UNIVERSIDADE DE LISBOA
Faculdade de Medicina

Developing Web Apps for Analyses of Transcriptomes

Nuno Daniel Saraiva Agostinho

Orientadores: Prof. Doutor Nuno Luís Barbosa Morais
Prof. Doutor Sérgio Alexandre Fernandes de Almeida

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

2023

UNIVERSIDADE DE LISBOA
Faculdade de Medicina

Developing Web Apps for Analyses of Transcriptomes

Nuno Daniel Saraiva Agostinho

Orientadores: Prof. Doutor Nuno Luís Barbosa Morais
Prof. Doutor Sérgio Alexandre Fernandes de Almeida

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

Júri:

Presidente: Doutor Mário Nuno Ramos de Almeida Ramirez, Professor Associado com Agregação e Vice-Presidente do Conselho Científico da Faculdade de Medicina da Universidade de Lisboa

Vogais: PhD Manuel Irimia, Group Leader e Research Professor do Centre for Genomic Regulation (Espanha)

Doutora Ana Rita Fialho Grosso, Professora Auxiliar da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

PhD Alejandro Reyes, Especialista de Reconhecido Mérito e Competência, Principal Scientist do Novartis Institutes for BioMedical Research (Switzerland)

Doutora Margarida Henriques da Gama Carvalho, Professora Auxiliar da Faculdade de Ciências da Universidade de Lisboa

Doutor Nuno Luis Barbosa Morais, Professor Associado Convidado da Faculdade de Medicina da Universidade de Lisboa (Orientador)

Instituição Financiadora: Fundação para a Ciência e Tecnologia
(SFRH/BD/131312/2017 e COVID/BD/151928/2021)

2023

As opiniões expressas nesta publicação são
da exclusiva responsabilidade do seu autor.

A impressão desta tese foi aprovada pelo Conselho Científico da Faculdade de Medicina de Lisboa em reunião de 15 de novembro de 2022.

Contents

Resumo

Durante o meu doutoramento no Instituto de Medicina Molecular João Lobo Antunes (iMM), desenvolvi aplicações *web* na área da transcritómica como recursos gratuitos e de código aberto disponíveis à comunidade científica, que tenho melhorado e mantido frequentemente. Estas 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 estão alojados no Bioconductor, um repositório de pacotes de R para análise de dados biológicos.

psichomics

O *splicing* alternativo é um mecanismo molecular que permite gerar vários transcritos funcionais a partir do mesmo gene e está envolvido no controlo de múltiplos processos celulares, sendo a sua desregulação associada a várias patologias. A sua relevância biológica e os avanços na tecnologia de sequenciação de RNA têm levado a um maior interesse no seu estudo e na publicação de dados de *splicing* alternativo [1–5]. Este tipo de dados tem sido disponibilizado em acesso livre por vários projectos científicos, incluindo o The Cancer Genome Atlas (TCGA), que cataloga dados clínicos e moleculares de múltiplos tumores humanos [6]; o Genotype-Tissue Expression (GTEx), que se foca em dados de múltiplos tecidos humanos normais [7]; e o recount2, que processou dados de mais de 2000 estudos oriundos do Sequence Read Archive (SRA) [8].

A utilização dos dados pré-processados provenientes destes projectos evita o armazenamento e processamento dos ficheiros com dados de sequenciação *em bruto*, facilitando a análise de expressão génica e *splicing* alternativo ao poupar tempo e recursos computacionais dispendiosos no processamento de dados. Também é importante notar que dados humanos não processados podem ter acesso limitado e requerer um pedido de autorização para acesso total por razões de privacidade dos dadores. No entanto, até 2018, nenhuma ferramenta permitia realizar uma análise diferencial de *splicing* alternativo com base em dados transcritómicos pré-processados descarregados destas fontes públicas e com agrupamento das amostras a partir dos seus metadados.

Assim, desenvolvi a ferramenta psichomics para quantificar, analisar e visualizar *splicing* alternativo, inicialmente a partir de dados pré-processados provenientes do

TCGA [6]. Através da sua interface gráfica intuitiva ou da linha de comandos em R, o psichomics permite ao utilizador realizar análises de *splicing* e expressão diferencial, de componentes principais, e de sobrevivência com incorporação de características moleculares e clínicas das amostras consideradas.

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

Após a publicação do artigo relativo ao psichomics em 2018 [9], fomos convidados a escrever um capítulo de métodos no livro *Stem Cell Transcriptional Networks*, publicado em 2020, no qual exemplificamos a utilização do psichomics para analisar *splicing* alternativo no contexto de células estaminais humanas [10]. Desde a sua publicação, o psichomics foi usado em projectos associados a vários artigos científicos [11–14]. Acreditamos que estas citações, juntamente com os comentários positivos que temos vindo a receber dos utilizadores, demonstram que vários investigadores 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 contextos fisiológico e de doença.

cTRAP

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

O CMap and LINCS Unified Environment (`clue.io`) foi desenvolvido como um conjunto de ferramentas intuitivas para explorar os dados do CMap e integrá-los com dados disponibilizados pelo utilizador [15]. No entanto, o `clue.io` limita o número máximo de genes considerados para comparar com o CMap, expressa os resultados através de um valor de significância não convencional (e, portanto, de interpretação não imediata), 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 aos compostos, que podem auxiliar na identificação de compostos que afectem apenas células específicas e que tenham genes específicos como alvos moleculares.

