications (web apps) or desktop apps. Containerisation (e.g. via Docker) also helps distributing complex software and its dependencies 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 [144] 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 [138]. 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) [138].
- 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 [138, 140, 142, 137].

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 [138, 140], as well as it facilitates analysis transparency and reproducibility [138]. Moreover, it has educational value, allowing others to learn from the developed code, implemented solutions and project organisation [140].

Open-source bioinformatic R packages are fully supported by Bioconductor and its community [144], facilitating the distribution of R packages. Bioconductor provides Docker images to allow easy access to their most popular R packages. Bioconductor also hosts packages that make use of the Shiny framework to create web apps from R [145].

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.

Specially used in business intelligence, dashboards are a common type of web app that allow interactive data analysis to explore key performance indicators (KPI). Dashboards have increasingly been the subject of health care research, from monitoring medical equipment performance [146] and assessing surgical performance [147] to predicting high-risk patients for primary palliative care [148]. Inspired by such dashboards, tools to explore and analyse bioinformatic data are also available, including to explore gene expression from user-provided data or from public datasets (e.g. [149, 150]).

To provide effective dashboards, good practices in data visualisation are paramount for intuitive communication of complex results [151]. 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 [151].

Another vital aspect of any app is its interface, no matter whether graphical or command-line based. User-centred interfaces assist users achieving their goals by considering the program's audience, including their intents 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 [138, 140, 151, 152]. Interfaces can be evaluated and improved by asking for and listening to user's feedback or by performing usability testing [151].

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

During my PhD, I developed transcriptomic web apps with graphical interfaces to be used by the scientific community, namely researchers with basic computational skills.

First, based on the work I developed during my MSc's thesis [], I continued working on psichomics [9, 10], an alternative splicing quantification, analysis and visualisation R package for pre-processed human data from TCGA [6]. During my PhD, psichomics was extended to support more data sources (including GTEx [7], recount2 [8] and user-provided data), analyse gene expression and support alternative splicing quantification for 14 different species, among other features.

Using the Connectivity Map (CMap), a public database containing millions of gene expression changes (i.e. perturbations) [15], our lab also developed cTRAP, an R package to identify candidate causal perturbations from differential gene expression data, as well as predict compounds that may promote or revert them. cTRAP also allows to list putative targeting drugs based on drug sensitivity datasets and includes a GSEA-based enrichment analysis of molecular descriptors for compounds from NCI60 and CMap.

Both psichomics and cTRAP feature web-based graphical interfaces to assist users interactively performing most of their functions following easy steps, properly detailed in online tutorials that are constantly updated according to user feedback. To make the tools more accessible and freely available via any modern web browser, we also developed an app server to deploy psichomics, cTRAP and multiple other R packages as web apps – including other programs built by my lab colleagues (Ageing Atlas, betAS and scStudio). I kindly invite you to pause, sit back and relax, visit our website at `https://compbio.imm.medicina.ulisboa.pt`, wander through the web apps there and enjoy the journey. The landing page is a gallery of work from our lab that I am deeply proud to support.

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 [145] that helped me to kick-start what would be later known as psichomics (Figure 3.1).

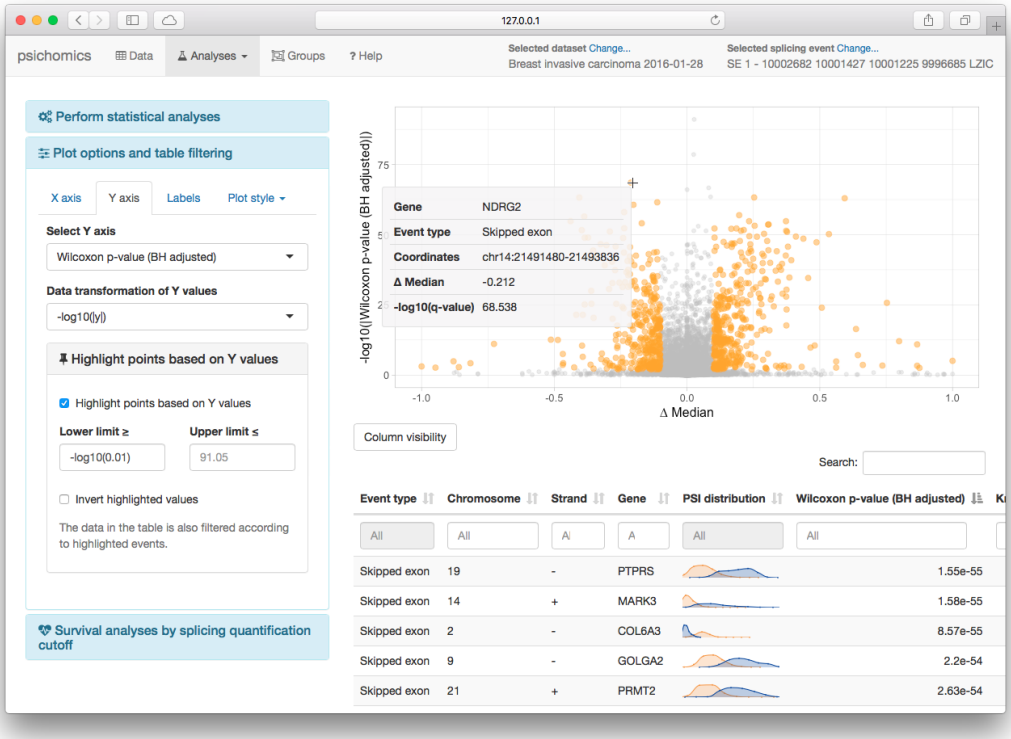


Figure 3.1: psichomics screenshot. TCGA breast cancer splicing analysis (28 Jan 2020).

psichomics was first made available in 2016 via Bioconductor to quantify, analyse and visualise human alternative splicing using TCGA data [6]. 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 [6], GTEx [7] and recount2 [8] (Table 3.1).

Table 3.1: Major milestones of psichomics.

Version	Release date	Main features
1.0	18 Oct 2016	Quantify and analyse alternative splicing from TCGA data[a]
1.0.8	18 Feb 2017	Analyse GTEx data
1.4	31 Oct 2017	Analyse gene expression from TCGA and GTEx data
1.4.2	19 Dec 2017	Support hg38 human genome assembly hg38
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 [9] is published online
1.8.2	27 Mar 2019	Add list of RNA-binding proteins [153]
	21 Jan 2020	psichomics' book chapter [10] 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	28 Oct 2021	Support alternative splicing annotation for 14 species[d]

[a] Bioconductor release. [b] First time supporting intron retention events (psichomics does not quantify intron retention). More information in subsection 3.2.4: **Alternative splicing quantification**. [c] First version available online. [d] Alternative splicing annotations for multiple species are available on-demand based on VAST-TOOLS annotation. Table 3.3 lists all supported species/assemblies. Custom alternative splicing annotations can also be imported.

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 [124, 125]. 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 to fully realise psichomics' potential – when it finally went online[1].

Following the publication of the first article describing psichomics in 2018 [9], we were invited to write a methodological book chapter published in 2020 [10]. Both publications were written by me (as the first and a co-corresponding author) and Dr. Nuno Morais. The content of those publications, along with some content from my MSc Thesis [], greatly inspired this chapter.

[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 RNA-seq, has progressively driven transcriptome-wide alternative splicing studies [1, 2, 3, 4, 5] 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 [6]; the Genotype-Tissue Expression (GTEx) project, that focuses on profiling normal human multi-tissue data [7]; and the recount2 project, a resource of processed RNA-seq data for over 2000 studies, mostly from the Sequence Read Archive (SRA) [8].

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 [2, 3, 4, 5, 154].

Several tools are currently available to quantify, analyse and visualise alternative splicing data. 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 [1]. Examples of such tools are AltAnalyze [155], MISO [156], SpliceSeq [157], VAST-TOOLS [124], rMATS [126], SUPPA [158] and Whippet [159]. 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 performs 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 [160] and DIEGO [161] 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 incorporation of molecular and clinical features.

psichomics is available online as a web app at `compbio.imm.medicina.ulisboa.pt/psichomics`, but can also be locally installed using Bioconductor (`bioconductor.org/packages/psichomics`) or Docker (`nunoagostinho/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).

The tool was designed as a modular R package to be easily modified and extended (Figure 3.3), including modules for automatic data retrieval from multiple sources, parsing and standardisation of alternative splicing event identifiers from different programs and a variety of data analysis methodologies.

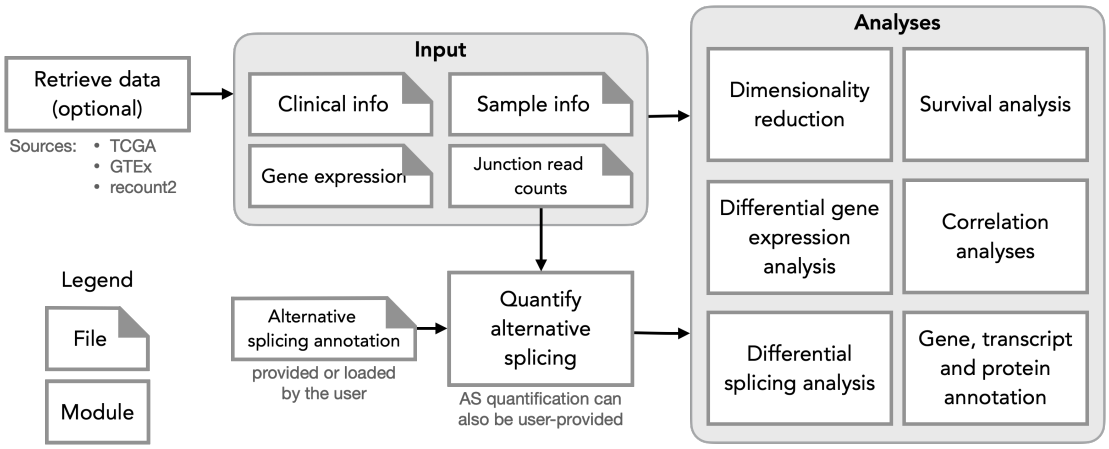


Figure 3.2: psichomics workflow. The user can provide their own input data or load data from TCGA, GTEx or recount2 to normalise gene expression data and quantify alternative splicing for downstream analyses.

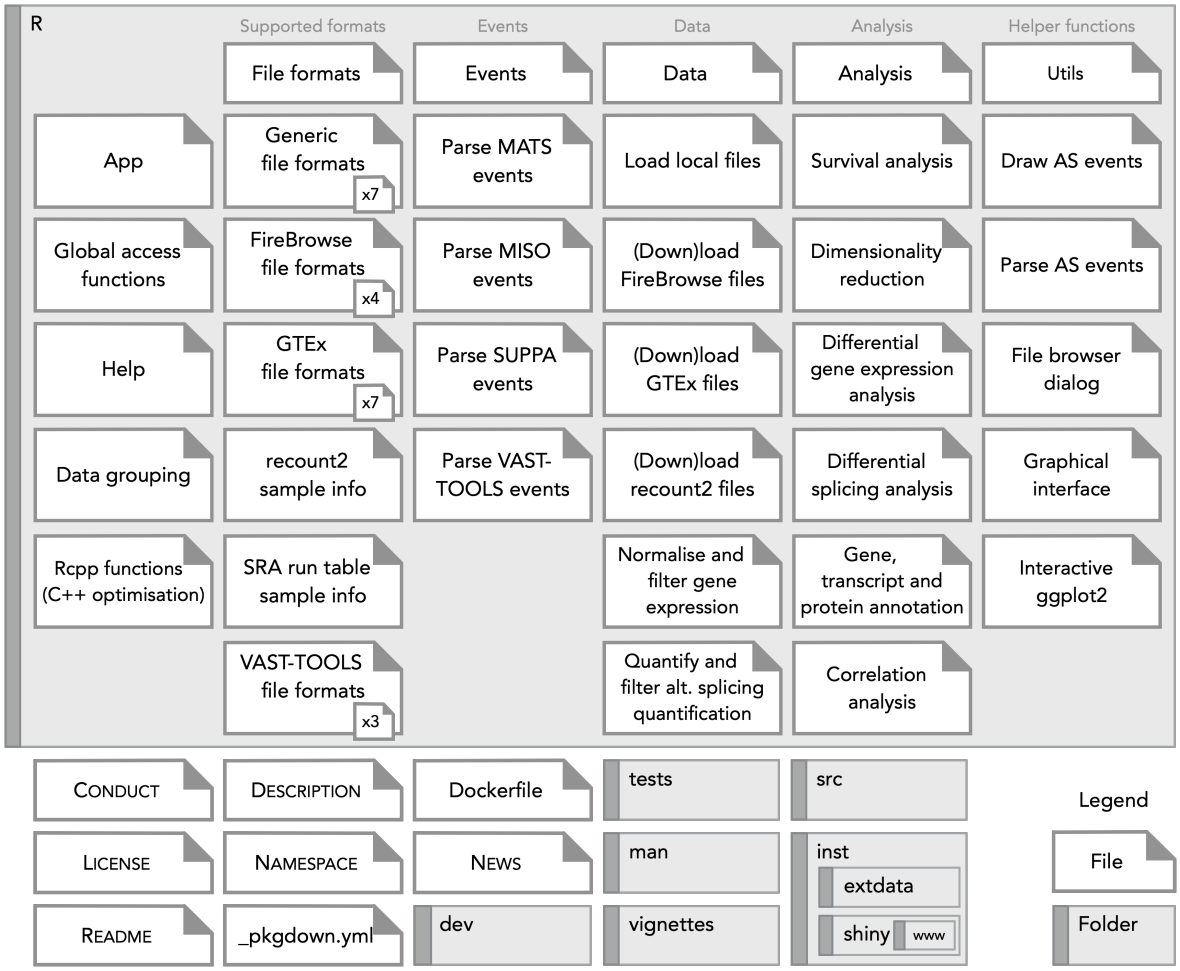


Figure 3.3: Visual representation of psichomics' file structure. psichomics is a modular program where, for instance, functions specific for different data sources and analyses can be found in different files. As usual for R packages, the `R` folder is the heart of the code and contains the main R scripts that define the logic and interface of the app. `dev` is a non-standard folder in R packages used to store supporting scripts (e.g. test workflows); its contents are not included when building the R package.

psichomics can load 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 [1], 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 [8]. Other SRA projects and user-provided files may also be loaded in appropriate formats (Table 3.2), allowing for subsequent alternative splicing analysis[2].

Table 3.2: Supported file formats in psichomics based on data 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.

[2] More information in the tutorial 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 `edgeR::calcNormFactors()` [162]. Afterwards, counts per million reads (CPM) can be computed and $\log_2$-transformed using `edgeR::cpm()`, as 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). To support multiple species, annotations were created based on VAST-TOOLS 23.06.20 using a function from psichomics (including for human, thus the redundancy with previous human annotations that were originated based on multiple sources). Custom annotation files are also supported[3].

Table 3.3: On-demand alternative splicing annotations for psichomics.

Species	Assembly	Source
Homo sapiens	hg19 + hg38	Multiple[a] VAST-TOOLS
Mus musculus	mm9 + mm10	VAST-TOOLS
Bos taurus	bosTau6	VAST-TOOLS
Gallus gallus	galGal3 + 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

[a] VAST-TOOLS, SUPPA, MISO and rMATS

The original hg19 annotation of human alternative splicing events was based on files used as input by MISO [156], VAST-TOOLS [124], rMATS [126] and SUPPA [158]. 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 [163] in

[3] More information in the appropriate tutorial at `nuno-agostinho.github.io/psichomics/articles/AS_events_preparation`.

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 using `rtracklayer::liftOver()` [164], 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 [1]. 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.

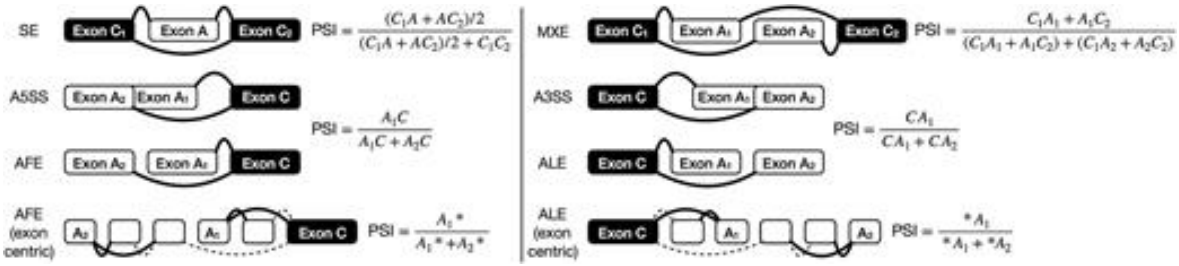


Figure 3.4: Alternative splicing quantification. Splice junctions required to quantify alternative splicing based on event type. C1A and AC2 represent read counts supporting junctions between a constitutive (C1 or C2, respectively) and an alternative (A) exon and therefore alternative exon A inclusion, 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 [165], 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 with `stats::fisher.test()`.

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 [166] and it is implemented through the singular value decomposition (SVD) algorithm provided by `stats::prcomp()`. The total contribution of each variable (splicing event or gene) towards data variance along selected principal components is measured based on `factoextra::fviz_contrib()` [].

Independent component analysis (ICA), used to decompose data into statistically independent components [167], can also be performed based on `fastICA::fastICA()`, preceded by data centering and/or scaling with `scale()`.

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 (based on `stats::kmeans()`), partitioning around medoids (PAM, `cluster::pam()`) or clustering large applications (CLARA, `cluster::clara()`) methods, with the latter being optimised for large datasets and thus the default recommendation.

3.2.7 Survival analysis

Kaplan-Meier estimators (and illustrating curves) [168] and proportional hazard (PH) models [169] 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` [170].

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 [171, 172], 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 `car::leveneTest.default()` [].

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 `stats::p.adjust()`. By default, multiple testing correction is performed using the Benjamini-Hochberg method.

Although the aforementioned statistical tests are also available to analyse the ex-

pression of single genes, genome-wide differential gene expression analysis is implemented based on gene-wise linear model fitting (`limma::lmFit()` [173]) 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 (`limma::eBayes()`) 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 analysing 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 `stats::cor.test()`, 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 [153], but the user can also define their own group of genes of interest for the test.

3.2.10 Feature annotation and literature support

The representational state transfer (REST) web services provided by Ensembl [174], UniProt [175], the Proteins API [176] and PubMed [177] 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 [178], GeneCards [179], the Human Protein Atlas [180], the UCSC Genome Browser [181], UniProt [175] and VAST-DB [125].

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 `MonoPoly::monpol()` [182]. The best polynomial fits (Figure 3.15) 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 [125] were aligned with splice-aware STAR [183] against the respective transcript-annotated genomes: UCSC hg19 genome assembly and GENCODE v19 annotation for human, UCSC mm10 genome assembly 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 `psichomics::quantifySplicing()` with default settings) were compared with the respective RT-PCR- and VAST-TOOLS-derived PSI values, available from VastDB [125].

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 [130] 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' web app is available at `compbio.imm.medicina.ulisboa.pt/psichomics`. Alternatively, users can install psichomics in their own computers, allowing them to use local computing resources. 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. To start the graphical interface in the local version, load the psichomics package in R via `library(psichomics)` and run `psichomics()`. The user's default web browser will be launched with a local version of the psichomics web app.