Para colmatar estas lacunas, desenvolvemos o cTRAP para identificar perturbações moleculares causais potenciais ao comparar resultados de expressão génica diferencial providenciados pelo utilizador com os do CMap, assim como prever compostos que

promovam ou revertam as diferenças observadas. O cTRAP também permite comparar aqueles resultados com associações entre expressão génica e sensibilidade aos compostos derivados do NCI-60 [16], do Cancer Therapeutics Response Portal (CTRP) [17] e do Genomics of Drug Sensitivity in Cancer (GDSC) [18], para identificar compostos que podem alvejar os fenótipos associados com os perfis de expressão diferencial do utilizador. Inclui também uma análise similar à do Gene-Set Enrichment Analysis (GSEA) [19] para identificar o enriquecimento de descritores moleculares nos compostos do NCI-60 e do CMap de interesse. No cTRAP, a similaridade entre resultados de expressão diferencial do utilizador e do CMap baseia-se no enriquecimento dos genes mais e menos expressos do fenótipo observado [15, 19] e nos valores de correlação entre estatísticas de expressão diferencial.

O manuscrito 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 perturbações moleculares responsáveis pelos fenótipos em estudo para melhor compreensão dos mecanismos biológicos associados, tal como prever agentes terapêuticos relevantes.

CompBio: servidor de aplicações *web*

As interfaces gráficas do psichomics e cTRAP auxiliam os utilizadores a explorar a maioria das funções dos respectivos programas de forma interactiva através de passos simples, demonstrados em tutoriais online, que são, tal como as próprias ferramentas, actualizados frequentemente mediante o parecer dos utilizadores.

No entanto, 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. Assim, procurei formas alternativas de disponibilizar as nossas aplicações, até encontrar o ShinyProxy, um programa de código aberto para hospedar aplicações R/Shiny e Python como aplicações *web* através de instâncias (*containers*) de Docker.

Com o ShinyProxy, é necessário um servidor para correr as suas aplicações em tempo real. Dessa forma, para disponibilizar as ferramentas de forma gratuita e acessível, dependendo apenas de navegadores de *Internet* modernos, criámos o servidor CompBio para alojar o psichomics, o cTRAP e diversos outros pacotes de R como aplicações *web* – incluindo programas construídos pelos meus colegas de laboratório (voyAGEr, 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 inte-

ragem entre si, incluindo o ShinyProxy, um *proxy* reverso para gerir os pedidos dos utilizadores (Nginx) e programas para correr processos em segundo plano (Celery, Redis e Flower), analisar o tráfego dos visitantes do *website* (Plausible, PostgreSQL e ClickHouse), monitorar recursos computacionais em utilização (Prometheus e Grafana) e facilitar o desenvolvimento (RStudio Web para testar e desenvolver soluções directamente no servidor).

Todo o código deste projecto está publicamente disponível em `github.com/nuno-agostinho/compbio-app-server`, é de fácil configuração e pode ser facilmente migrado para qualquer máquina, dependo apenas da instalação da plataforma Docker. 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 usadas no projecto e reiniciar esses serviços com o Docker Compose.

Quero aproveitar este momento para sugerir 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 que tenho um tremendo orgulho em apoiar.

Palavras-chave: bioinformática, aplicações *web*, *splicing* alternativo, expressão génica, perturbações genéticas e farmacológicas.

Summary

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

psichomics

Alternative pre-mRNA splicing generates functionally distinct transcripts from the same gene and is involved in the control of multiple cellular processes, with its dysregulation being linked to a variety of pathologies. The advent of next-generation sequencing has enabled global studies of alternative splicing in different physiologic and pathologic contexts. However, bioinformatics tools for alternative splicing analysis from RNA-seq data used to be user-unfriendly, disregard available exon-exon junction quantification or have limited downstream analysis features.

To overcome such limitations, we developed psichomics, an R package with an intuitive graphical interface for alternative splicing quantification and integrative analyses of alternative splicing and gene expression from large transcriptomic datasets, including those from The Cancer Genome Atlas (TCGA), the Genotype-Tissue Expression (GTEx) project, and the recount2 project, as well as user-provided data. psichomics assists the user in integrating sample-associated features (molecular and clinical) 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 pathological contexts.

cTRAP

The Connectivity Map (CMap) hosts differential expression profiles associated with thousands of genetic and pharmacologic perturbations (perturbagens) of human cells. We developed the cTRAP R package to 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 envisage cTRAP allowing users to identify putative causal perturbations to better understand the molecular mechanisms associated with the observed phenotypes, as well as to predict therapeutic targets.

CompBio app server

Both psichomics and cTRAP feature graphical interfaces to assist users in exploring most of their functionality. We set up the CompBio app server based on Docker Compose to deploy our lab's web apps, publicly available at `compbio.imm.medicina.ulisboa.pt`.

Keywords: bioinformatics, web apps, alternative splicing, gene expression, perturbagens.

Acknowledgements

My PhD has been an incredible journey where I met fantastic people, some of which I could never imagine living without. It was a journey where I faced difficult challenges, some of which made me the person I am today. But no matter the obstacles, I am glad that I walked this path full of mesmerising moments and immeasurable learning.

First, a big thank you to Nuno Morais for teaching me during all these years and for inviting me to his lab and iMM, opening the doors to this crazy adventure that now comes to an end. I also want to thank my former lab mates for all the scientific discussions and scientifically-proven fun, specially Marie and Lina that are inspirational people with big smiles and even bigger hearts.

iMM is a remarkable research institute with a vibrant community composed of minds that are as brilliant as they are supportive. Thank you to all those friends that I intend to continue hanging out and sharing my life with: Inês Mendes, Elvira, Eunice, Helena, Filipe, João, Madalena, Inês Faleiro and Ana Duarte. For all the nights out, partying out throughout my PhD and teaching me invaluable lessons while holding one or more beers, thank you Debanjan and Vasco.

To my greatest chaperon over this whole journey, thank you Mariana. No matter how hard the journey was, I could always count on you to share any thoughts, frustrations, achievements, books, computer fonts, colours and nerdy facts. I can only thank you for our time senescing together.

I will carry all my friends in my heart, including those that were already there for a long time: Alicia and João. Our friendship knows no distance. No matter where each one of us lives, we will always be together.

Last, but not least, one special thank you to my mother, father, sister, brother and my whole family. To my grandfather, my biggest supporter even before I was born and the person that made me love biology and science, I dedicate my PhD thesis – I would not be here today if it was not for you. Thank you.

It took millions and millions of years for the Earth to cool down, for the first life forms to reproduce and for mammals to evolve so I could be where I am today. No wonder I feel like these 4 years went by in the blink of a supernova. Although unsure of what lies ahead, I know that I can count on my friends and family to cheer me up. To many more years of friendship, learning and living! See you all in my next step.

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 [21, 22].

And what is life? Although this question is not easy to answer, living organisms as we know them are complex, carbon-based systems composed of nucleic acids, proteins, carbohydrates and lipids. Together with some smaller molecules, these are known as biomolecules and are crucial for the survival of living organisms [23].

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) [23]. Regarding nucleic acids, deoxyribonucleic acid (DNA) stores the genetic data, the blueprint required to generate many of the molecules in the cell, including ribonucleic acid (RNA) molecules for protein synthesis and regulation [23].

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, 23, 24]. 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 RNAs 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, 23].

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* [25]. 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 [25, 26].

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 [27]. Albrecht Kossel (a former professor of Miescher) and colleagues described 5 organic compounds from nuclein: adenine, cytosine, guanine, thymine, and uracil [28–31]. Nuclein was eventually renamed *nucleic acid*, but its importance was not recognised at the time [27].

In the first decades of the 20th century, the scientific consensus was that proteins carried genetic information, but Boveri and Sutton theorised otherwise [27, 32]:

> 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. ([32])

The Boveri-Sutton chromosome theory of genetic inheritance followed Mendel's work from 1865 [32] and was later supported by fruit fly experiments from an initially skeptical Thomas Morgan [33]. 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 interestingly titled *The Chromosomes as Bearers of Hereditary Material* [33]. 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 [34][1].

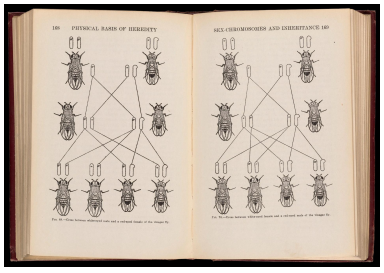


Figure 1.1: Thomas Morgan's fruit fly experiments. Public domain image from Wellcome Library, London.

[1] Muller's 1927 Science paper [34] that earned him the Nobel prize was not peer-reviewed, cited no references and lacked the methods section [35]. 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, who believed his data were not strong enough to confirm the induction of mutations [35]. It would take until 1930 for Muller to publish results addressing Altenburg's criticism, although Muller was aware of the issues before 1927 [35].

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) [39], 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) [40], 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. ([41])

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 [42]. Together with Joshua Lederberg, Tatum demonstrated in 1946 that bacteria can exchange genetic material in a process called genetic recombination [43].

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 [44]. 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 [44].