3.3.1 Case study

Several splicing factors have been reported to be involved in pluripotency, including SRSF3, MBNL1/2, RBFOX2, and U2AF1 [184, 185, 186, 187]. For instance, MBNL1/2 regulates the mutually exclusive inclusion of two FOXP1 exons, inducing a switch from its pluripotency-associated FOXP1-ES protein isoform, that promotes the expression of OCT4, NANOG, and other key pluripotency transcription factors, to the canonical differentiation-inducing FOXP1 isoform [188].

The early stage of somatic cell reprogramming, characterised by acquisition of pluripotency features, is related with mesenchymal-to-epithelial transition, a crucial development-related process affecting cell polarity and adhesion that is mediated by the aforementioned splicing regulators [184, 189]. Consistently, the alternative splicing modulation of epithelial-to-mesenchymal transition is linked with both cancer progression and metastasisation and with the generation of cancer stem cells, characterised by enhanced self-renewal, proliferation, and other stemness properties [184, 189, 190].

Using the graphical interface of psichomics, we analysed SRA project SRP063867 [191] containing genetically (i.e., isogenic) and not genetically (nonisogenic) matched human induced-pluripotent stem cells (iPSC), embryonic stem cells (ESC), and fibroblasts to compare changes in alternative splicing between isogenic stem cells and isogenic fibroblasts. The code to run this analysis is publicly available at `github.com/nuno-agostinho/stem-cell-analysis-in-psichomics`.

Data loading

We used psichomics to downloaded preprocessed RNA-seq data for SRP063867 via recount2 [8], including sample annotation, raw gene expression and exon–exon junction read counts. psichomics automatically downloads the data, loads the workspace and displays information per dataset (Figure 3.5).

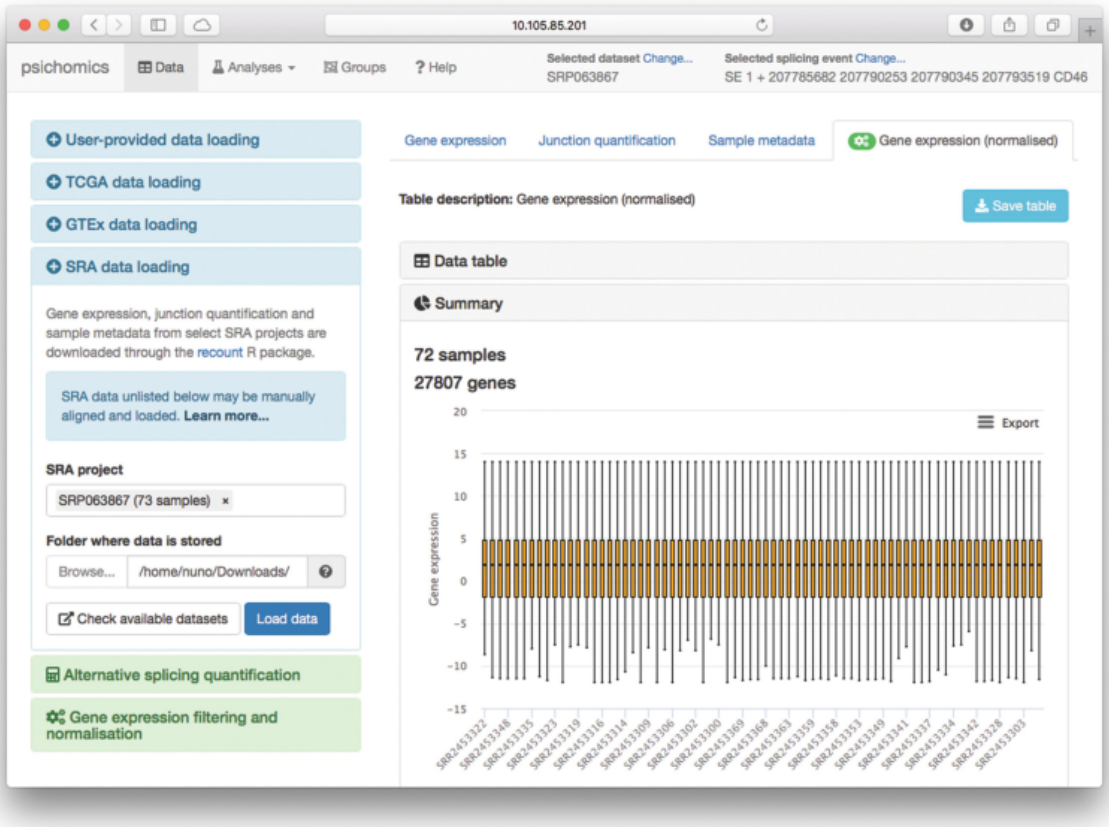


Figure 3.5: Summary information on datasets. psichomics presents information on the loaded datasets in a dedicated tab. That information includes summary statistics, like the numbers of samples and genes profiled in each dataset, and plots, like the shown boxplot to visualise the distribution of normalised gene read counts per sample.

Gene expression filtering and normalisation

Next, we performed gene expression filtering and normalisation on the loaded raw gene expression read counts using psichomics default settings (based on the `edgeR` [162] and `limma` [173] R packages). We filtered out lowly-expressed genes with a minimum of 10 read counts for at least one sample and with a minimum total read counts of 15 (default settings in psichomics). We noticed that the density plot of the samples' library size (i.e. the total number of mapped reads) suggest relatively low read coverage for sample SRR2453313 (Figure 3.6a). At this time, we decided to keep this sample to compare with other samples after data normalisation.

Afterwards, we normalised gene expression by scaling raw library sizes across samples. Default gene expression normalisation scales for raw library sizes based on weighted trimmed mean of M-values (TMM) [162], followed by computation of log2-transformed counts per million values. The default normalisation is not fully effective, as very different distributions between samples are observed (Figure 3.6b).

We used voom instead, as it incorporates the mean-variance relationship of the data to normalise expression levels between samples [173]. The distributions remain

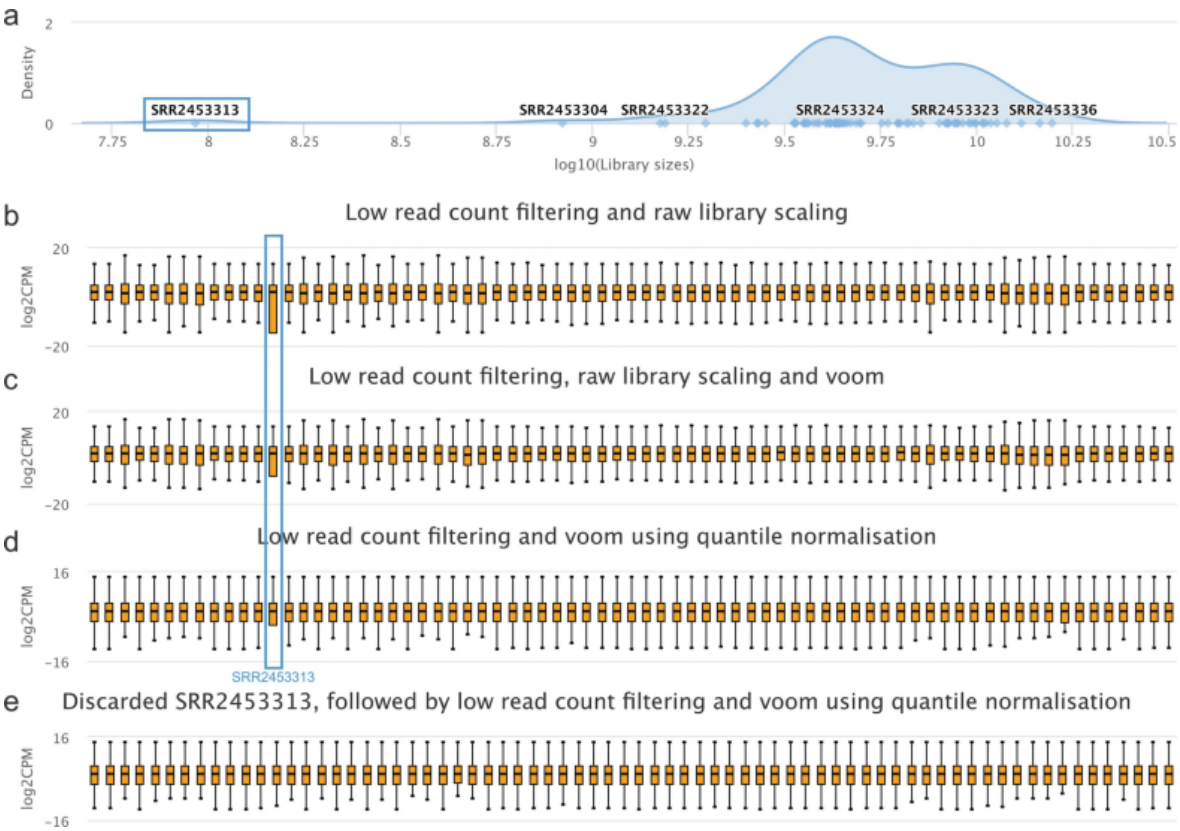


Figure 3.6: Gene expression normalisation. (a) Density plot of the distribution of raw library sizes (i.e.. total number of mapped read counts) across samples. Highlighted with the blue label is the sample with the smallest library. (b–e) Boxplots of distribution of gene expression in log2-transformed counts per million (log2CPM per sample after low read count filtering and raw library scaling (b), this procedure followed by voom modeling (c), voom modeling using quantile normalisation instead of raw library scaling (d), and the latter after discarding the sample with the smallest library, highlighted in blue (e).

heterogeneous (Figure 3.6c) based on the default weighted trimmed mean of M-values (TMM) [162], used to normalise for library sizes, so we replaced it by quantile normalisation [173]. This more vigorous normalisation of gene read counts made their distributions comparable across samples, except for that already identified sample as having the lowest library size (Figure 3.6d), so we discarded sample SRR2453313. No obvious outlying gene expression distribution is apparent after discarding that sample and renormalising (Figure 3.6e). The filtered and normalised gene expression dataset was now composed of 72 samples and 27,807 genes.

Alternative splicing quantification

The percent spliced-in (PSI) metric is commonly employed to measure the relative abundance of the inclusion isoform of an alternative splicing event [1]. For each annotated event, psichomics was used to quantify PSI values based on the ratio of splice (exon–exon) junction read counts that support the inclusion of the alternative sequence. The selected alternative splicing annotation was *Human hg38 (2018-04-30)*.

The default event types were quantified: skipped exon (SE), mutually exclusive exon (MXE), alternative first and last exon (AFE and ALE, respectively), and alternative 3′ and 5′ splice site (A3SS and A5SS, respectively). By default, only alternative splicing events with a minimum of 10 junction read counts supporting either inclusion or exclusion of the alternative sequence are considered to avoid quantifying events with insufficient evidence. For consistency with gene expression analysis, we discarded sample SRR2453313 with the lowest library size.

In total, a high number of 135,717 alternative splicing events were quantified. However, only events exhibiting some variance across samples are informative when analysing differential splicing. We therefore filtered out low-variance events with median PSI values between 0.05 and 0.95 (Figure 3.7a), avoiding those whose median PSI is consistently near 0 and 1 (i.e. alternative splicing events that are mostly constitutive). This concomitantly filters out events of very low variance (Figure 3.7b). To further select events that vary across samples based on a minimum PSI variance, we set $log_{10}(variance) > -3$ (Figure 3.7b). Alternative splicing events were further filtered based on their PSI range (maximum — minimum PSI value across samples), as a surrogate for the minimum changes in alternative splicing that can be considered biologically meaningful, by setting the minimum PSI range to 0.15 (Figure 3.7b). In the end, the number of potentially interesting events was reduced from 135,717 to 27,401.

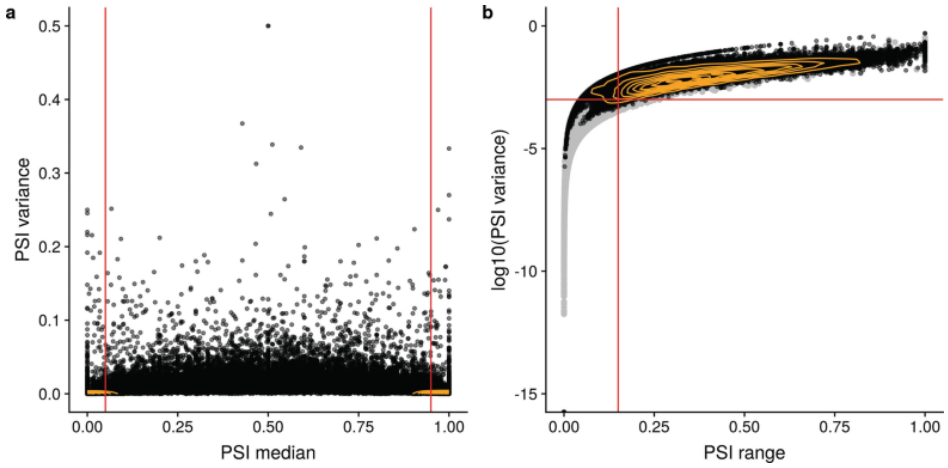
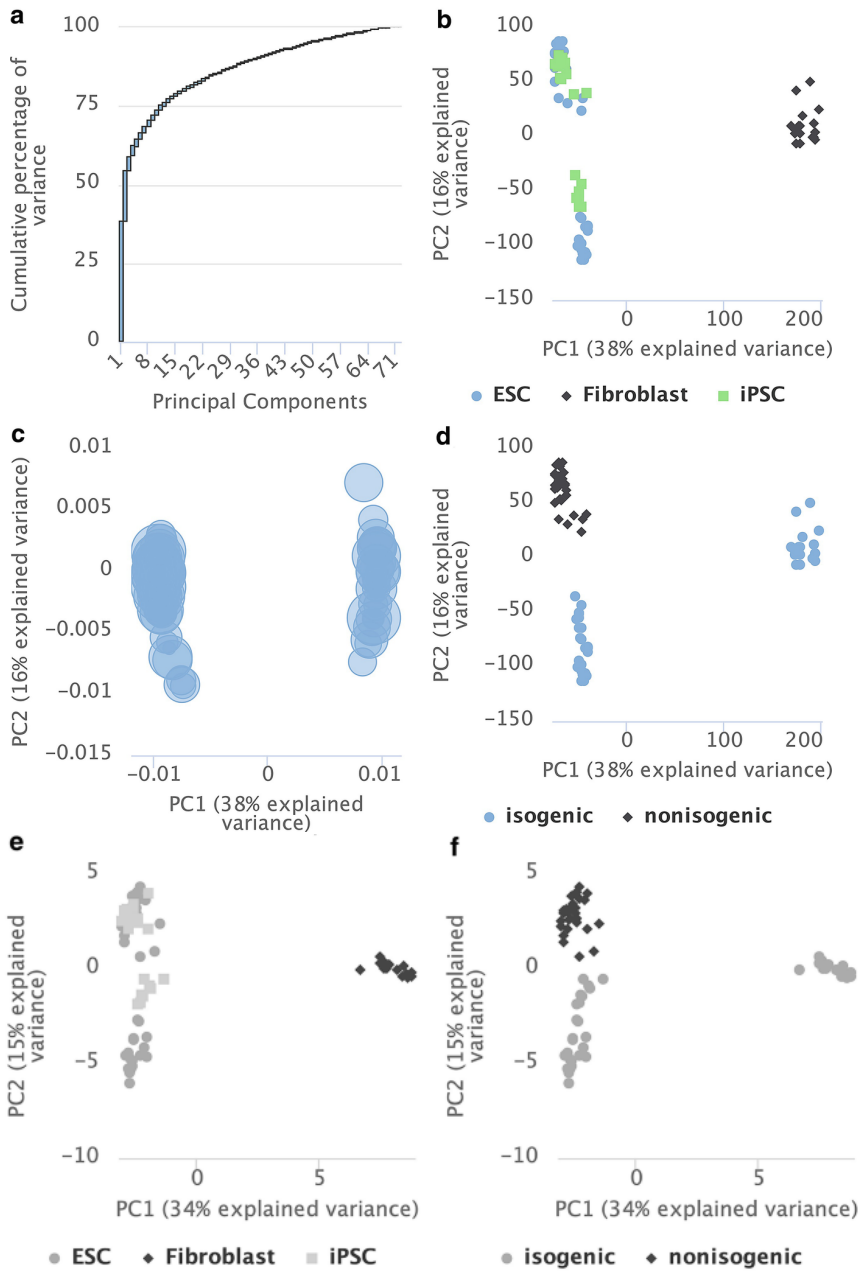


Figure 3.7: Alternative splicing quantification filtering. (a) Selection, for further analyses, of alternative splicing events with median PSI values between 0.05 and 0.95. (b) Further filtering those events with PSI range > 0.15 and log_{10}(variance) > −3. For illustration purposes, grey points represent the events discarded in panel a.

Principal component analysis (PCA)

Data groups can be created in psichomics based on sample/subject matched metadata or on alternative splicing events and respective genes. We grouped ESC and iPSC samples together to compare them against Isogenic Stem Cells and Isogenic Fibroblasts.



g

Rank	Gene	Event type	Chromosome	Strand	Contribution to PC1 (%)	Contribution to PC2 (%)	Contribution to PC1 and PC2 (%)
1	TMEM63B	Skipped exon	6	+	0.773936985786204	0.00406440276777961	0.537175533660727
2	SPAG9	Skipped exon	17	-	0.720386187642082	0.00255913011045217	0.499630467455487
3	MEST	Alternative first exon	7	+	0.642142302742031	0.0289812048145162	0.453574849978623
4	SCARB1	Alternative last exon	12	-	0.589350555318185	0.0134035507012602	0.412227675288103
5	SCARB1	Skipped exon	12	-	0.588841154632387	0.0119374182542734	0.41142404773143

Figure 3.8: PCA of normalised gene expression (a–d) and alternative splicing quantification data (e–g). (a) Plot displaying the cumulative percentage of total data variance explained by each principal component. (b, d) Scatter plots of scores of each sample on principal components 1 and 2, with samples coloured based on cell type (b) and isogenicity (d). For performance reasons, scatter plots only show the 100 most contributing variables (i.e., genes or alternative splicing events) to the selected principal components are plotted by default. (c) Scatter plot of loadings of each gene on principal components 1 and 2. Each gene's bubble size is proportional to its relative contribution to principal components 1 and 2. (e, f) Scatter plots of scores of each sample on principal components 1 and 2, with samples coloured based on cell type (e) and isogenicity (f). (g) Table of loadings of the 5 alternative splicing events contributing the most to principal components 1 and 2.

Later we performed PCA on normalised gene expression after centring and scaling the values. We allowed to impute at most 4 (i.e., around 5% of the 72 samples) tolerated missing values per row[4]. The first two principal components explain around 50% of the observed variance in the data.

The variance observed across principal component 1 seems to be related with the cell type (fibroblast versus stem cell) (Figure 3.8b), whereas principal component 2 is associated with isogenicity (i.e., isogenic vs nonisogenic; respective column named dataset type in the SRA metadata) (Figure 3.8d). From the 15 most variance-contributing genes, as displayed in the table below the loading plot, at least *DNMT3B* and *RBPMS2* have been previously associated with pluripotency [192]. Specifically, *RBPMS2* has been reported to play a role in self-renewal following the knockdown of *ESRP1*, reported to act as a regulator of pluripotency [192].