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 [45]. 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 [45]. 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.) [36]. In 1959, Robert Whitakker suggested a fungi kingdom amid three others [37], a proposal that later blossomed into his popular five-kingdom classification system published in 1969 [38]. 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 [46].

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 [47, 48]. 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 [48, 49]. Multiple ribosomes were found to bind to a single RNA molecule (polysomes), allowing for parallelised protein synthesis [50].

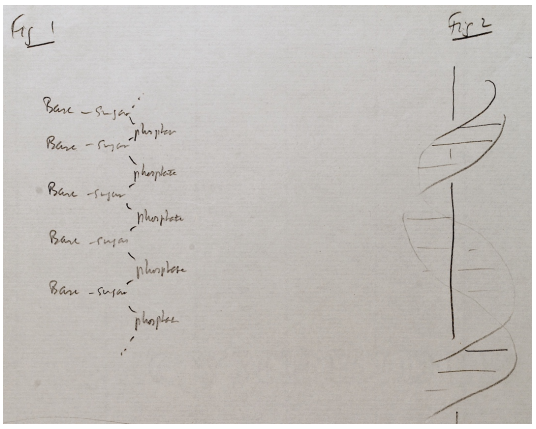


Figure 1.2: Figure drafts for a manuscript on DNA structure from Francis Crick and James Watson. Public domain image from Wellcome Library, London.

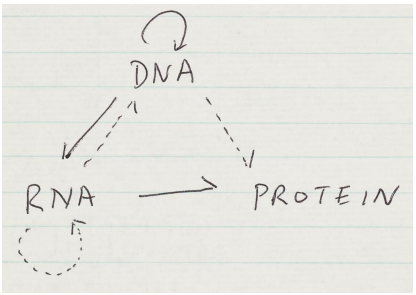


Figure 1.3: Central dogma of molecular biology as drafted by Francis Crick. Public domain image from Wellcome Library, London.

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* [51, 52]. 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 [51, 53] and that those amino acids would be responsible for the protein's three-dimensional structure – and consequently, its functionality [51]. 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 [54, 55]. 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 [56–58] [3].

[3] In 1955, Severo Ochoa and Marianne Grunberg-Manago discovered the polynucleotide phosphorylase (PNPase) enzyme that they thought to synthesise RNA polymers from DNA [59]. Ochoa was erroneously awarded a Nobel prize in 1959 for discovering the biological mechanism of RNA synthesis.

François Jacob and Jacques Monod speculated in 1961 that ribosomal protein synthesis required an intermediate molecule with the template message to convert from DNA to protein and that would act as the *messenger* [48, 60]. 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 [48]. Based on contemporary experiments, Jacob and Monod proposed unstable RNA molecules as relevant candidates and named them messenger RNA (mRNA) [48, 60]. 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 [48].

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. ([61])

Among the first ideas 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 [61].

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* [62]. *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 [62,63]. 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 [62,63]. 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 [64, 65].

1.2.1 Alternative splicing

First reported in mammalian cells infected with human adenovirus 2 [66, 67] and later observed in endogenous mammalian and eukaryotic genes [68, 69], mRNA-DNA hybridisation experiments starting in 1977 suggested that genes are composed by intervening non-coding sequences, based on experiments from Richard Roberts, Philip Sharp and colleagues. During or after transcription of the precursor mRNA (pre-mRNA), a process called *splicing* is responsible for excising segments of the pre-mRNA (named introns), leaving behind the sequences required for protein synthesis (exons) [66, 67, 70]. Moreover, multiple different transcripts may be produced from the same primary transcript by *alternative splicing* of segments of their sequence, thus promoting transcriptome diversity [66, 67, 71–73].

In 1985, an RNA-protein complex composed by the so-called U1, U2, U4, U5 and U6 small nuclear ribonucleoproteins (snRNPs) was reported to be central for RNA splicing: the spliceosome [75]. 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 just upstream of the intron's 3′ end) [76,77]. 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 (Figure 1.4) [75, 78, 79]. The intron lariat is debranched (i.e., converted to a linear form) before its degradation [78, 80].

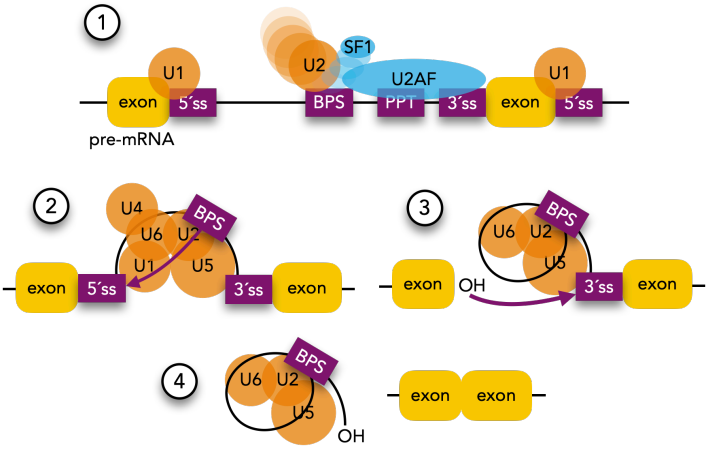


Figure 1.4: Spliceosome assembly and splicing reactions. (1) U1 snRNP binds to the 5′ splice site (5′ss), whereas the splicing factor 1 (SF1) and U2AF proteins bind to the branch point site (BPS), the polypyrimidine tract (PPT), and 3′ splice site (3′ss). The interaction between U1 and U2 snRNPs results 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. Image created by me for [74] (adapted).