Similarly, we performed and plotted PCA on alternative splicing data[5]. Akin to the observations from PCA plots on normalised gene expression, principal component 1 appear to be associated with cell type and principal component 2 with isogenicity (Figure 3.8e-f). The table below the loading plot allows to assess which alternative splicing events contribute the most to those separations (Figure 3.8g). Some of the cognate genes of those events have already been reported to be involved in conserved splicing programs in stem cell differentiation, including KIF13A and PALM [186].

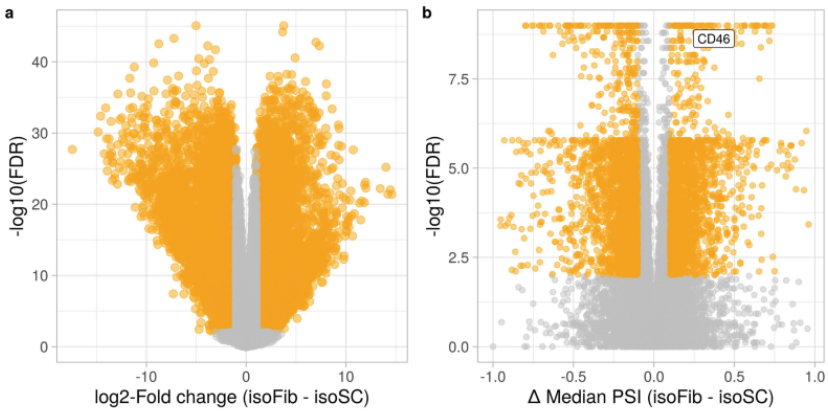
Differential expression and splicing analysis

We performed differential gene expression and differential splicing analysis between isogenic stem cells and isogenic fibroblasts (Figure 3.9). First, normalised gene expression was linearly modelled, with explanatory variables defined based on the selected groups. Moderated t-tests and log-odds of differential expression were then computed by empirical Bayes moderation of the standard errors towards a common value [173].

The volcano plot of differential splicing analysis between isogenic stem cells and fibroblasts (Figure 3.9b) exhibits two strata, that is, two modes of Wilcoxon's test significance. The top significance stratum is the result of using such nonparametric test (motivated by the non-normality of PSI distributions) when all values in one of the groups are higher than those in the other group; the number of tested groups also affects the significance of the difference. The lower significance stratum relates to a consistent number of repeated values between samples (usually occurring when one of the groups is closer to a PSI value of 0 or 1). As the Wilcoxon's test is rank-

[4]As PCA cannot be performed on data with missing values, these either have to be removed (discarding genes or alternative splicing events from the analysis) or imputed (i.e., replaced by estimates). By default, if a gene (or alternative splicing event) has 5% or less missing values in psichomics, the median expression (or PSI) of that gene (or event) across samples is assigned to the missing values. Genes (or events) with more than 5% missing values (by default) are discarded from the PCA.

[5]PSI values are not scaled by default, given they are dimensionless ratios that range from 0 to 1.



c

Event type	Chromosome	Strand	Gene	PSI distribution	Samples (isoSC)	Samples (isoFib)	Wilcoxon p-value (BH adjusted)
All	All	A	A	All	All	All	All
Skipped exon	1	+	PLEKHM2		25	17	9.36829674399094e-10
Skipped exon	1	+	NASP		25	17	9.36829674399094e-10
Skipped exon	1	+	NASP		25	17	9.36829674399094e-10
Skipped exon	1	-	C1orf43		25	17	9.36829674399094e-10
Skipped exon	1	+	CD46		25	17	9.36829674399094e-10
Skipped exon	1	+	RCOR3		25	17	9.36829674399094e-10

Figure 3.9: Differential expression (a) and splicing (b, c) analyses. (a, b) Volcano plots where orange-highlighted genes/events correspond to adjusted p-value < 0.01 and either $|log_2(\text{Fold change})| > 1$ (a) or $|\Delta \text{ median PSI}| > 0.1$ (b). The *CD46* penultimate exon inclusion is labeled in b with the cognate gene symbol. (c) Table showing a subset of the differential splicing results, sorted in ascending order by the adjusted p-value of Wilcoxon's rank-sum test. PSI distributions are colored by groups: pink for isogenic stem cells and green for isogenic fibroblasts. The selected alternative splicing event (in blue) depicts the *CD46* penultimate exon inclusion. Legend: isogenic fibroblasts (isoFib) and isogenic stem cells (isoSC).

based, some ranks are not unique if there are two identical values; this occurrence (called a tie) hampers the computation of exact p-values. Increasing the number of identical values when performing the Wilcoxon's test decreases the significance of the comparison, which may bias the significance of differentially spliced events when one of the groups is characterised by PSI values close to 0 or 1 (constitutive splicing) and will therefore present many 0's or many 1's.

Skipping of *CD46* penultimate exon

The skipping of the penultimate exon of *CD46* is one of the most significantly differentially spliced sequences between isogenic fibroblasts and isogenic stem cells in our analyses (Figure 3.9b-c). Based on PSI distributions for the *CD46* penultimate exon in the different cell types, higher inclusion of the *CD46* penultimate exon is associated with fibroblasts, whereas lower inclusion is associated with stem cells, both ESC and iPSC (Figure 3.10a). The inclusion of *CD46* penultimate exon leads to a premature termination codon (PTC) that may cause the respective transcript to be targeted for nonsense-mediated decay (NMD) [193].

We also tested the correlation between the gene expression of a list of RNA-binging proteins [153] against the quantification of the *CD46* penultimate exon inclusion, thus identifying a potential regulatory role from the RNA-binding epithelial splicing regulatory proteins 1 and 2 (ESRP1 and ESRP2; Figure 3.10b-c). The skipping of *CD46* penultimate exon has been previously reported as being regulated by the ESRP1/2 proteins that are involved in the epithelial–mesenchymal transition and associated with the generation of cancer stem cells [189, 193]. ESRP1 has also been reported to regulate ES cell differentiation [192].

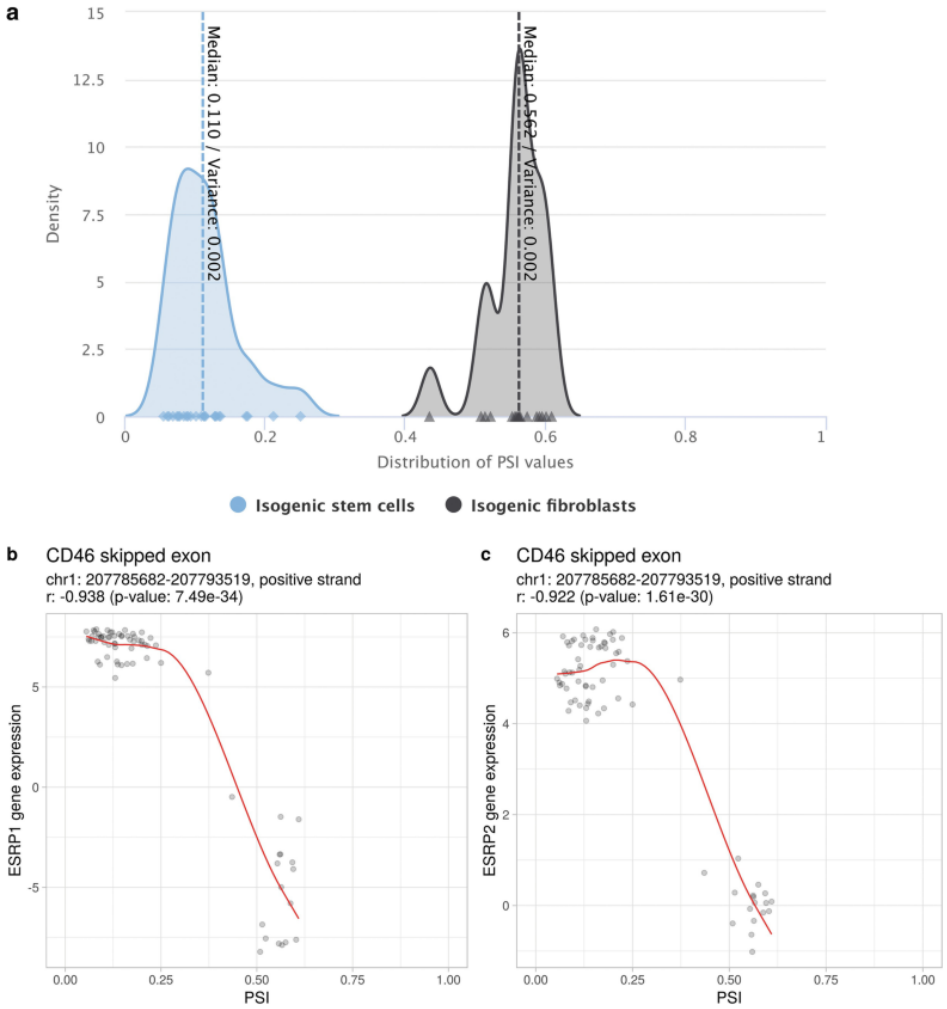


Figure 3.10: Alternative splicing of the *CD46* penultimate exon. (a) Density plots of distributions of *CD46* penultimate exon PSI values across samples, colored by isogenic stem cells (pink) and isogenic fibroblasts (green). (b-c) Scatterplots of PSI values for *CD46* penultimate exon inclusion versus normalised *ESRP1* (b) and *ESRP2* (c) expression across samples. The red line illustrates the fitted Loess regression curve. The Pearson's correlation coefficients (r) and associated p-values are shown. (d) Genomic alignment of *CD46* transcript isoforms with penultimate exon highlighted in an orange shade (the gray shade includes the neighboring constitutive exons to define the entire alternative splicing event).

Extending the analyses to GTEx and TCGA

To analyse the correlation between *ESRP1/2* gene expression across GTEx tissues and TCGA tumour types against the alternative splicing quantification of the skipping of the penultimate exon of *CD46*, gene expression data was filtered and normalised using voom with quantile normalisation [173] and alternative splicing was quantified based on the original human hg19 alternative splicing annotation combining events from multiple sources.

In GTEx tissues where *ESRP2* expression substantially varies across individuals (e.g., breast, testis, vagina, small intestine, stomach, and prostate), it is, as expected, negatively correlated with *CD46* penultimate exon inclusion (Figure 3.11). Correlation with *ESRP1* expression cannot be performed, as the gene was filtered out for having low read counts across samples, suggesting its low expression in differentiated GTEx tissues. As for TCGA, the negative correlation between *ESRP1/2* expression and *CD46* penultimate exon inclusion is observed in most cancer types, including breast cancer (Figure 3.12 and Figure 3.13).

Pancancer prognostic value of the skipping of *CD46* penultimate exon

The prognostic value of a given alternative splicing event (or gene) may be evaluated by separating subjects based on a PSI cutoff for a given alternative splicing event (or expression cutoff for a given gene). The survival differences are then log-rank tested based on Kaplan-Meier estimators. In this context, we used clinical and transcriptomic data from TCGA to evaluate the prognostic value of the skipping of *CD46* penultimate exon based on overall survival curves across TCGA tumour types to compare tumour samples[6] with low and high inclusion of the *CD46* penultimate exon (Figure 3.14).

We performed overall survival analysis by selecting the follow-up time as days to death and the event of interest as death. Analysing days to death as the follow-up time and death as the event of interest is known as an overall survival analysis, that is, the study of the time until the subject's death following diagnosis. In cases where days to death is not available, the days to last follow-up are considered instead in psichomics. In some TCGA records, the days to last follow-up may be lower than days to death; as we are interested in prognosis, we used days to death instead.

A $-log_{10}$(p-value) plot by cutoff displays the p-values of the log-rank test of survival across multiple PSI cutoffs for the selected alternative splicing event. The PSI cutoff

[6]Survival analysis is performed on individuals (patients), not samples. As such, when separating individuals by a given PSI cutoff of a user-provided alternative splicing event, the PSI value of a sample is assigned to its respective subject. If a subject has multiple samples, their average PSI is used. However, not all samples should be used in some cases; for instance, PSI values from normal tissue samples should be discarded when studying disease. For TCGA data, only tumour samples (by default) are used to assign PSI values per subject for survival analysis.

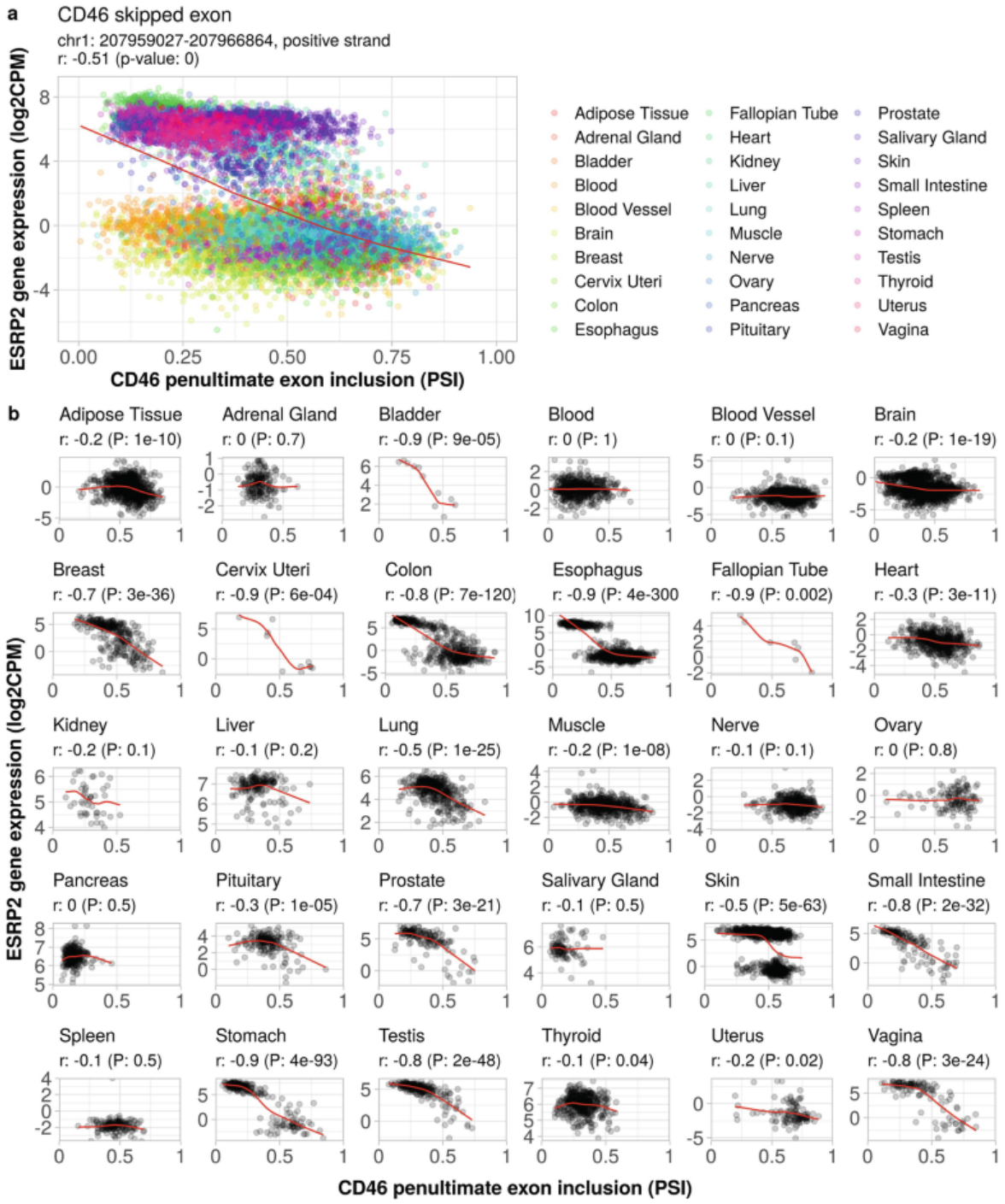


Figure 3.11: Scatterplots of normalised *ESRP2* expression versus PSI values for *CD46* penultimate exon inclusion across GTEx tissues, altogether (a) and by tissue (b). For each plot, the red line illustrates the fitted Loess regression curve. The Pearson's correlation coefficients (r) and associated p-values (P) are shown.

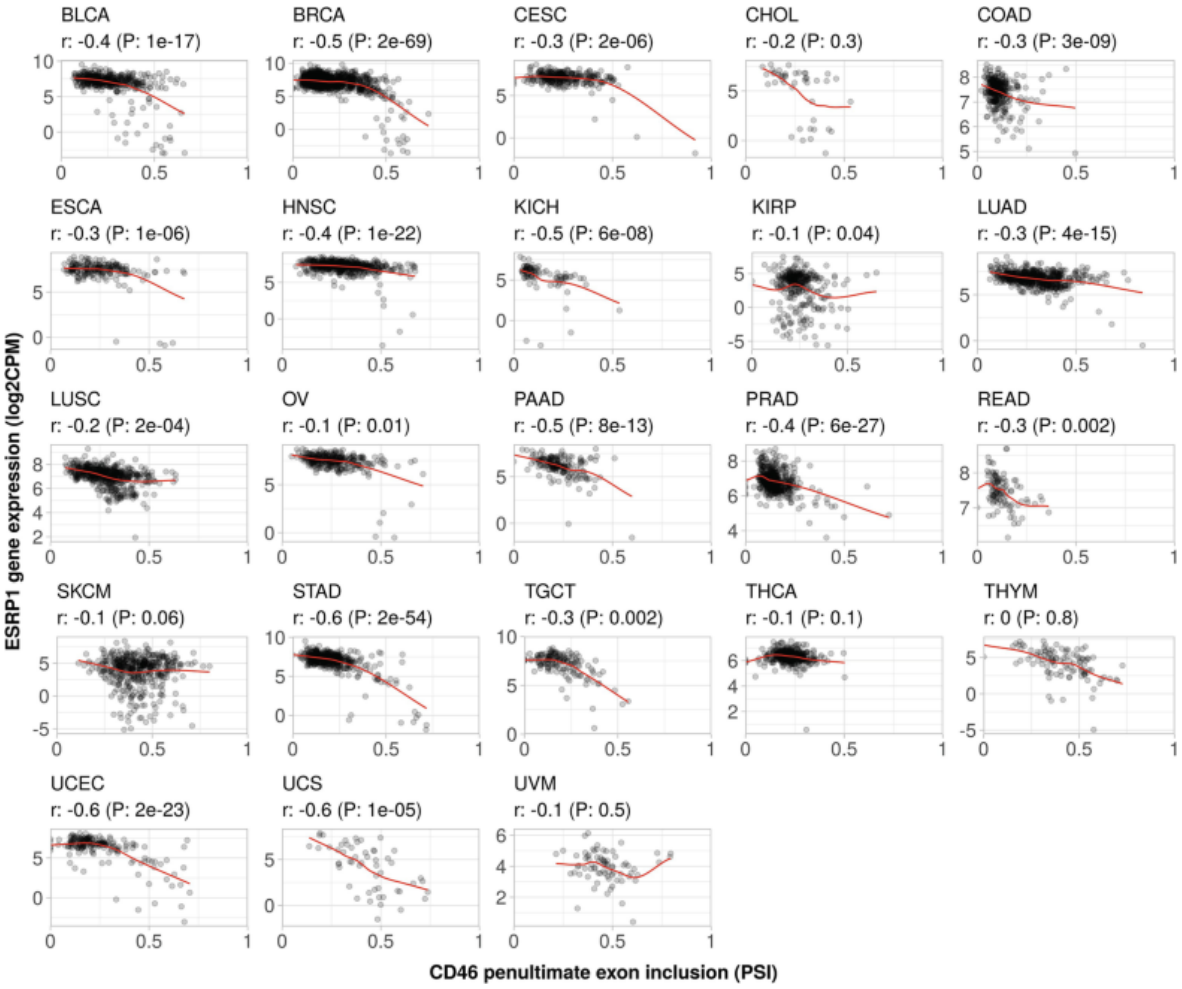


Figure 3.12: Scatterplots of normalised *ESRP1* expression versus PSI values for *CD46* penultimate exon inclusion across TCGA tumour types. For each plot, the red line illustrates the fitted Loess regression curve. The Pearson's correlation coefficients (r) and associated p-values (P) are shown.