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 [81], 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* [82]. 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 [20, 81]. 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 [24]. Interestingly, primary transcripts from yeast can be spliced with the mammalian splicing machinery [24].

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 [83]. 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 [83]. 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 exons and intron retention [83].

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 [84]. After all, the Australian lungfish (*Neoceratodus forsteri*) is the animal with the largest haploid 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) [85–87]. 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 genomes of these species are composed of intergenic regions and introns with high repeat content [85–87]. Notably, 38 000 alternative transcripts may be generated from a single gene in the fruit fly (*Dscam*): some of those transcripts are suggested to play a role in the immune system and may lead to more antigen diversity, thus increasing the evolutionary flexibility of the fruit fly [73].

Alternative splicing is deregulated in multiple disease contexts, including cancer, neurodegeneration and muscular dystrophies [74, 88]. For instance, changes in splicing factors and subsequent perturbations to splicing can affect multiple hallmarks of cancer [89]. Therefore, multiple potential splicing-targeting therapies have been developed based on antisense oligonucleotides, small molecules and novel techniques [88].

1.3 Bioinformatics

Bioinformatics is a multidisciplinary field based on the usage and development of computer programs to analyse large-scale biological data [90]. The first bioinformatic analyses were performed on proteins. Following Sanger's work in 1959 on identifying the amino-acid composition of protein insulin in multiple species, many proteins started being sequenced [90–93]. Pointing to such studies, Crick predicted in 1958:

> 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. ([51])

For that to come to fruition, there was a need to sequence many 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 [90, 94]. All these limitations meant that only 6 different proteins were fully sequenced by 1962 [95]. To overcome those difficulties in the reconstruction step, Margaret Dayhoff and Robert Ledley developed COMPROTEIN, the first bioinformatics program [90, 95]. COMPROTEIN allows to compare a high number of small peptide fragments and suggests possible full protein sequences [95]. 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* [96].

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 [97]. 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 [90]. Since 1970, computer-assisted phylogenetics started being a reality with the introduction of the Needleman-Wunsch algorithm [98] and variants, such as the Smith-Waterman algorithm [99]. These programs computationally measure the distance between two sequences by pair-wise comparison of amino acids between protein sequences [98, 99].

Years after automatic protein sequencing machines being available based on Edman degradation – *protein sequenators* as first called in 1967 [100] –, DNA sequencing methods were presented based on electrophoresis: the enzymatic Sanger method (also known as dideoxy method) [101] and the chemical degradation method Maxam-Gilbert

sequencing [102]. These tedious and slow processes required manual intervention *at both the experimental and interpretative levels* [103].

Leroy Hood and Lloyd Smith published in 1987 a 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 (Figure 1.5) [103]. 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 [103].

Figure 1.5: Applied Biosystems DNA sequencer prototype. Public domain image from Science Museum Group Collection, London.

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 [104] and GenBank [105]. 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 [106]. 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 [107].

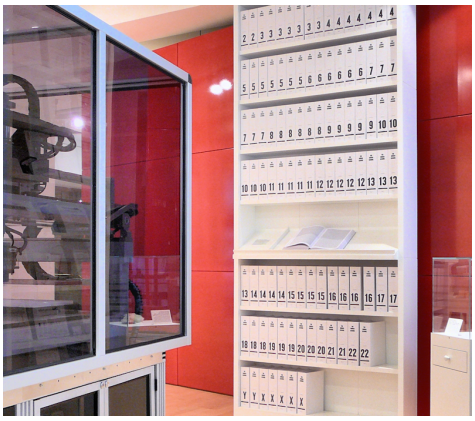


Figure 1.6: Human genome project bookcase. More than a hundred volumes contain the printed human genome sequence located in the Wellcome Collection, London. Public domain image from Wikipedia.

In 1995, the first complete sequence of a free-living organism was published for bacterium *H. influenzae* [108]. From 1996 to 2000, the whole genomes of multiple organisms were sequenced, including for yeast *S. cerevisiae* [109], nematode *C. elegans* [110], fruit fly *D. melanogaster* [86, 87], and plant *A. thaliana* [111]. In 2004, the Human Genome project was considered finished with its

goal of publishing the human haploid genome sequence to the scientific community (Figure 1.6), leaving 8% to be determined due to technical limitations [112,113]. Many advantages flow 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. ([112])

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 sequence for use in biomedical research [113] [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 [114]. 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 [114, 115]. The genomics field is concerned with the study and cross-species comparison of genomes [115].

In the same vein, transcriptomics is the field that studies the transcriptome: the full set of RNA transcripts[5], as first defined in 1996 [116]. Transcriptomics usually leverages

[4] Although the most recent T2T pre-print manuscript from 2021 describes ongoing work for the missing chromosome Y [113], 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.

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 genes identified as expressed via the Human Genome project [117]. EST allow 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* [118].
- **Microarrays**: first mentioned in a 1995 study as a method to simultaneously measure the expression of multiple genes in an high-density array with small wells via cDNA hybridisation [119]. 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* [119]. The first genome-wide microarray study was later conducted in yeast during 1997 [120], followed by the whole human transcriptome based on a cDNA library from infant human brain in 1999 [116].
- **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 [121]. This method is followed by computational mapping of resulting short reads (spanning 50 bases at the time, currently along the lines of 150-200 bases) 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 [121].
- **Long-read RNA-seq**: RNA-seq methods that generate reads between 1 000 and 10 000 bases, allowing to sequence full transcripts – specially advantageous to sequence overly similar transcripts [122]. Given the higher error-rate and cost of this technology compared to short-read RNA-seq, combining both short- and long-read technologies allows to mitigate the technical limitations of both approaches [122].

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 extrac-

ted 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 where 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 (Figure 1.7) [122].

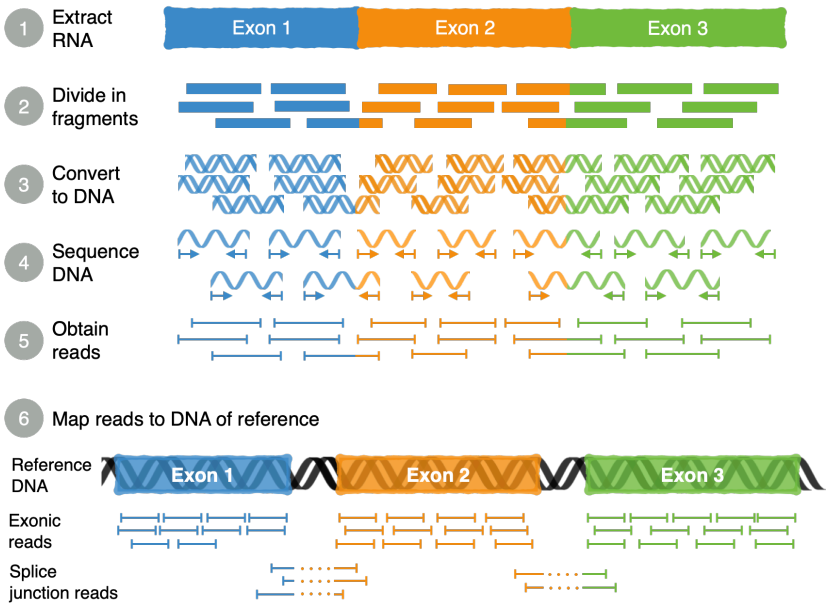


Figure 1.7: RNA sequencing and read mapping. RNA is first extracted from a sample (1) and divided into small fragments (2). These fragments are then converted to DNA (3) and sequenced, producing short text strings called reads (4). Finally, these reads are mapped to a DNA of reference (5) allowing to reconstruct the extracted mRNA and identify exon coordinates in the reference DNA.

Quality control is an important 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 mitigated by trimming reads or even discarding samples [123].

To reconstruct the snapshot 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, some of

[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.

those sequences match multiple genomic sites, such as reads from repetitive regions, and so are distributed based on empiric evidence and/or randomness. Shorter reads are more prone to mapping to multiple genomic regions and their alignment may prove ambiguous. Such issues can be mitigated by gathering more information on the sequence via paired-end sequencing (i.e., sequencing both ends of the cDNA fragment during RNA-seq) or higher read coverage (i.e., higher number of reads).

Based on the number of reads that were aligned to each region of the genome or transcriptome, it is possible to quantify features of interest, e.g., to estimate gene, transcript or exon expression. These counts are then adjusted to make them comparable across samples and to minimise technical variability, a process known as data normalisation. Afterwards, expression values can be linearly modelled across conditions to identify differentially expressed features [122].

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 as it captures more fragments from less expressed isoforms [122]. One approach to study alternative splicing changes is based on the differential expression of isoforms of a single gene (e.g., CuffDiff2 [124]). However, isoform expression estimation based on short reads can be inaccurate because of reads that map to sequences shared by multiple isoforms [122]. Other methods based on exon and junction reads specifically identify alterations in events of alternative splicing [122]: comparing the expression of different exons (differential exon usage, e.g., DEXseq [125]), estimating the percentage of alternative sequence inclusion (percent spliced-in, e.g., VAST-TOOLS [126, 127] and rMATS [128]) or based on graph theory to identify alternative splicing modules (e.g., DiffSplice [129]).