Legend: ACC adrenocortical carcinoma, BCLA urothelial bladder carcinoma, BRCA breast invasive carcinoma, CESC cervical squamous cell carcinoma and endocervical adenocarcinoma, CHOL cholangiocarcinoma, COAD colon adenocarcinoma, DLBC lymphoid neoplasm diffuse large B-cell lymphoma, ESCA esophageal carcinoma, GBM glioblastoma multiforme, HNSC head and neck squamous cell carcinoma, KICH kidney chromophobe, KIRC kidney renal clear cell carcinoma, KIRP kidney renal papillary cell carcinoma, LGG brain lower grade glioma, LIHC liver hepatocellular carcinoma, LUAD lung adenocarcinoma, LUSC lung squamous cell carcinoma, MESO mesothelioma, OV ovarian serous cystadenocarcinoma, PAAD pancreatic adenocarcinoma, PCPG pheochromocytoma and paraganglioma, PRAD prostate adenocarcinoma, READ rectum adenocarcinoma, SARC sarcoma, SKCM skin cutaneous melanoma, STAD stomach adenocarcinoma, TGCT testicular germ cell tumours, THCA thyroid carcinoma, THYM thymoma, UCEC uterine corpus endometrial carcinoma, UCS uterine carcinosarcoma, UVM uveal melanoma

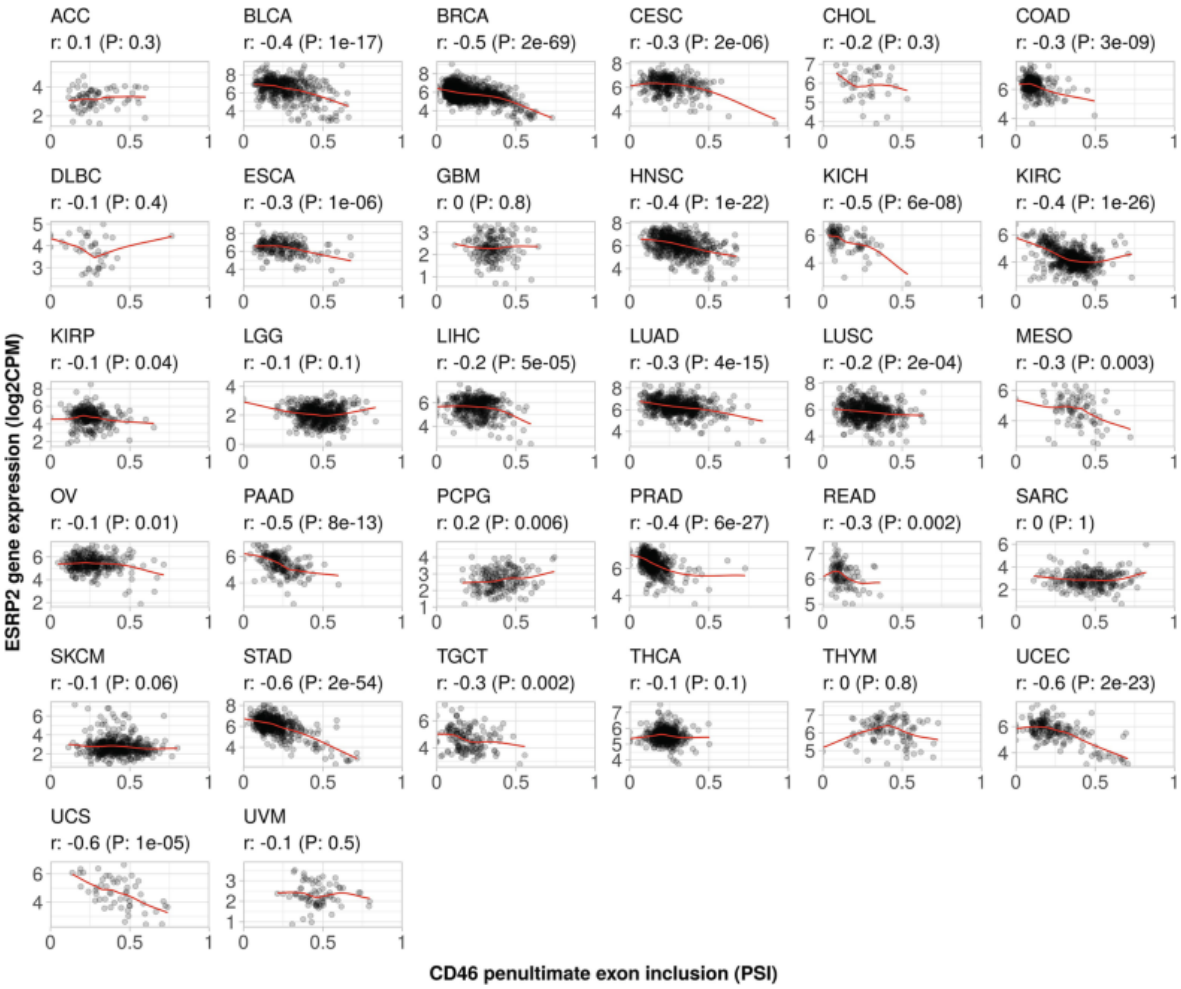


Figure 3.13: Scatterplots of normalised *ESRP2* expression versus PSI values for *CD46* penultimate exon inclusion across TCGA tumour types. For each plot, the red line illustrates the fitted Loess regression curve. The Pearson's correlation coefficients (r) and associated p-values (P) are shown. See caption of Figure 3.12 for legend.

maximising the significance of the survival difference is automatically selected[7]. The splicing of *CD46* penultimate exon seems to have prognostic value in select cancer types, such as brain lower-grade glioma and lung adenocarcinoma (Figure 3.14) .

3.3.2 Time benchmarking

The runtime 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 in TCGA, thus being the cohort taking more time to load, quantify and analyse alternative splicing

[7]Survival differences are tested based on multiple PSI cutoffs for a given alternative splicing event. If the PSI cutoff suggested by psichomics has unbalanced number of subjects between groups (e.g., in Figure 3.10d, the lowest log-rank p-value corresponds to a comparison of 1 versus 506 subjects), we should use another cutoff with more reasonably balanced groups and for which the p-value is still significant. Sample size calculations can be performed based on test assumptions (e.g., probability of failure for each group during the study) using existing R packages, such as *powerSurvEpi* [194].

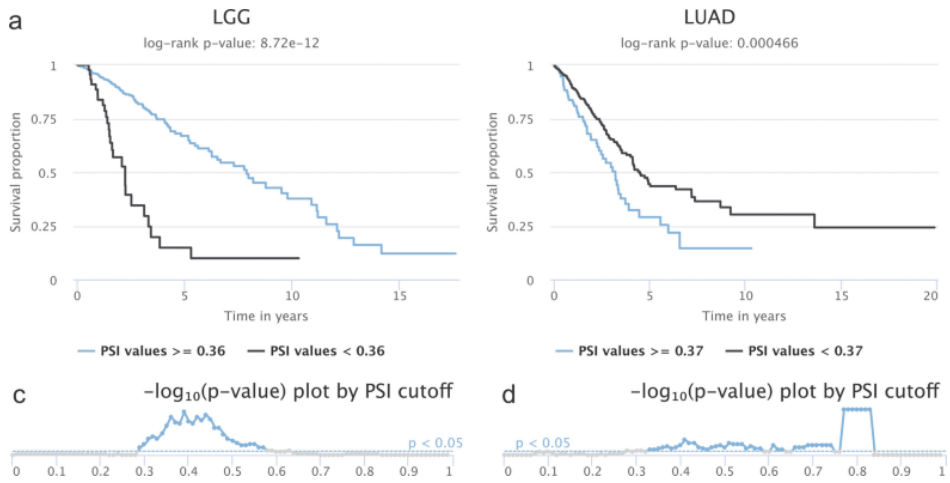


Figure 3.14: Prognostic value of *CD46* penultimate exon inclusion across select TCGA cancer types. (a, b) Kaplan-Meier plots of overall survival for all patients stratified by the respective alternative splicing event's PSI cutoff that maximised the significance of differences in survival between patient groups with a reasonable number of subjects within each group. Each patient was assigned the PSI value of their tumour sample(s). (c, d) Log-rank's $-log_{10}$(p-value) plot by PSI cutoff. Note that in panel d, for PSI values around 0.8, there are high log-rank $-log_{10}$(p-value) although only one individual is being compared against 506 subjects. Legend: LGG brain lower grade glioma, LUAD lung adenocarcinoma.

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 3.15A).

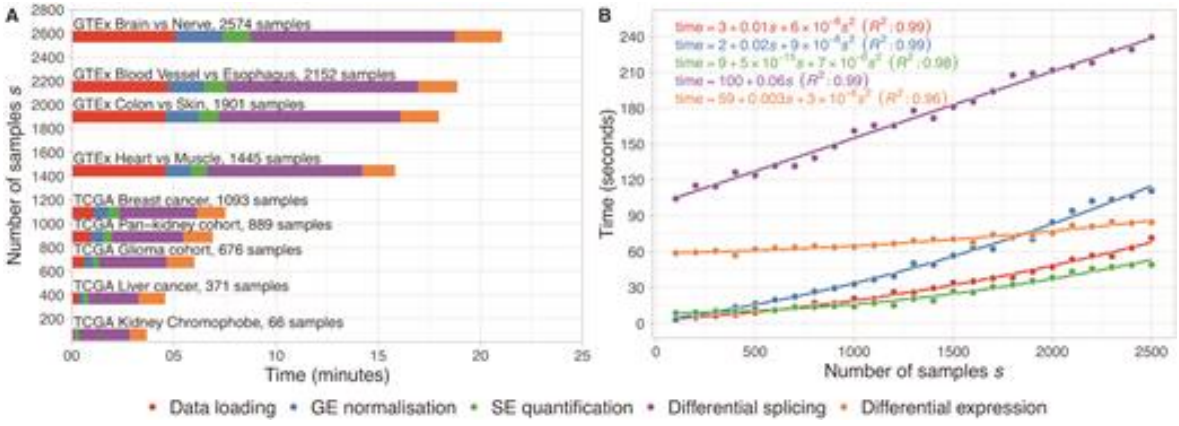


Figure 3.15: Performance benchmark for alternative splicing analysis using RNA-seq data from multiple TCGA and GTEx sample types. (A) Median times of 10 runs of data loading, gene expression (GE) normalisation, skipped exon (SE) event quantification and differential expression and splicing analysis (normal 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.

Synthetic datasets for gene expression and exon-exon junction quantification of mul-

tiple 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 3.15B). 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.

3.3.3 Alternative splicing quantification benchmarking

Although jSplice's [160] and DIEGO's [161] 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 [124] across multiple tissue and cell line samples from human, mouse and chicken [125]. 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, particularly for intermediate PSI values, with 90% prediction intervals < 0.1 for coverage higher than a few hundred reads.

Alternative splicing events annotated by TCGASpliceSeq [130], 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).

3.4 Conclusion

Alternative splicing is a regulated molecular mechanism involved in multiple cellular processes and its dysregulation has been associated with diverse pathologies [195, 196, 1, 197]. 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 [196, 1, 81, 2, 3]).

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.

To ensure researchers with different skills can take the most out of psichomics, we added an intuitive and more accessible graphical interface, while still supporting a command-line interface. psichomics has recently been deployed online at `compbio.imm.medicina.ulisboa.pt/psichomics`[8] to allow the on-demand use of the latest version of psichomics 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, psichomics only quantifies alternative splicing events based on exon–exon junction read counts, limiting the types of alternative splicing events profiled. 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 transcriptional variations [165]. However, although GTEx (but neither TCGA nor recount2) readily provides intron and exon body read quantification for retrieval, none provides exon–intron junction quantification. To overcome this, psichomics allows to import alternative splicing events quantified from other programs, including VAST-TOOLS that quantifies intron retention events.

Another limitation is psichomics 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. Even so, 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 [126], which is able to generate de novo annotations[9].

Since its publication, psichomics has been used to analyse alternative splicing in multiple scientific articles, such as [11, 12, 13, 14]. Based on these citations and positive user feedback, we believe that fellow researchers and clinicians are 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.

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

[9] More information in `nuno-agostinho.github.io/psichomics/articles/AS_events_preparation`.

Chapter 4

cTRAP

During a stormy day in our 2017 Madeira Lab retreat, we brainstormed the unique propositions of the lab that could most benefit 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 [15], a repository of transcriptomic signatures for thousands of genetic (gene overexpression or knockout) and pharmacological perturbations in human cancer cell lines.

We thus developed cTRAP, an R package and web app to compare user-provided differential gene expression profiles with the perturbations available from CMap, allowing to infer putative candidate molecular causes for the observed differences, as well as compounds that may promote or revert them (Figure 4.1).

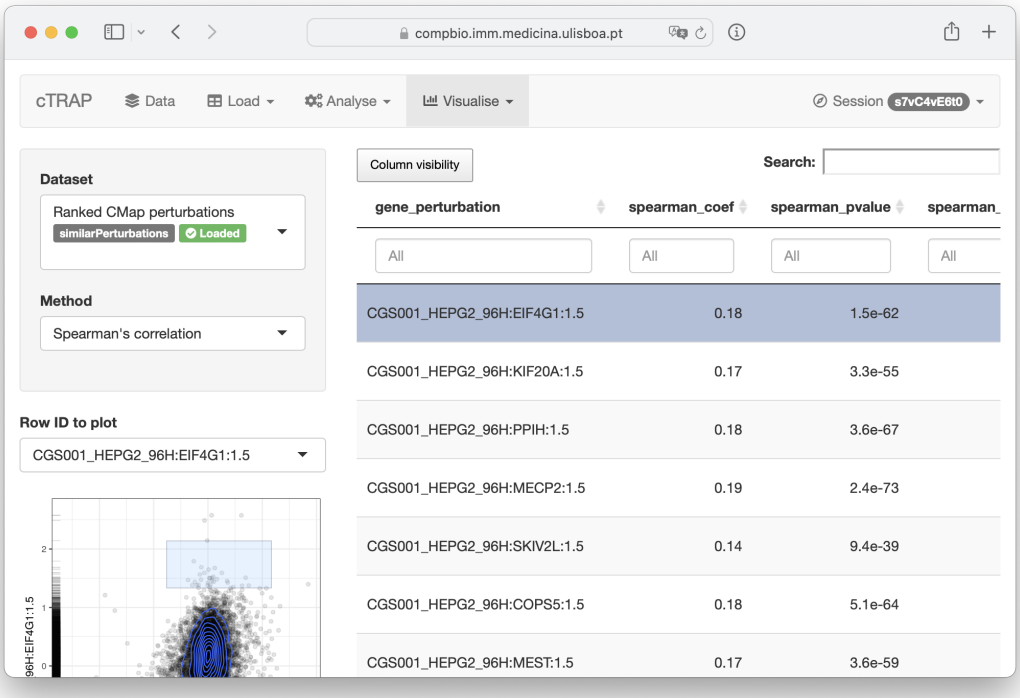


Figure 4.1: cTRAP global interface screenshot (21 Dec 2021).

After releasing the first version in Bioconductor, multiple features were added (Table 4.1). Inspired by the method used to compare gene expression changes against the CMap database, we also added a way to predict targeting drugs by using drug sensitivity datasets that featured gene expression for multiple genes in many cell lines. Additionally, cTRAP also allows to analyse the enrichment of molecular descriptors for compounds from NCI60 and CMap. More recently, we developed a visual interface to host cTRAP online with support for user sessions and background tasks.

Table 4.1: Major cTRAP milestones.

Version	Release date	Main features
1.0	2 Nov 2018	Compare differential expression profiles against CMap data[a]
1.4	12 Nov 2019	Predict targeting drugs using NCI60, CTRP and GDSC data Analyse drug set enrichment using molecular descriptors Load and process 21GB CMap z-scores file by chunks
1.8	30 Oct 2020	Include graphical functions to load data and analyse results
1.10	20 May 2021	Improve speed and memory usage when comparing data Set custom size for data chunks (1 GiB by default)
1.12	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 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 [15]. 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 [15]. However, `clue.io` limits the maximum number of input genes for CMap queries (150 up-regulated and 150 down-regulated genes), 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 [16], the Cancer Therapeutics Response Portal (CTRP) [17] and the Genomics of Drug Sensitivity in Cancer (GDSC) [18], to identify compounds that could target the phenotypes associated with the user-provided differential expression profiles. In cTRAP, similarity between differential gene expression results is measured by gene set enrichment [15, 19] and correlation scores.

cTRAP is available online as a web app at `compbio.imm.medicina.ulisboa.pt/cTRAP`, but can be locally installed using Bioconductor (`bioconductor.org/packages/cTRAP`) or Docker (`nunoagostinho/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 (Figures 4.2 and 4.3).

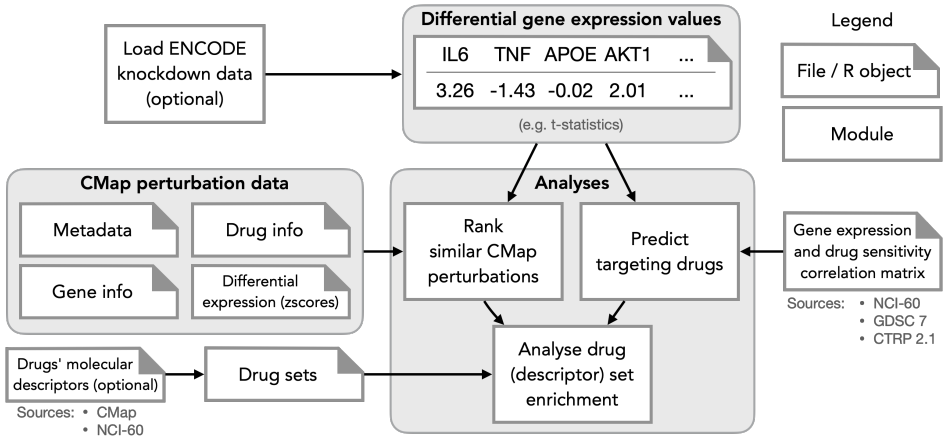


Figure 4.2: cTRAP workflow. cTRAP allows to perform three analyses: **(1) rank similar CMap perturbations** by comparing user-provided differential gene expression values against CMap perturbation data, **(2) predict targeting drugs** by comparing user-provided differential gene expression values against 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.