Transcriptomic studies like those performed using RNA-seq 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. This type of studies can be enhanced by integrating genetic variants, methylation, proteomics and other omics data [6, 122, 130].

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 from grants to repeat experiments already performed by others,

enabling new advances in personalised medicine via data sharing [131]. 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 [90], specially when used to train machine learning models. These analyses can be quite time-consuming 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 human cancer types (e.g., breast cancer and glioma) from more than 10 000 samples [6]. Multiple web apps tap into the data from this behemoth, including TCGASpliceSeq [132] and Xena Browser [133].

- **The Genotype Tissue Expression (GTEx)** project, a repository of gene expression data for more than 40 human tissues, totalling more than 15 000 *post-mortem* samples [7]. The 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 the Sequencing Read Archive (SRA) [8, 134].

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 have been used in multiple contexts, including to identify inhibitors of the stemness signature in multiple TCGA cancer types [135], to train machine learning models for designing molecules inducing desired transcriptomic changes [136] and to predict clinically-approved drugs for repurposing as SARS-CoV-2 antiviral agents [137].

To find effective novel treatments, it is important to understand how different cells respond to specific compounds. Multiple drug sensitivity datasets contain compound toxicity data across diverse cell lines, including the NCI-60 [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 and inefficient communication between different platforms [131]. Another major hurdle of data dissemination is privacy issues and insufficient anonymisation of clinical data, 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.

Releasing software as open source is important to 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 and portability across machines [138].

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 [122]. 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 [139, 140].

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 [139–141]. 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) [140, 142]. 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 [140, 143].

- **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, allow to easily integrate code from different developers and compare files across diffterent versions [140, 144].

  - Kanban-like boards allow to visualise and manage the project workflow, where features and bugs can be commented on and tracked [140, 142].

  - 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 [139, 140, 143, 144]. 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 [145] and Sphinx for Python [140, 142].

- **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) [140, 143, 144]. 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 [142].

- **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 [144].
  - 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 [146] 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 [140]. 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) [140].
  - 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 current 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 [139, 140, 142, 144].

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 [140, 142], as well as it facilitates analysis transparency and reproducibility [140]. Moreover, it has educational value, allowing others to learn from the developed code, implemented solutions and project organisation [142].

1.3.4 Web apps and graphical user interfaces

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 locally rendered by the user's 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 [147] and assessing surgical performance [148] to predicting high-risk patients for primary palliative care [149]. 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., [150, 151]).

To provide effective dashboards, good practices in data visualisation are paramount for intuitive communication of complex results [152]. 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 [152].

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 [140, 142, 152, 153]. Interfaces can be evaluated and improved by asking for and listening to user feedback or by performing usability testing [152].

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.

1.3.5 R programming language

The work presented in this thesis was mostly developed using R [154]. R is a programming language with a strong foundation in statistics and includes native support for

data manipulation, clustering, survival analysis, principal component analysis, linear modelling and classical statistical tests, such as Mann-Whitney U, Kruskal–Wallis and Fligner–Killeen tests [154,155]. We also used RStudio (`rstudio.com`), a popular integrated development environment (IDE), to assist using R for data analysis and package development.

Besides the built-in functions, R is extendable with open-source packages from multiple repositories, e.g., CRAN for generic R packages and Bioconductor for bioinformatic R packages [146]. We used packages from such repositories to:

- perform bioinformatic analyses, including differential gene expression (`limma` [156] and `edgeR` [157]);
- create interactive data visualisations based on `ggplot2` [158] for standard R plots and `highcharter` [159] for interactive JavaScript plots;
- design web apps using the `shiny` [160] framework;
- integrate C++ in R with `Rcpp` [161], a more efficient and low-level programming language, allowing to speed up computationally-intensive tasks;
- profile time and memory to pinpoint bottlenecks that need improvement by using `profvis` [162] and `microbenchmark` [163];
- develop R packages with `devtools` [164] and write their documentation using `roxygen2` [165].

To facilitate the use of the code in this thesis, our programs are distributed as open-source in GitHub and Bioconductor. Moreover, our tools were encapsulated with their dependencies in Docker images, making them quick to set up in any machine and easy-to-use in high-performance computing (HPC) environments. We also used these Docker images to deploy our tools as free web apps.

The LaTeX code to compile this PhD thesis is open-source and can be found at `github.com/nuno-agostinho/phd-thesis`.