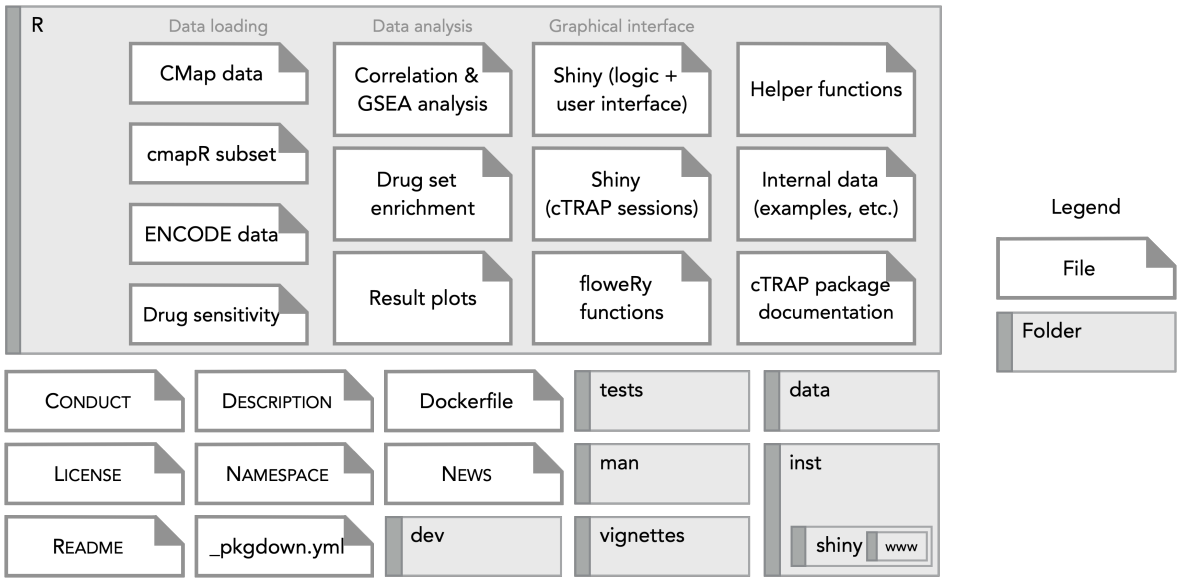


Figure 4.3: Visual representation of cTRAP's file structure. As usual in an R package, the `R` folder contains the scripts with cTRAP functions and data. `dev` is a custom folder that stores supporting scripts (e.g. test workflows and benchmarks); its contents are not included when building the R package.

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 [173]. 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

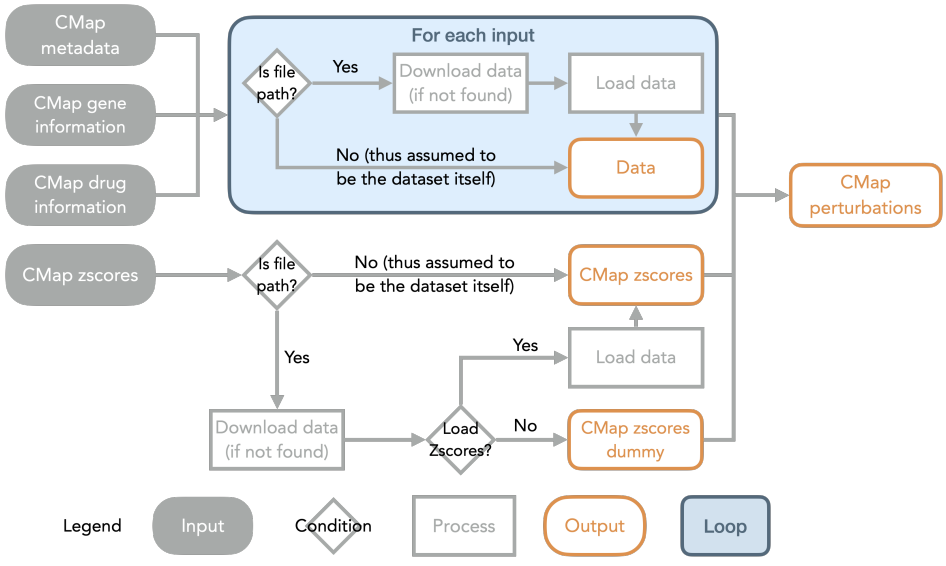


Figure 4.4: Loading data from CMap perturbations. Input arguments support either the data itself (as data frames) or their respective file path. If the file path directs to a non-existing file, data is first downloaded and then saved to the given file path. To avoid high memory usage, CMap perturbations' 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.

file is not found) and process CMap differential expression profiles z-scores (GCTX file) and gene and compound information (Figure 4.4). 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 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) [15] 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 (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 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 per comparison method via `parallel::mclapply()`[2], enabled by setting the number of threads to 2 or higher.

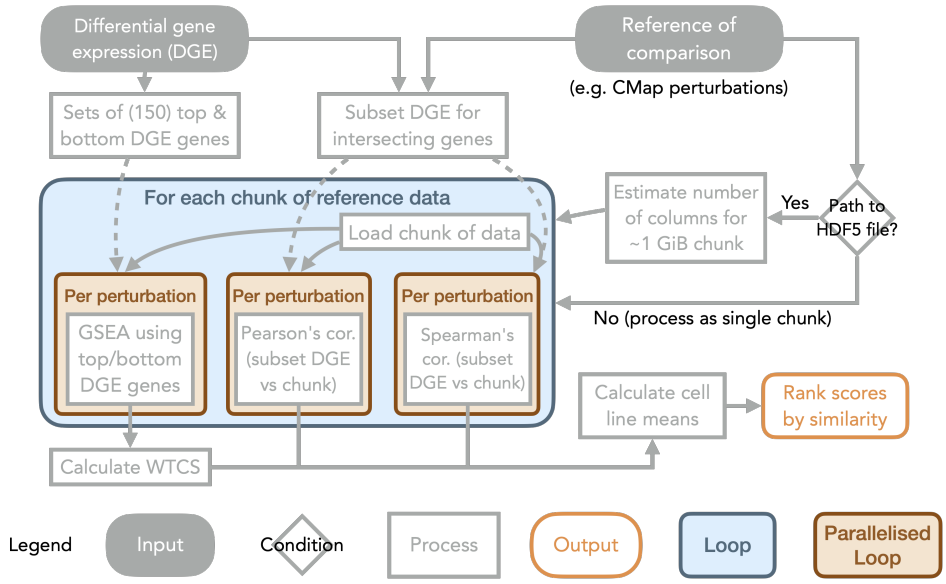


Figure 4.5: cTRAP similarity analysis. User-provided differential gene expression is compared against a reference (e.g. differential expression z-scores of CMap perturbations) and ranked by similarity. If the reference of comparison is a HDF5 file path, the file is processed in 1GiB chunks to minimise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

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 [16], Cancer Therapeutics Response Portal (CTRP) 2.1 [17] and Genomics of Drug Sensitivity in Cancer (GDSC) 7 [18]. 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()` parallelises tasks via forking where multiple child processes are spawned and share their parent's memory. Forking is unavailable in Windows and its alternatives were deemed unsatisfactory, given that they copy 1GiB chunks per thread, significantly slowing down runtime.

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 against the rank product's rank of similar perturbations.

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) characteris-

tics. 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 `analyseDrugSetEnrichment()` function 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 a development version of cTRAP 1.10 (commit `296f9b2` from January 2022). 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 (`cTRAP::diffExprStat` object).

Using the same input, we measured heap memory usage of cTRAP 1.10 dev (`296f9b2`) while ranking CMap perturbations when running R 4.0.3 in debug mode with heaptrack 1.0.0[3]. For R to work properly with heaptrack, the `/usr/bin/R` file was edited – all lines of the last *if* statement were commented out, except for:

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

Afterwards, we benchmarked the memory usage while running cTRAP with:

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

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

[3]heaptrack is an open-source memory allocation profiler available at `github.com/KDE/heaptrack`

scripts are open-source and describe how to profile time and memory in cTRAP and plot subsequent results: `github.com/nuno-agostinho/cTRAP/tree/master/dev/benchmark`.

4.2.6 Continuous integration

Akin to psichomics (subsection 3.2.13: **Continuous integration**), GitHub Actions are used with cTRAP to update its Docker images in Docker Hub (`nunoagostinho/ctrap`) and GitHub (`github.com/nuno-agostinho/cTRAP`); update website documentation via `roxygen` [143] and `pkgdown` [198]; and check for errors and warnings when building cTRAP in Windows, macOS and Linux.

4.3 Results

cTRAP's web app is available at `compbio.imm.medicina.ulisboa.pt/cTRAP`. Alternatively, users can install cTRAP in their own computers, allowing them to use local computing resources. Similarly to psichomics, cTRAP offers both graphical and command-line interfaces. Although most features are common to both interfaces, we recommend less experienced users to opt for the Shiny-based graphical interfaces.

4.3.1 Case study

For this case study, we used RNA-seq data from EIF4G1 shRNA knockdown experiments in HepG2 cell line from the ENCODE project. The RNA-seq processed data (gene quantifications from RSEM method) for the EIF4G1 knock-down and respective controls (two replicates each) was automatically downloaded by cTRAP.

Differential gene expression analysis of ENCODE RNA-seq data

Gene expression data (read counts) were quantile-normalized using voom, followed by differential expression analysis performed using `limma` [173]. We used the respective t-statistic as our metric of differential expression to compare with CMap's gene knockdown perturbations in the same cell line (HepG2)[4].

Afterwards, we filtered the metadata to CMap gene knockdown perturbations in HepG2 and loaded associated gene information and differential gene expression data based on the given filename. The differential gene expression z-scores from CMap were also loaded for small molecule perturbations.

Differential gene expression data for each CMap perturbation are available in normalised z-score values [15].

[4]This comparison could also be done to perturbations in a different cell line (or in all cell lines using the average result across cell lines).

Comparison with CMap perturbations

The `rankSimilarPerturbations()` function compares the differential expression metric (the t-statistic, in this case) against the CMap perturbations' z-scores using the available methods:

- Spearman's correlation
- Pearson's correlation
- Gene Set Enrichment Analysis (GSEA), where the most up- and down-regulated n genes from the user's differential expression profile are used as gene sets (by default, $n = 150$ genes)

The output table contains the results of the comparison with each perturbation tested, including the test statistics (Spearman's correlation coefficient, Pearson's correlation coefficient and/or GSEA score), the respective p-value and the Benjamini-Hochberg-adjusted p-value (for correlation statistics only). When performing multiple methods, the rank product's rank will be included to summarise other method's rankings.

The Gene Set Enrichment Analysis (GSEA) score is based on the Weighted Connectivity Score (WTCS), a composite and bi-directional version of the weighted Kolmogorov-Smirnov enrichment statistic (ES) [15]. To calculate the GSEA score, GSEA is run for the most up- and down-regulated genes from the user's differential expression profile. The GSEA score is the mean between EStop and ESbottom (however, if EStop and ESbottom have the same sign, the GSEA score is 0).

If a perturbation has a similar differential expression profile to our data (higher GSEA score), we expect to see the most up-regulated (down-regulated) genes in the perturbation enriched in the top (bottom) n differentially expressed genes from our data.

To analyse the relationship between the user-provided differential expression profile with that associated with a specific perturbation, scatter plots (for Spearman and Pearson analyses) and GSEA plots are available.

For instance, let's plot the relationship between EIF4G1 shRNA knockdown from ENCODE with the CMap knockdown perturbations.

Predicted targeting drugs

Compounds that target the phenotypes associated with the user-provided differential expression profile can be inferred by comparing against gene expression and drug sensitivity associations. The gene expression and drug sensitivity datasets derived from the following sources were correlated using Spearman's correlation across the available cell lines.

To use an expression and drug sensitivity association based on CTRP 2.1 (GDSC 7 and NCI60 could be used instead) to infer targeting drugs for the user's differential expression profile.

Compounds are ranked by their relative targeting potential based on the input differential expression profile (i.e. the 1st-ranked compound has higher targeting potential than the 2nd-ranked one).

Table 4.2: Drug sensitivity datasets statistics. Number of screened compounds and human cancer cell lines.

Source	Compounds	Cell lines
NCI60	> 100 000	60
GDSC 7	481	860
CTRP 2.1	138	∼ 700

Candidate targeting drugs were plotted against the similarity ranking of their perturbations towards the user's differential expression profile. Note that the highlighted values are the same compounds for the following plots annotated with their name, gene target and mechanism of action (MOA), respectively.

Molecular descriptor enrichment analysis

Using our candidate targeting drugs, we analysed the enrichment of 2D and 3D molecular descriptors based on CMap and NCI60 compounds. Our list of targeting drugs is particularly enriched in specific drug descriptors that allows us think about the relevance of these descriptors for targeting a phenotype of interest.

4.3.2 Time and memory optimisation

cTRAP allows comparing user-provided differential expression results against those from CMap perturbations that is loaded based on their public data, including a 21GB GCTx file containing the CMap perturbation z-scores. The GCTX format stores annotated, high-dimensional data matrices and is based on the HDF5 format for efficient indexing and loading of subsets of the whole data [199]. Instead of loading the whole GCTx file into memory, cTRAP 1.4 and newer minimises peak RAM usage by loading and processing subsets of perturbation z-scores in one of two ways:

- Load and process selected perturbations only (e.g. compound perturbations), avoiding loading perturbations that will not be compared.
- Load and process selected perturbations by 1 GiB chunks (default).

Although the concept of processing by chunks helped reducing the memory footprint of cTRAP, we later refactored the code of the comparisons in cTRAP 1.10, a milestone that included multiple improvements to speed and memory:

- Faster GSEA-based score calculation by improving code efficiency.

- Slightly improved runtime by avoiding redundant loading of data chunks. This change was also required to enable multi-thread support in systems that support process forking (e.g. Linux and macOS, but not Windows).
- Optional argument to set a custom size for data chunks (1 GiB by default).
- As the NCI60 gene expression and drug sensitivity correlation matrix is also big (3.3GB), we saved that matrix as a HDF5 file that is then loaded and processed by cTRAP in chunks of 1 GiB (by default), minimising peak RAM usage when predicting targeting drugs based on the NCI60 data.

We benchmarked time and memory when ranking CMap perturbations based on Spearman's correlation, Pearson's correlation and GSEA-based score using a development version of cTRAP 1.10 (commit `296f9b2`) and using the t-statistics for differential expression between EIF4G1 knockdown versus control based on ENCODE gene expression data from HepG2 cell line as input. The number of CMap compound perturbations (241 258) is much higher than the knockdown (48 862) and over-expression (24 627) perturbations and this is expected to reflect on time and memory for data processing.

When ranking CMap compound perturbations in cTRAP 1.8 and 1.10, the runtime was decreased from 95 to 53 minutes when loading the whole data and from 77 to 28 minutes when loading by chunks (Figure 4.6). For each chunk, we parallelised the calculations performed per perturbation and it can help to speed up runtime comparing against CMap compound perturbations with 4 threads from 28 to 12 minutes. However, the usage of 8 threads is not recommended as the runtime is similar to using 4 threads

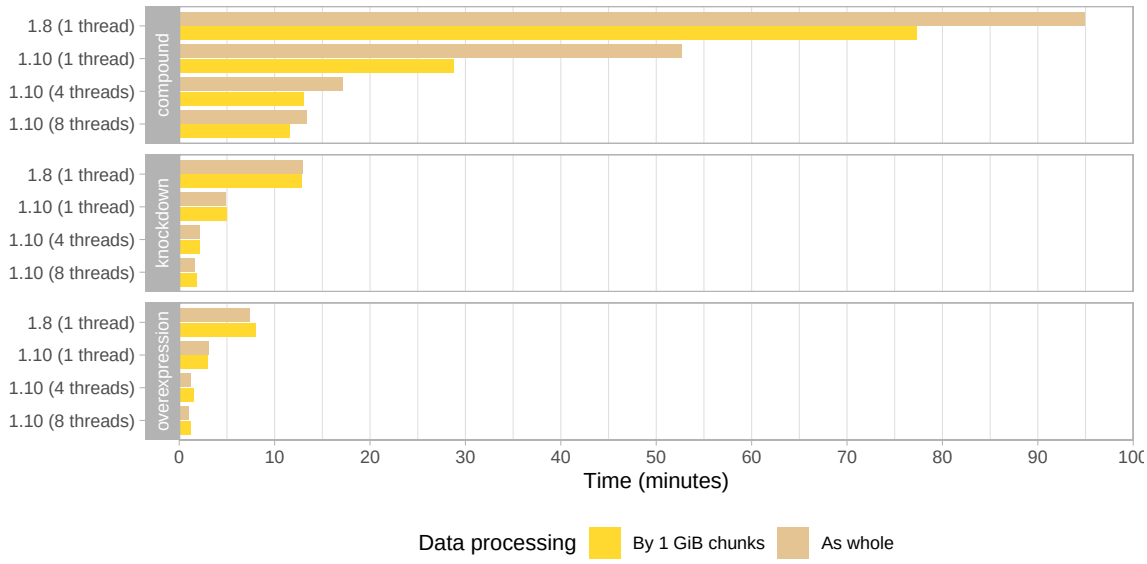


Figure 4.6: Time benchmark of CMap perturbation ranking. Times were measured for cTRAP 1.8 (1 thread only) and 1.10 (1, 4 and 8 threads) for over-expression, knockdown and compound perturbations whose data was loaded by 1 GiB chunks or as whole.

while consuming the double number of threads.

Benchmarked memory was annotated based on the different internal steps performed by cTRAP (Figure 4.7). When ranking against the CMap compound perturbations, loading and processing their z-scores at once requires 59.7 GiB, compared to 5.4 GiB when loading and processing 1 GiB chunks. The differences were not as stark when using the knockdown and over-expression perturbations, but were still crucial to allow the possibility of running cTRAP in a common laptop with 8 GiB of RAM.

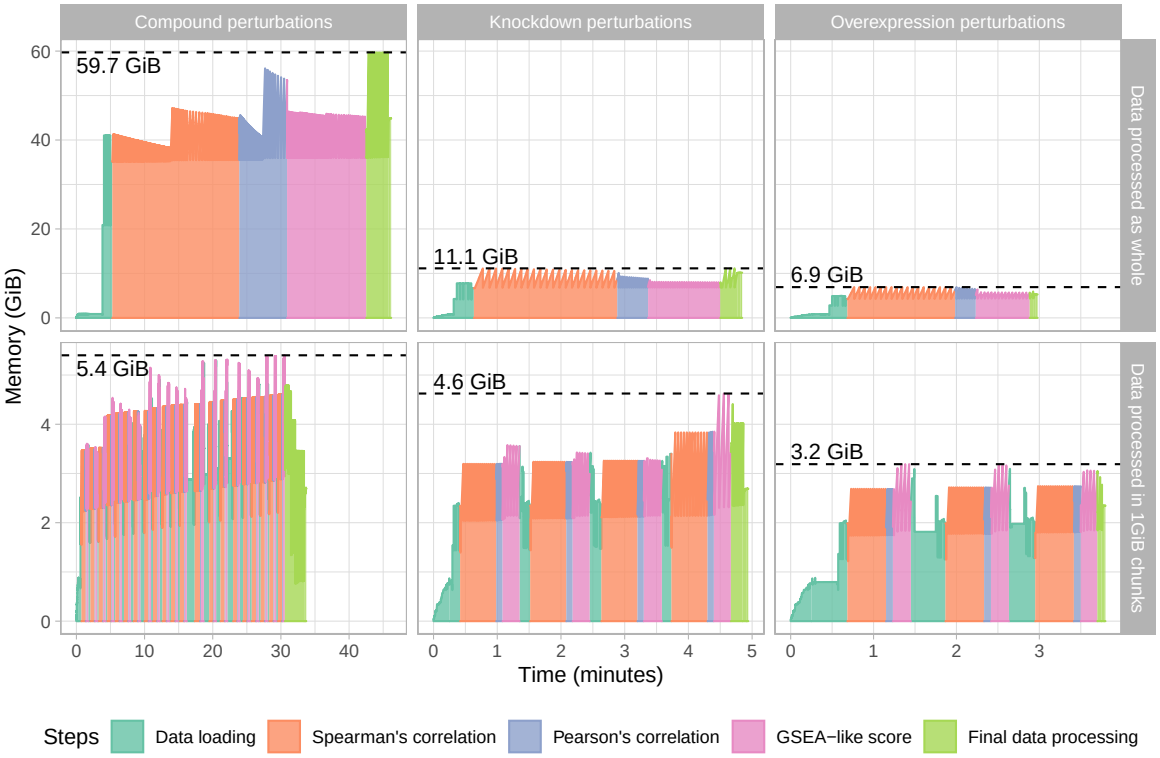


Figure 4.7: Memory benchmark of CMap perturbation ranking, annotated using the different steps performed by cTRAP. Memory was benchmarked when ranking compound, knockdown and overexpression perturbations against differential expression results. The maximum memory for each condition is labelled.

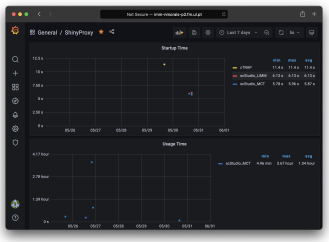
4.3.3 Graphical interface

To assist users that may prefer graphical interfaces, most cTRAP functionality is exposed via 5 modular and independent functions that launch web apps:

- `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.

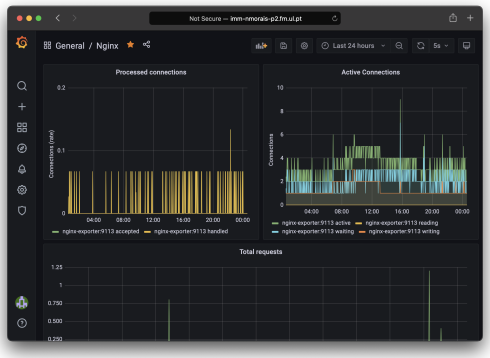
- `launchMetadataViewer()` to check metadata of given cTRAP objects.
- `launchResultPlotter()` to view and plot cTRAP results given as input.
- `launchDrugSetEnrichmentAnalyser()` to analyse drug set enrichment and visualize respective results.

(a) `launchDiffExprLoader`

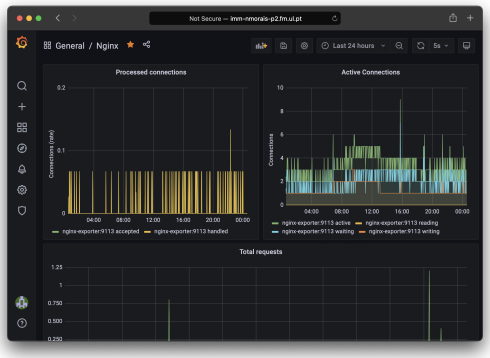


(b) `launchCMapDataLoader`

(c) `launchMetadataViewer`



(d) `launchResultPlotter`



(e) `launchDrugSetEnrichmentAnalyser`

Figure 4.8: cTRAP graphical interface functions.

Like usual R functions, these graphical interfaces functions accept input arguments and may return output, allowing to intertwine them with R code (Listing 4.2). Therefore, users can interactively prepare and explore data before running long-running tasks in the command-line, avoiding the need to have Shiny working while the tasks run (which may also slow down those tasks) and allowing to parallelise them.

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")
drugSets    <- prepareDrugSets(descriptors)

launchDrugSetEnrichmentAnalyser(drugSets, compareCompounds)
launchDrugSetEnrichmentAnalyser(drugSets, predicted)
```

cTRAP also provides a global visual interface that encompasses all graphical interface modules into one web app by running function `cTRAP()`, the basis for its online version[5]. However, by having a single interface, an entire cTRAP session needs to be live and consuming useful resources during the long-running cTRAP analyses, which does not properly scale in a web server with multiple users using heavy memory resources simultaneously. To avoid this, long-running tasks can be optionally managed via job queues running in the background. But this also means that the system must allow for users to get their results back once they finish calculating. And thus the idea

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

of user sessions was born.

User sessions

Session data are saved in folders named after a random alphanumeric string (token) that uniquely identifies each session and can be downloaded as RDS files. Given that the downloaded file is a list of all datasets from the user session, users can load these RDS files in any R session and in local instances of cTRAP. Downloading user sessions is encouraged because cTRAP session folders are removed from our server if not accessed in the last 30 days based on the access timestamp[6].

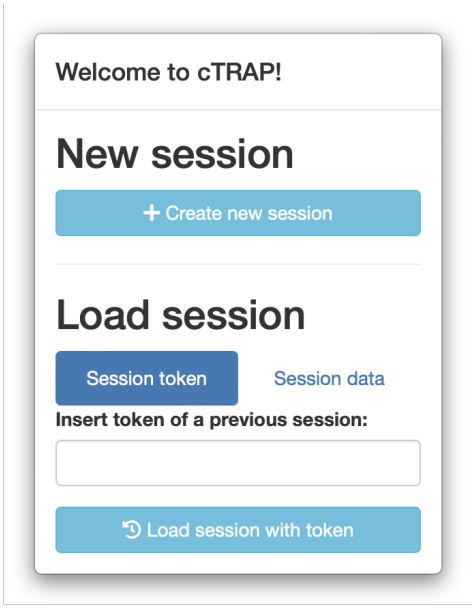


Figure 4.9: Welcome modal.

cTRAP visitors are greeted with a welcome screen that allows them to create a new session or restore previous ones (Figure 4.9), a dialog that can be opened at any time from the session menu.

When creating a new session, a unique a token is created. As soon as session-specific data is loaded, a new folder is created in the working directory and named after the session token (Figure 4.10). Any updates to the session data are automatically saved to the session folder. To avoid downloading commonly-used files (e.g. the 21GB CMap perturbations z-scores file), an appropriate folder stores data shared across sessions, thus avoiding downloading, storing and processing redundant data.

When restoring a session via a token, cTRAP loads the contents of the folder named after the token located in cTRAP working directory or warns the user if no such folder exists (Figure 4.10). In case the user uploads a RDS file of a previous session, cTRAP will load its contents in a new session (Figure 4.10).

Background tasks

When running a Shiny app, the user has to wait for the long-running tasks to finish before the app starts responding to user's commands again. Critically, if cTRAP is shut down, all running processes will stop.

This can be solved by using the R packages `promises/future` or by manually running another R process in the background. However, these solutions require the Shiny app to be active during the whole process, which can be especially egregious if the R session is consuming many computing resources, disallowing other apps or users

[6]In Linux, the access timestamp (`atime` attribute) for a directory indicates the last time a file within was read/written or its contents were listed.

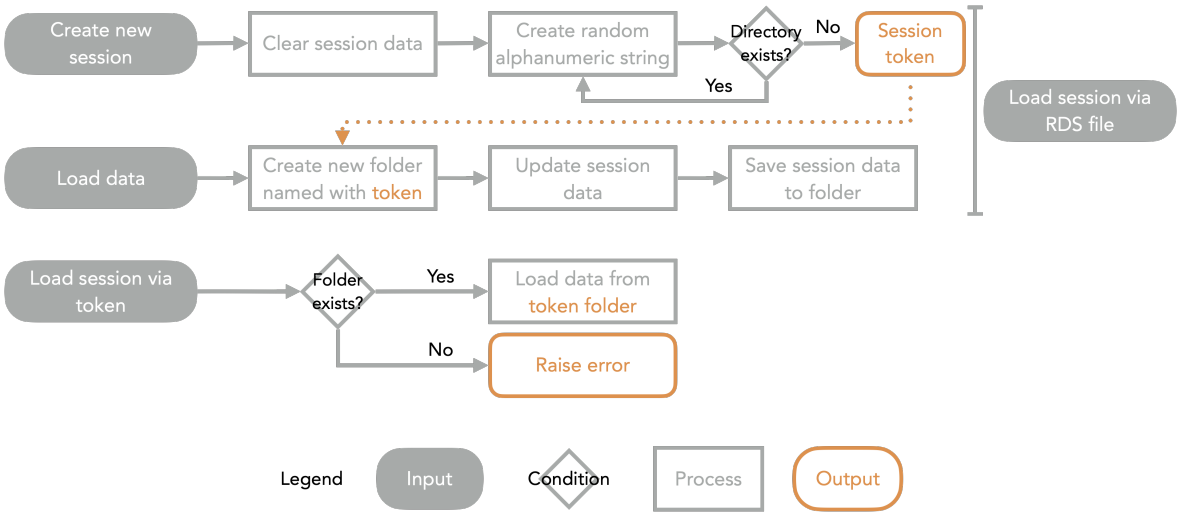


Figure 4.10: User session workflow. cTRAP allows to create new sessions or load a previous one via a given token or RDS file. When loading session via a RDS file, a new cTRAP session is created before loading the data from RDS file. Sessions can also be loaded using a token if any folder with such token exists in cTRAP's working directory.

to take advantage of those resources until the whole session is terminated.

What are the best options to run large tasks in the background? In Lobo, iMM's computing cluster, we use a job scheduler for that function: SLURM. However, SLURM is too complex to install – I was not able to create a working prototype of SLURM for a single machine using Docker or otherwise. Anyway, we can use these light-weight alternatives instead:

- **Celery**, a task queue manager written in Python; and
- **Flower**, a monitoring app that provides an HTTP API and graphical interface to manage Celery jobs.

Celery requires a `tasks.py` file with the code of the jobs to run. Given that Celery will run cTRAP functions, cTRAP must be installed alongside Celery. Moreover, as we intend to run R code, one of the Celery tasks is to run R via Python's module `subprocess` to spawn new processes with the `Rscript` command (Listing 4.3).

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

```
import os, time
from datetime import datetime
from subprocess import run, PIPE

# 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()
```

Flower can send jobs to Celery via HTTP methods, facilitating the communication between cTRAP and Celery. To assist using Flower in R, I created the R package `floweRy` (`github.com/nuno-agostinho/floweRy`) that contains wrapper functions for most of its HTTP API functions. Internally, `floweRy` calls HTTP methods with `httr` R package, creating dedicated commands that make it easier than using just plain `httr`, as briefly demonstrated in Listings 4.4 and 4.5.

Listing 4.4: Job submission with `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: Job submission with `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 by submitting jobs to Celery with the exact R commands to run via the `Rscript` command [200]. The status of background tasks can be monitored by users in the respective cTRAP's user session (Figure 4.11)[7].

All Celery jobs are saved as *dummy* objects in cTRAP's user session data, containing the job identifier and metadata from expected results. When the background

[7]Flower allows to monitor and manage Celery jobs, but its web interface is only accessible in the iMM network (via ethernet cable or VPN). More information in subsection 5.3.4: **Background tasks**.

Dataset	Progress
Ranked CMap compounds	Loaded
Ranked CMap knockdowns	Running
Ranked CMap overexpressions	Waiting
Ranked CMap knockdowns HepG2	Error
Ranked CMap perturbations MCF7	Not Found

Figure 4.11: Progress of Celery jobs in cTRAP. The job status are updated every 5 seconds. Their status can be: Waiting in job queue to start, Running, Loaded, Error for unknown failures, and Not Found if the job results cannot be found (e.g. when re-uploading the same RDS file, the job results were already cleaned-up in the first upload). When jobs are loaded, the dataset name is updated with a link to directly access their results (blue colour).

processes finish, their output is saved into the session folder. If the user is actively using that session in the cTRAP website, the data is automatically loaded – replacing the previous *dummy* objects – and the user is informed of such via a notification in cTRAP (Figure 4.12). Otherwise, next time that session is loaded by the user (either via its token or an RDS file), the job for every *dummy* object in the session data is returned if finished.

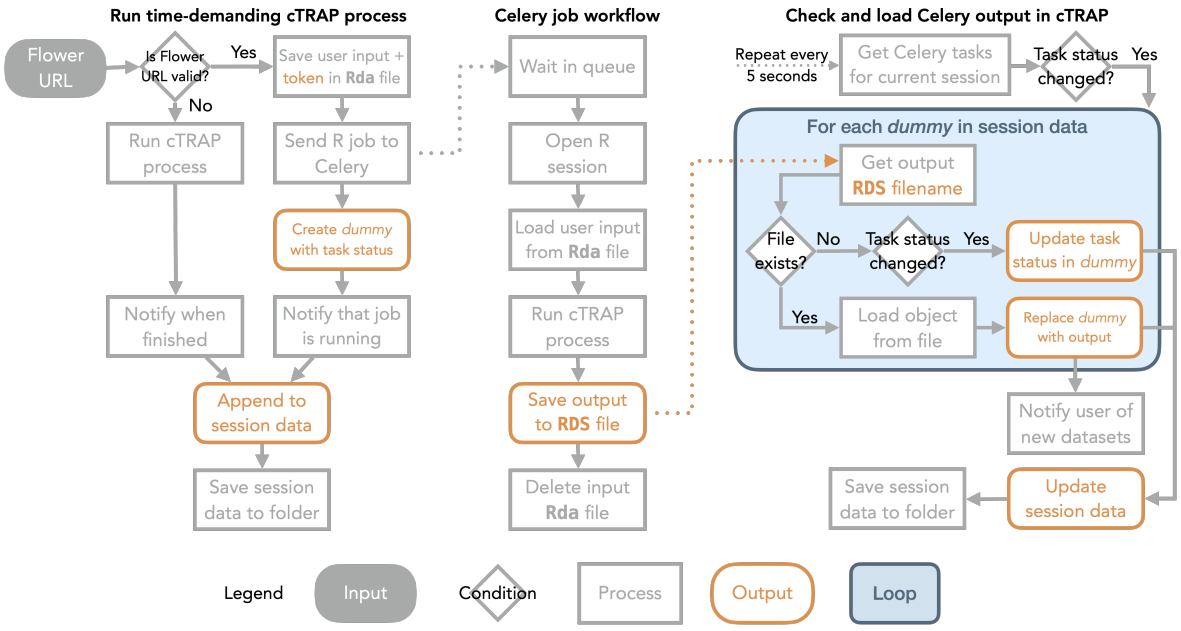


Figure 4.12: cTRAP process running in Celery. Time-demanding cTRAP processes can be run in the background using Celery/Flower. While running in Celery, the output of the cTRAP process is saved to the folder associated with the token of the user's session. When that specific session is active, all finished files are automatically loaded as part of the data session and the user is notified.

4.4 Conclusion

CMap is a repository of gene expression signatures for thousands of genetic and pharmacological perturbations that can be explored via a collection of web apps available at `clue.io`, which allows to upload user-provided results to compare against the CMap database. However, their queries are limited to a maximum of 150 up-regulated and 150 down-regulated genes, their results use a non-standard significance score and their apps do not run locally. Also, `clue.io` does not make use of publicly available drug sensitivity data that could help narrow down compounds that selectively target cells.

To overcome such issues, we developed cTRAP to identify causal perturbations from differential expression data, as well as pinpoint compounds that may promote or revert observed differences in gene expression. cTRAP allows to analyse the ordered list of resulting compounds to predict targeting drugs based on publicly available drug sensitivity data from NCI60, GDSC and CTRP. Moreover, cTRAP allows to perform enrichment analysis of molecular descriptors based on NCI60 and CMap compounds to potentially identify common characteristics in ordered lists of compounds.

Besides its command-line interface, most of cTRAP's functionality can be accessed via multiple graphical user interface functions that can be intertwined with R code. cTRAP also features a global user interface that combines all other visual interfaces and that is available as a web app at `compbio.imm.medicina.ulisboa.pt/cTRAP`. From our experience with psichomics, we expect the graphical interface to be popular among users that are less comfortable with coding in R.

Given the large input data from CMap, we optimised cTRAP for speed and memory usage. When comparing user-provided data against the 241 258 CMap compound perturbations in our benchmarks, cTRAP took 28 minutes to run in a single thread with a peak memory consumption of 5.4 GiB. This reduced memory consumption is the result of loading and processing CMap data in 1 GiB chunks that can be customised by users.

Regarding future cTRAP iterations, we aim to add support for CMap LINCS 2020, a CMap data expansion described as a *3-fold expansion on the previous resource, and [whose] notable new subsets of data include CRISPSR knockout of >5k genes and hematopoietic and non-cancer cell models* (`clue.io/data/CMap2020#LINCS2020`). However, this dataset is still in beta and requires some adjustments to cTRAP given the files are now provided individually per perturbation type.

Regarding the background task system, the web app could benefit from automatically emailing users when jobs finish (successfully or not). This would require to set up an email address to which to send emails from, preferentially from an official institutional account.

We hope that users will be able to successfully employ cTRAP in order to identify

candidate causal perturbations and compounds to better understand the biological mechanisms underlying differences in gene expression alterations, as well as prioritise targeted therapeutic agents for disease-associated queries.

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 the 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`.

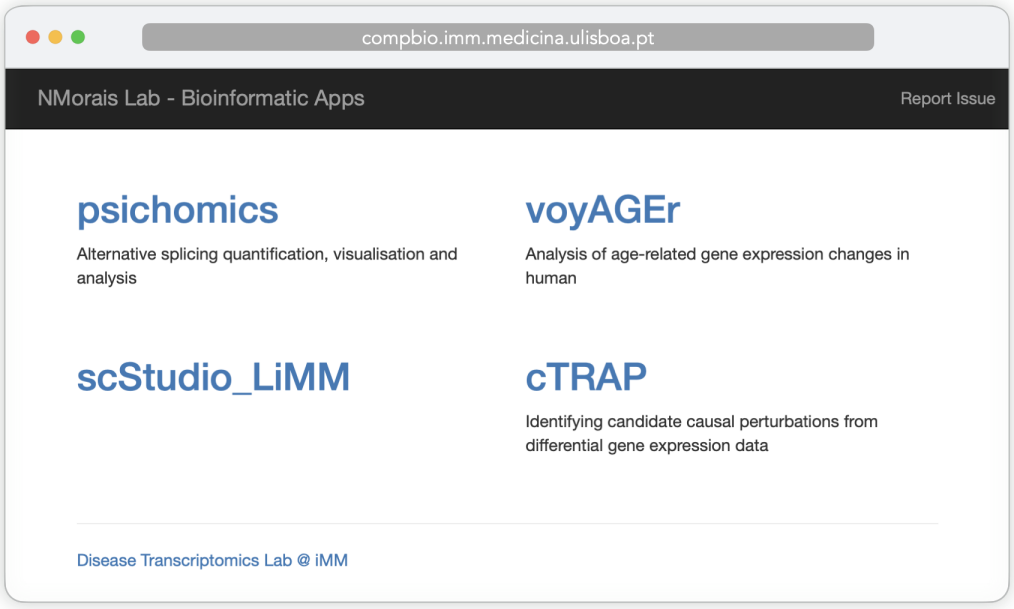


Figure 5.1: CompBio's homepage screenshot. List of hosted web apps (11 Nov 2021).

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 [145]. 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 [200, 145]. 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 [138]. 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 locally host Shiny apps:

- **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 Grafana), 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 services.

Role	Service	Port	Docker image[a]
Web app deployment	ShinyProxy	8080	`openanalytics/shinyproxy`
Reverse proxy	Nginx	443	`nginx`
Background tasks	Celery + cTRAP		Based on `nunoagostinho/ctrap`[b]
	Redis		`redis`
	Flower	5555	`mher/flower`
Website analytics (i.e. track visitor metrics)	Plausible	8000	`plausible/analytics`
	PostgreSQL	5432	`postgres`
	ClickHouse		`yandex/clickhouse-server`
Resource monitoring	Prometheus	9090	`prom/prometheus`
	Grafana	3000	`grafana/grafana`
	Nginx monitoring		`nginx/nginx-prometheus-exporter`
	System monitoring		`prom/node-exporter`
RStudio (testing)	RStudio Web[c]	8787	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.

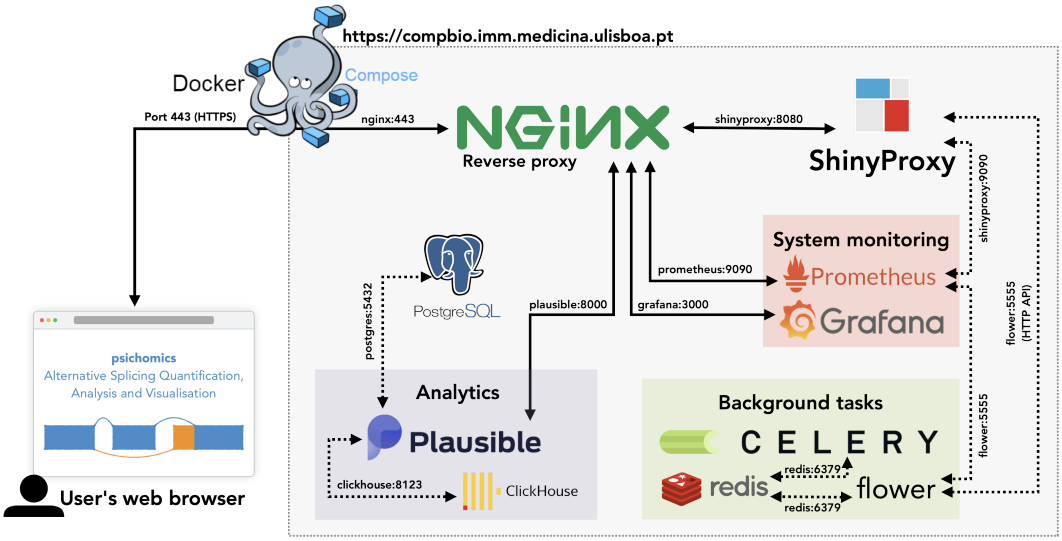


Figure 5.2: App server architecture is based on Docker Compose. All services are provided via Docker images and communicate with each other via a Docker-created network using the name of the service and a specific port (e.g. Nginx communicates with ShinyProxy via `shinyproxy:8080`). The groups (analytics,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

Services whose ports are listed in Table 5.1 may be accessed when connected via an ethernet cable at iMM or via the VPN of Universidade de Lisboa by using an internal HTTP (not HTTPS) URL specifying the service port. For instance, opening `http://imm-nmorais-p2.fm.ul.pt:8000` allows to access the Plausible dashboard[4].

5.3 Results

The CompBio app server was developed to be easily maintained and extended, allowing to add new and update existing Shiny apps and other modules. The server also supports running background processes[5], tracking simple visitor metrics (e.g. Shiny app usage time, number of visitors and user countries) and monitoring system usage. This project is open-source and free (`github.com/nuno-agostinho/compbio-app-server`) and the app server can be publicly accessed at `compbio.imm.medicina.ulisboa.pt`.

The app server 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 same server. The server itself is a virtual machine running in Lobo (iMM computing cluster) with 16 CPU threads, 64GB RAM and 200GB SSD. By exploiting a powerful infrastructure, the virtual machine can be manually modified to increase or decrease associated computing resources depending on system usage.

[4]If the web browser starts redirecting HTTP requests to HTTPS, the website should be accessed in private mode to avoid that behaviour, given that HTTPS disallows specifying ports.

[5]More information in subsection 5.3.4: **Background tasks**.

The code can be run on Linux[6] machines with Docker Compose installed, thus making the setup easily portable and requiring minimal user setup. Docker Compose also confers modularity and maintainability to the project, given that system components are easy to update and replace without affecting other components.

5.3.1 Docker Compose

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: 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 Docker containers. But how to coordinate several artists to beautifully perform the Swan Lake?

With the modularity from Docker Compose, multiple applications are run isolated in their own Docker containers, allowing to easily update or replace them without affecting other system components, as well as make the code of this project publicly available and portable. All services spawned in Docker Compose are based on Docker

[6] Although not the scope of this project, CompBio may be compatible with other operating systems.

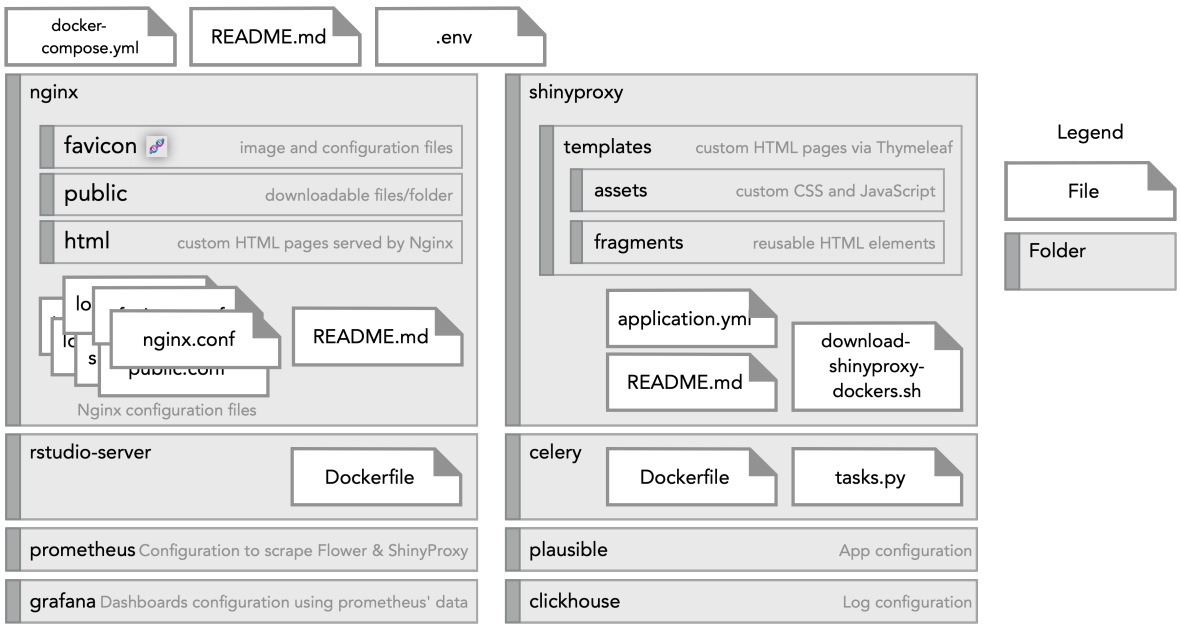


Figure 5.3: Visual representation of the file structure of the CompBio app server. Each folder contains files associated with a specific service. Folders `rstudio-server` and `celery` contain Dockerfiles for building custom Docker images of the respective services. Multiple `README.md` document the usage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

images that are either pre-created (e.g. downloaded from Docker Hub) or built on-demand – some services of this project have their own Dockerfiles, a recipe file to create custom Docker images.

For organisation purposes, the project is structured by folders named after each service, where each folder stores files associated with the respective application (e.g. Dockerfile, configuration and data; Figure 5.3). A single file named `docker-compose.yml` (Listing 5.1) contains the main configuration of each application in the server and extra configuration files are available in the local directory.

Listing 5.1: Shortened version of the `docker-compose.yml` file used for the project. This version only contains the configuration for Nginx and ShinyProxy.

```
version: "3.9"
services:
  nginx:
    image: nginx
    container_name: nginx
    restart: always
    ports:
      - 80:80
      - 443:443
    volumes:
      - ./nginx:/etc/nginx
      - /etc/ssl/imm:/certs:ro
      - ./nginx/public:/public:ro
    depends_on:
      - shinyproxy
  shinyproxy:
    image: openanalytics/shinyproxy:2.6.0
    container_name: shinyproxy
    restart: always
    ports:
      - 8080:8080
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./shinyproxy/application.yml:/opt/shinyproxy/application.yml
      - ./shinyproxy/templates:/opt/shinyproxy/templates:ro
      - shinyproxy-server:/log
      - shinyproxy-containers:/container-logs

  networks:
    default:
      name: shiny-net

  volumes:
    shinyproxy-server:
    shinyproxy-containers:
```

To start all the Docker Compose services, running the command

```
docker compose up -d --build
```

downloads Docker images in `docker-compose.yml`, builds Docker images from Dockerfiles and starts the services in detached mode.

Although data from Docker containers is temporarily stored while the container is running, specific files and directories can be preserved in Docker volumes to avoid data loss. When starting the `docker-compose.yml` project, Docker volumes are mounted for specific directories labelled volumes in Listing 5.1.

Docker Compose has multiple commands to manage the associated services. For instance, to apply new configurations, it is useful to restart a single service:

```
docker compose restart shinyproxy
```

To apply changes from `docker-compose.yml`, all services need to be restarted:

```
docker compose restart
```

Secrets

Most configuration files of the server are public, including default passwords that should only be used for testing purposes. To define sensitive information (i.e. secrets), all we need is to set custom information in a `.env` file at the root of the project directory (Listing 5.2). When starting the services, Docker Compose will replace the default environment variables from `docker-compose.yml` with those from `.env`.

Listing 5.2: Template of a `.env` file that defines sensitive data.

```
RSTUDIO_PASSWORD=rstudio_pass

POSTGRES_USER=postgres_user
POSTGRES_PASSWORD=postgres_pass

GRAFANA_USER=grafana_user
GRAFANA_PASSWORD=grafana_pass

PLAUSIBLE_EMAIL=someone@email.com
PLAUSIBLE_USER=plausible_user
PLAUSIBLE_PASSWORD=plausible_pass
```

Staging and production environment

The services in the app server (**production environment**) are live for the whole world to access. Any changes made to this server will be publicly seen by active users and should be avoided to also mitigate potential issues. Instead, changes should be tested in another system (**staging environment**), such as a personal computer in a testing

environment that resembles the production one. Preparing the staging environment is easy (Listing 5.3) and automatically performs the following:

- Creates a copy of the default Nginx and ShinyProxy configuration. This configuration files are not tracked by git and can be modified at will.
- Pulls and builds any Docker images used by Docker Compose and ShinyProxy.
- Modifies Nginx configuration to ignore SSL certificates. Nginx would throw an error otherwise because the SSL certificates only match the computer currently hosting the app server.
- Creates empty directories for web apps that may be populated with test data.

Listing 5.3: Setup testing environment.

```
# setup files for testing and download Docker images
./setup-testing-mode.sh
# start services and RStudio in detached mode
docker compose --profile dev up -d
```

The services should be fully operational in about 30 seconds after running these commands and accessible via `http://localhost` of the machine[7]. Some services are only available via their specific ports (Table 5.1), e.g. `http://localhost:8000` for Plausible and `http://localhost:8787` for RStudio.

Automated Testing

Testing is automatically performed via GitHub Actions. Every change to the GitHub repository is automatically checked to see if the command `docker compose up` works without throwing errors. In the future, automated testing could be extended to check specific functionalities of each service in the project, allowing to better understand if everything is working as expected following changes to the code.

5.3.2 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. ShinyProxy offers multiple built-in features, including:

[7] When using a remote machine, it is necessary to set up port forwarding via `SSH` to access the remote machine's localhost.

- **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[8].
- **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)[9].

Add and update web apps

Deploying new Shiny apps in the app server is as simple as making a Docker image available in Docker Hub, pulling it to the app server and then listing it in the ShinyProxy configuration. It is important to check first if the Shiny app can be launched via the Docker image in a local machine.

Afterwards, we add the configuration of the Docker image in the ShinyProxy configuration file (`shinyproxy/application.yml`; for instance, Listing 5.4), based on the available fields from ShinyProxy. The most important fields are `id`, `display-name` and `description` to identify an app, `container-image` to identify the associated Docker image and `container-cmd` to start up the Shiny app (although the command to start up the app can be included directly in the `Dockerfile` instead). If the app requires any data, volumes can be mounted using `container-volumes`. The `container-network` should remain `"$proxy.docker.container-network"` for all apps, given that it is required for proper communication between ShinyProxy and Docker Compose.

Listing 5.4: Simplified ShinyProxy configuration with `psichomics`.

```
proxy:
  title: NMorais Lab - Bioinformatic Apps
  template-path: /opt/shinyproxy/templates
  container-wait-time: 30000
  docker:
    internal-networking: true
    container-network: shiny-net
  specs:
  - id: psichomics
    description: Alternative splicing visualisation and analysis
    container-image: nunoagostinho/psichomics:1.18.6
```

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

```
    container-cmd: ["R", "-e",
      "psichomics::psichomics(host='0.0.0.0', port=3838)"]
    container-network: "${proxy.docker.container-network}"
    container-volumes: [ "/srv/apps/psichomics/data:/root/Downloads" ]
    template-properties:
      startup-time: 15s
      listed: true
```

Custom properties (`template-properties`) are also set for this project, including whether an app should be publicly listed (`listed`) and a rough estimate of its startup time to show a progress bar to visitors (`startup-time`). These custom properties are described in more detail ahead.

After defining this script, we only need to restart the ShinyProxy with `docker compose restart shinyproxy` and any configured apps will be available for use. Updating an app is as easy editing `shinyproxy/application.yml` with the most recent Docker version and pulling that version to the app server, before restarting ShinyProxy.

Custom HTML pages

Custom HTML pages are located in folder `shinyproxy/templates`. ShinyProxy uses custom files located there if available, falling back to its own default files otherwise. In other words, to get the original ShinyProxy behaviour for the default HTML pages, we only need to remove the files from that folder and restart ShinyProxy.

The HTML pages provided by ShinyProxy are based on the Thymeleaf template engine that uses HTML-like code scripting. Directly editing HTML pages provided by ShinyProxy allows to add the custom features described in the following subsections, as well as custom error pages (e.g. 404 page not found or issues when starting containers).

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 [201, 202].

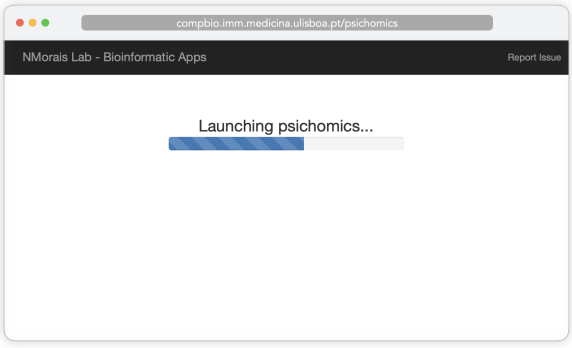


Figure 5.4: Progress bar displayed while psichomics loads (11 Nov 2021).

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 need to be perfect to serve its purpose [201, 202].

To create this progress bar, `shinyproxy/templates/app.html` was edited to remove the spinning wheel and to include an empty progress bar. The progress bar's width is changed from 0% to 100% using JavaScript. By default, the CSS width transition applied to the progress bar is `transition: width 5s ease-in-out;` (animating a change of width that last for 5 seconds in an ease-in animation) where `5s` is replaced by the `template-properties.startup-time` parameter if set.

Private web apps

In the website's landing page (Figure 5.1), ShinyProxy lists all apps described in the configuration file by default (`shinyproxy/application.yml`). This may not be desired when hosting apps with confidential results to be shared with specific collaborators. For this reason, we added the key `template-properties.listed` that can either be `false` (default) or `true`. The file `shinyproxy/templates/index.html` was edited to show only apps whose `template-properties.listed` key is set to `true`. Thus, non-listed web apps are not displayed in the landing page, but are still directly accessible via URL based on their app ID, e.g. `compbio.imm.medicina.ulisboa.pt/app/psichomics`.

However, if the information contained in the web app should not be accessible to strangers at all, apps can also implement a password input form (e.g. `shiny::passwordInput()`) in the code itself before loading any data and/or information. That password should be securely shared with the intended audience only.

5.3.3 Nginx

Nginx is a reverse proxy, i.e. an intermediary that controls what is shown to the user depending on the URL visited – akin to those switchboard operators seen in old movies. In CompBio, Nginx fulfills user requests and performs many other functions:

- **Ensure HTTPS traffic is encrypted via SSL certificates** from the IT team at iMM. We simply point to the correct location of those certificates.
- **Serve publicly available files** in the `nginx/public` folder, whose directory structure is accessible at `https://compbio.imm.medicina.ulisboa.pt/public/`.

- **Show a custom error page if ShinyProxy is not responding** (e.g. temporarily down or overloaded). When ShinyProxy is down, Nginx informs end-users to retry refreshing the page and that ShinyProxy is probably down, informing end-users to retry refreshing the page. ShinyProxy can be down for multiple reasons, such as during a restart or due to resource overloading.
- **Display the website favicons** stored in folder `nginx/favicon`.

5.3.4 Background tasks

In our app server, we use Celery to run background tasks, alongside Flower to manage Celery jobs via its graphical interface and HTTP API[10]. We also use the Redis broker to communicate between the two Docker containers.

Currently, cTRAP is the only web app in our server that exploits background tasks. We built a Docker image based on the official cTRAP Docker image (`nunoagostinho/ctrap`) with the Celery app installed on top.

Celery was configured to use 3 to 10 processes based on demand, as well as CPU and memory usage via an independent plugin (`github.com/jcushman/celery-resource-autoscaler`). For instance, each process in Celery can use up to 10GiB of RAM and run up to 12 hours, thus avoiding rogue tasks.

To run other programs in Celery, we need to create a custom Dockerfile containing those programs (e.g. based on their Docker images) with Python and Celery installed. The Nginx configuration needs to include the new Celery service (Listing 5.5).

Listing 5.5: Configuration of the Celery service for cTRAP in `docker-compose.yml`.

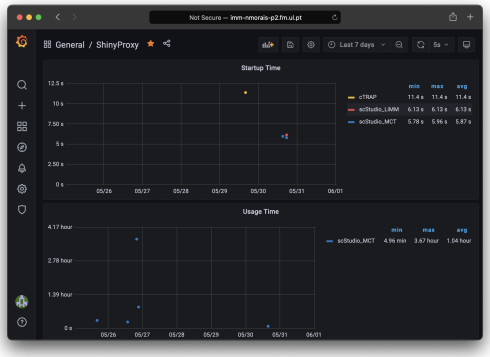
```
celery-ctrap:
  container_name: celery-ctrap
  build: ./celery
  command: celery -A tasks worker -c5 -l info -E -n ctrap
  volumes:
    - ./celery:/celery:ro
    - ../apps/cTRAP/sessions:/data
  depends_on:
    - redis
```

5.3.5 Resource monitoring

Prometheus monitors the server resources and the collected data can be visualised using Grafana (Figure 5.5). The tracked resources include Celery job usage, ShinyProxy metrics (app usage time, app failures, users per app, etc.), Nginx status and Linux system resources (e.g. RAM usage, available disk space and CPU stress).

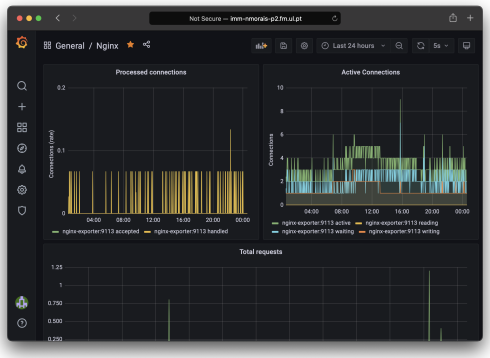
[10] More information in section 4.3.3: **Background tasks**.

(a) Linux system metrics.



(b) ShinyProxy metrics.

(c) Celery metrics.



(d) Nginx metrics.

Figure 5.5: Grafana dashboards showing tracked metrics (1 Jun 2021).

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 (Figure 5.6). 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 of users.

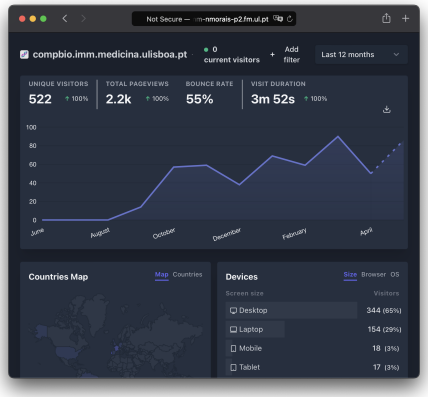


Figure 5.6: Plausible dashboard showing CompBio website analytics for the last year (as of 31 May 2021).

Plausible uses the database management systems ClickHouse and PostgreSQL to store tracking data. PostreSQL can also be used in the future as the SQL database of the server if desired, although currently no web apps in the server use this database.

Using the self-hosted version of Plausible guarantees that the tracking of user data is performed locally in the server. Plausible also protects user privacy by making their data hard to individually trace and by complying with current privacy laws.

5.3.7 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 its components, including Docker, Docker Compose, Nginx and ShinyProxy. 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[11]. On the other hand, Docker images of the server (including Nginx and ShinyProxy) 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[12]. The advantage of using Docker Compose is that if something goes wrong with the updated Docker images, we simply need to revert `docker-compose.yml` to a previous working state and restart all services.

5.4 Conclusion

The CompBio app server was developed to host web apps from NMorais Lab using ShinyProxy and Docker Compose, allowing to easily add new or update existing Shiny apps containerised via Docker. It also contains multiple components to run background cTRAP tasks, track app usage and monitor computing resources.

CompBio currently runs in a virtual machine in Lobo, iMM computing cluster. The hardware is 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 update. Components can be manually updated by simply editing the intended version in `docker-compose.yml` and restarting all the services. In case of issues, it is easy to rollback to a stable, working version of the app server based on previously used Docker images. Testing new changes to the server can be performed using the staging mode, allowing to mirror the app server and test changes locally before pushing them live to the app server.

The project also makes uses of Nginx as a reverse proxy. An issue with using Nginx is that it is especially verbose compared to more recent reverse proxies. Although I would have liked to replace Nginx with a simpler reverse proxy – such as Caddy (`caddyserver.com`) –, Nginx is more popular and widely used, thus making it easier to find documentation and to search for issues.

In the future, we can adapt available computing resources of our virtual machine as needed. In case we prefer to port the app server to a new machine, as the project was

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

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

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.

By publicly hosting the project code in GitHub, we hope to demonstrate the flexibility of setting up Docker Compose to other labs and entities, promoting an easily portable, reproducible and documented configuration of a Shiny web app server that can facilitate sharing public apps among the scientific community and beyond.

Chapter 6

Discussion

Continuing the path I started walking during my MSc, my PhD gave me the opportunity to further develop web apps to be used by the scientific community and to enrich my knowledge about app development and explore multiple new technologies.

6.1 psichomics

Since starting to work in psichomics, I struggled with the lack of guidelines on how to properly design and test interactive bioinformatic web apps. Although there are resources to build generic web apps, the field of bioinformatics could be richer if we better understood how researchers and clinicians explore data to help them find what they are looking for. Also, the lack of a systematic approach to app design is notable in multiple bioinformatics programs in the wild, reflected by popular apps lacking in efficiency and usability, as well as abandoned programs due to completed and/or unrenewed grants.

Ultimately, I think that bioinformaticians that want to create apps to be adopted by the scientific community should be aware of software design to properly write apps following requirement analysis and that are designed for the long-term; data visualisation to design interactive plots that intuitively convey the desired information and allow to conveniently explore the data; and user interface design to improve the user experience and unleash the full potential of the software functionality.

psichomics was the most challenging program I ever created and the project of my lifetime. It allowed me to develop an app based on multiple topics of my interest, including transcriptomics, data visualisation, web technologies and user interface design. The positive user interaction I received, some of which have led to citations in scientific articles, and the invitation to write the methods book chapter following the publication of psichomics were uplifting and made me feel that psichomics is a good contribution to the field and I can only hope that it stays relevant and useful in the near future.

6.2 cTRAP

Thanks to working on psichomics, my knowledge of R was more mature while developing cTRAP, which allowed me to flourish my creativity. In the optimisation department, it was challenging to monitor memory usage and to improve cTRAP's runtime, reduce peak memory usage and add multicore support in order to properly perform cTRAP functions as efficiently as possible. This lead to rewriting a lot of the core code in cTRAP 1.4 (subsection 4.3.2: **Time and memory optimisation**).

Regarding its user interface, we created graphical interface functions that can be intertwined with R code, which I found a fun experiment. Although their practical usefulness may be limited, their modular design made it relatively easy to create an intuitive global interface that brought life to the cTRAP web app.

Creating the web app itself with support for background tasks via Celery and user sessions was also demanding and required a lot of experimentation to design the current implementation, including the creation of the floweRy R package to interact with Flower/Celery, testing the whole integration in the web server and to make all the code and documentation available in cTRAP so users can host their own servers with background task and user session support. Nevertheless, the web app could benefit from email support to improve the user experience, but it is not a trivial functionality as Celery does not have built-in support for sending emails.

6.3 CompBio

Creating the app server to host web apps was an enormous challenge that I am completely proud of. With the exception of R/RStudio and Docker, I had to learn about all of the software stack that I used (Docker Compose, ShinyProxy, Nginx, Celery[1], Plausible, Prometheus, Grafana, etc.) and how these pieces interact amongst each other to properly fine-tune the server to our needs.

6.4 PanAShé

There are many other projects that I have been part of during my PhD but ultimately did not lead to much. However, there is still one that I was a part of and I hope my colleagues will be able to complete: PanAShé.

In a collaborative lab effort, we are developing a Nextflow pipeline to process raw RNA sequencing data from TCGA [6] and GTEx [7] in order to provide processed gene expression and alternative splicing data from samples from multiple normal and

[1] I started developing the app server at the same time I was researching how to run background tasks for cTRAP. I decided on Celery only after confirming it worked with the app server.

diseased tissues. The aims of this project extend those of recount2 [8] 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.

All the software stack in PanAShé is based on Docker images for portability and reproducibility. This means that only Docker and Nextflow are required to run 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.

6.5 Conclusion

With the work I hereby present, I hope to provide researchers and clinicians with useful tools to analyse gene expression and alternative splicing, predict therapeutic drugs and deploy web apps. Amongst my personal objectives to do a PhD, I have successfully completed one thanks to psichomics: to create something useful to someone's research. I can only hope that the rest of my work is as successful as psichomics has been in contributing to science.

As small as all my contributions may have been, these last 4 years were worthy for the prospect of having a (tiny little bit) part in helping unraveling the biological mysteries of this world, along with everything I learned and all the friends I made and danced with along the way.

Bibliography

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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`.

[11] Coomer AO, Black F, Greystoke A, Munkley J, Elliott DJ. Alternative splicing in lung cancer. Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms. 2019;1862(11):194388. doi:https://doi.org/10.1016/j.bbagrm.2019.05.006.

[12] Baeza-Centurion P, Miñana B, Schmiedel JM, Valcárcel J, Lehner B. Combinatorial Genetics Reveals a Scaling Law for the Effects of Mutations on Splicing. Cell. 2019;176(3):549–563.e23. doi:https://doi.org/10.1016/j.cell.2018.12.010.

[13] Munkley J, Li L, Krishnan SRG, Hysenaj G, Scott E, Dalgliesh C, et al. Androgen-regulated transcription of *ESRP2* drives alternative splicing patterns in prostate cancer. eLife. 2019;8:e47678. doi:10.7554/eLife.47678.

[14] Baeza-Centurion P, Miñana B, Valcárcel J, Lehner B. Mutations primarily alter the inclusion of alternatively spliced exons. eLife. 2020;9:e59959. doi:10.7554/eLife.59959.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

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

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

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

[25] 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.

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

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

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

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

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

[31] 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`.

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

[33] 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.

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

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

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

[37] 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.

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

[39] 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.

[40] Lederberg J, Tatum EL. Gene Recombination in Escherichia Coli. Nature. 1946;158(4016):558–558. doi:10.1038/158558a0.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] Crick FHC. On protein synthesis. Symp Soc Exp Biol. 1958;12:138–163.

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

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

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

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

[66] 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.

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

[68] 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.

[69] 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.

[70] 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.

[71] 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.

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

[73] 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.

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

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] 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.

[80] 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.

[81] 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.

[82] 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.

[83] Zhang Y, Qian J, Gu C, Yang Y. Alternative splicing and cancer: a systematic review. Signal Transduction and Targeted Therapy. 2021;6(1):78. doi:10.1038/s41392-021-00486-7.

[84] 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.

[85] Sanger F, Thompson EO, Kitai R. The amide groups of insulin. The Biochemical journal. 1955;59(3):509–518. doi:10.1042/bj0590509.

[86] 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.

[87] 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.

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

[89] 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`.

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

[91] 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.

[92] 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.

[93] 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.

[94] 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.

[95] 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.

[96] 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.

[97] 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.

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

[99] 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.

[100] 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.

[101] 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.

[102] Fleischmann RD, Adams MD, White O, Clayton RA, Kirkness EF, Kerlavage AR, et al. Whole-Genome Random Sequencing and Assembly of Haemophilus influenzae Rd. Science. 1995;269(5223):496–512. doi:10.1126/science.7542800.

[103] Goffeau A, Barrell BG, Bussey H, Davis RW, Dujon B, Feldmann H, et al. Life with 6000 Genes. Science. 1996;274(5287):546–567. doi:10.1126/science.274.5287.546.

[104] The C elegans Sequencing Consortium. Genome Sequence of the Nematode C. elegans: A Platform for Investigating Biology. Science. 1998;282(5396):2012–2018. doi:10.1126/science.282.5396.2012.

[105] Myers EW, Sutton GG, Delcher AL, Dew IM, Fasulo DP, Flanigan MJ, et al. A whole-genome assembly of Drosophila. Science. 2000;287(5461):2196–2204. doi:10.1126/science.287.5461.2196.

[106] Adams MD, Celniker SE, Holt RA, Evans CA, Gocayne JD, Amanatides PG, et al. The genome sequence of Drosophila melanogaster. Science. 2000;287(5461):2185–2195. doi:10.1126/science.287.5461.2185.

[107] The Arabidopsis Genome Initiative. Analysis of the genome sequence of the flowering plant Arabidopsis thaliana. Nature. 2000;408(6814):796–815. doi:10.1038/35048692.

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

[109] 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.

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

[111] 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.

[112] 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.

[113] Adams MD, Kelley JM, Gocayne JD, Dubnick M, Polymeropoulos MH, Xiao H, et al. Complementary DNA Sequencing: Expressed Sequence Tags and Human Genome Project. Science. 1991;252(5013):1651–1656. doi:10.1126/science.2047873.

[114] Velculescu VE, Zhang L, Vogelstein B, Kinzler KW. Serial Analysis of Gene Expression. Science. 1995;270(5235):484–487. doi:10.1126/science.270.5235.484.

[115] Schena M, Shalon D, Davis RW, Brown PO. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. Science. 1995;270(5235):467–470. doi:10.1126/science.270.5235.467.

[116] Lashkari DA, DeRisi JL, McCusker JH, Namath AF, Gentile C, Hwang SY, et al. Yeast microarrays for genome wide parallel genetic and gene expression analysis. Proceedings of the National Academy of Sciences of the United States of America. 1997;94(24):13057–13062. doi:10.1073/pnas.94.24.13057.

[117] Nagalakshmi U, Wang Z, Waern K, Shou C, Raha D, Gerstein M, et al. The transcriptional landscape of the yeast genome defined by RNA sequencing. Science (New York, NY). 2008;320(5881):1344–1349. doi:10.1126/science.1158441.

[118] Johnson WE, Li C, Rabinovic A. Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics. 2006;8(1):118–127. doi:10.1093/biostatistics/kxj037.

[119] Zhang Y, Parmigiani G, Johnson WE. ComBat-seq: batch effect adjustment for RNA-seq count data. NAR Genomics and Bioinformatics. 2020;2(3). doi:10.1093/nargab/lqaa078.

[120] Andrews S, Krueger F, Segonds-Pichon A, Biggins L, Krueger C, Wingett S. FastQC; 2019. Babraham Institute.

[121] 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.

[122] 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.

[123] 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.

[124] 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.

[125] 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.

[126] 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.

[127] 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.

[128] Perez-Riverol Y, Zorin A, Dass G, Vu MT, Xu P, Glont M, et al. Quantifying the impact of public omics data. Nature Communications. 2019;10(1):3512. doi:10.1038/s41467-019-11461-w.

[129] 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.

[130] 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.

[131] Goldman MJ, Craft B, Hastie M, Repečka K, McDade F, Kamath A, et al. Visualizing and interpreting cancer genomics data via the Xena platform. Nature Biotechnology. 2020;38(6):675–678. doi:10.1038/s41587-020-0546-8.

[132] 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.

[133] 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.

[134] 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.

[135] 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.

[136] Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. Nature Biotechnology. 2017;35(4):316–319. doi:10.1038/nbt.3820.

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

[138] 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.

[139] 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.

[140] 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`.

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

[142] 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`.

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

[144] 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.

[145] 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`.

[146] 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.

[147] 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.

[148] 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.

[149] Reyes ALP, Silva TC, Coetzee SG, Plummer JT, Davis BD, Chen S, et al. GENAVi: a shiny web application for gene expression normalization, analysis and visualization. BMC Genomics. 2019;20(1):745. doi:10.1186/s12864-019-6073-7.

[150] Cardoso-Moreira M, Halbert J, Valloton D, Velten B, Chen C, Shao Y, et al. Gene expression across mammalian organ development. Nature. 2019;571(7766):505–509. doi:10.1038/s41586-019-1338-5.

[151] 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`.

[152] 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.

[153] 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.

[154] 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.

[155] 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.

[156] 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.

[157] 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.

[158] 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.

[159] 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.

[160] 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.

[161] 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.

[162] 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.

[163] 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.

[164] 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.

[165] 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.

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

[167] 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.

[168] 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.

[169] 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.

[170] Therneau T, Grambsch P. Modeling Survival Data: Extending The Cox Model. vol. 48. New York, NY: Springer; 2000.

[171] 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.

[172] 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.

[173] 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.

[174] 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.

[175] 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.

[176] 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.

[177] 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.

[178] 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.

[179] 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.

[180] 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.

[181] 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.

[182] 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.

[183] 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.

[184] Zavolan M, Kanitz A. RNA splicing and its connection with other regulatory layers in somatic cell reprogramming. Current Opinion in Cell Biology. 2018;52:8–13. doi:https://doi.org/10.1016/j.ceb.2017.12.002.

[185] Han H, Irimia M, Ross PJ, Sung HK, Alipanahi B, David L, et al. MBNL proteins repress ES-cell-specific alternative splicing and reprogramming. Nature. 2013;498(7453):241–245. doi:10.1038/nature12270.

[186] Venables JP, Lapasset L, Gadea G, Fort P, Klinck R, Irimia M, et al. MBNL1 and RBFOX2 cooperate to establish a splicing programme involved in pluripotent stem cell differentiation. Nature Communications. 2013;4(1). doi:10.1038/ncomms3480.

[187] Chen K, Dai X, Wu J. Alternative splicing: An important mechanism in stem cell biology. World journal of stem cells. 2015;7(1):1–10. doi:10.4252/wjsc.v7.i1.1.

[188] Gabut M, Samavarchi-Tehrani P, Wang X, Slobodeniuc V, O'Hanlon D, Sung HK, et al. An Alternative Splicing Switch Regulates Embryonic Stem Cell Pluripotency and Reprogramming. Cell. 2011;147(1):132–146. doi:10.1016/j.cell.2011.08.023.

[189] Pradella D, Naro C, Sette C, Ghigna C. EMT and stemness: flexible processes tuned by alternative splicing in development and cancer progression. Molecular Cancer. 2017;16(1). doi:10.1186/s12943-016-0579-2.

[190] Aponte PM, Caicedo A. Stemness in Cancer: Stem Cells, Cancer Stem Cells, and Their Microenvironment. Stem Cells International. 2017;2017:1–17. doi:10.1155/2017/5619472.

[191] Choi J, Lee S, Mallard W, Clement K, Tagliazucchi GM, Lim H, et al. A comparison of genetically matched cell lines reveals the equivalence of human iPSCs and ESCs. Nature Biotechnology. 2015;33(11):1173–1181. doi:10.1038/nbt.3388.

[192] Fagoonee S, Bearzi C, Di Cunto F, Clohessy JG, Rizzi R, Reschke M, et al. The RNA Binding Protein ESRP1 Fine-Tunes the Expression of Pluripotency-Related Factors in Mouse Embryonic Stem Cells. PLOS ONE. 2013;8(8):1–13. doi:10.1371/journal.pone.0072300.

[193] Warzecha CC, Jiang P, Amirikian K, Dittmar KA, Lu H, Shen S, et al. An ESRP-regulated splicing programme is abrogated during the epithelial–mesenchymal transition. The EMBO Journal. 2010;29(19):3286–3300. doi:https://doi.org/10.1038/emboj.2010.195.

[194] Qiu W, Chavarro J, Lazarus R, Rosner B, Ma J. powerSurvEpi: Power and Sample Size Calculation for Survival Analysis of Epidemiological Studies; 2021. Available from: `https://CRAN.R-project.org/package=powerSurvEpi`.

[195] 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.

[196] 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.

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

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

[199] Enache OM, Lahr DL, Natoli TE, Litichevskiy L, Wadden D, Flynn C, et al. The GCTx format and cmap
Py, R, M, J
packages: resources for optimized storage and integrated traversal of annotated dense matrices. Bioinformatics. 2018;35(8):1427–1429. doi:10.1093/bioinformatics/bty784.

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

[201] Myers BA. The Importance of Percent-Done Progress Indicators for Computer-Human Interfaces. SIGCHI Bull. 1985;16(4):11–17. doi:10.1145/1165385.317459.

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