Chapter 2

Objectives

The aim of the scientific work discussed in this thesis is to develop transcriptomic apps with graphical and command-line interfaces to retrieve and analyse publicly available transcriptomic data from popular repositories, such as TCGA [6], GTEx [7], recount2 [8] and CMap [15]. Besides, we intend their graphical interfaces to be intuitive, flexible and deployed online as web apps, so that they are used by a wide range of researchers. We think that bridging the gap between big data and researchers with basic computational skills will inspire them to learn more about bioinformatic analyses and thereby potentiate their ability to extract novel biological insights from genome-wide molecular information.

Alternative splicing is involved in multiple cellular processes, with its deregulation being linked to diverse pathologies. The use of pre-processed alternative splicing data from public repositories exempts researchers from storing and processing large raw files that require expensive computational resources. However, until 2018, no tools interactively fetched those public data and allowed their analyses using user-defined sample groups based on associated metadata. Thus, there was a need to develop a program with a graphical interface that allows to quantify, analyse and visualise publicly available alternative splicing data. We used the R programming language to create psichomics [9, 10], an open-source R package with a Shiny-based web interface that automatically downloads biological data from TCGA [6], GTEx [7] and recount2 [8]. By making use of popular R packages like `edgeR` [157] and `limma` [156] for gene expression analysis, psichomics interactively performs dimensionality reduction, differential splicing and gene expression and survival analyses with incorporation of molecular and clinical features.

CMap is a public database containing millions of gene expression changes in response to induced molecular and pharmacological perturbations [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. To facilitate this approach, we strived to build a program

to identify potentially causal molecular perturbations by comparing user-provided differential gene expression results with those from CMap, using correlation and gene set enrichment scores. Moreover, the program should also use gene expression/drug sensitivity associations from public databases to identify compounds that may target the phenotypes associated with the user-provided differential expression profiles. We used R/Shiny to create cTRAP, an R package with graphical interface functions to assist users performing the aforementioned features.

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 Shiny apps accessible via a web browser, we intended to build an open-source and portable codebase to easily deploy web apps using our own custom server. Therefore, I led the development of an app server based on Docker and ShinyProxy to deploy psichomics, cTRAP and multiple R packages as web apps – including other programs built by my lab colleagues in the lab (voyAGEr, betAS and scStudio). I kindly invite you to pause, sit back and relax, visit our website at `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 [160] that helped me to kick-start what would be later known as psichomics (Figure 3.1).

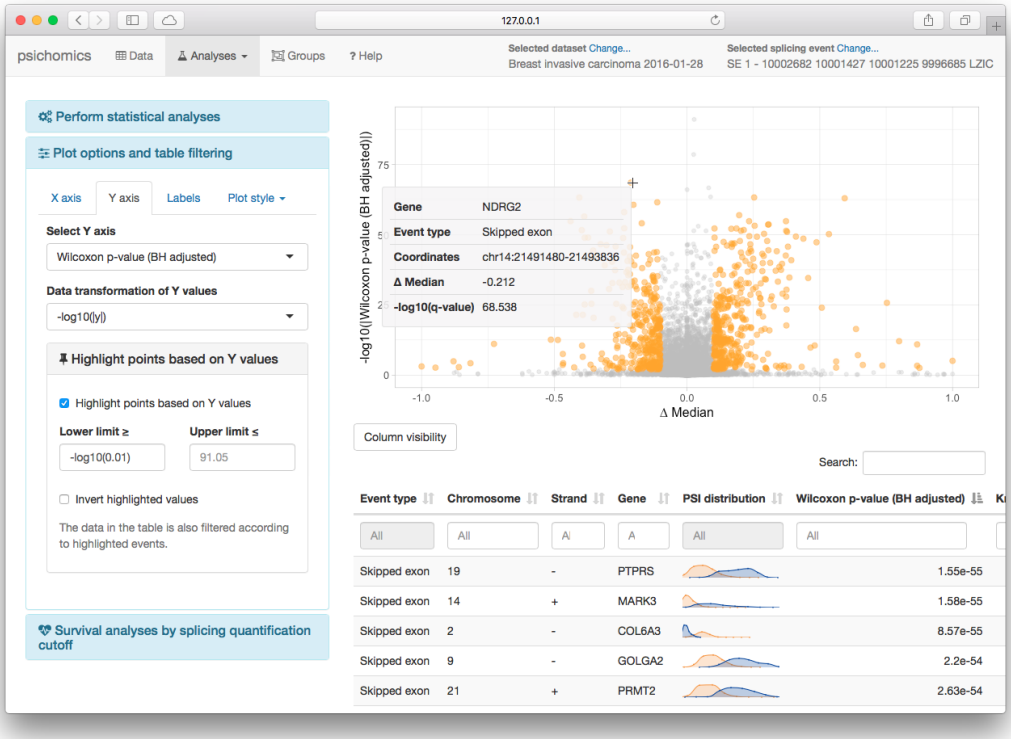


Figure 3.1: psichomics screenshot. TCGA breast cancer splicing analysis (28 Jan 2020).

psichomics was first available in 2016 via Bioconductor to quantify, analyse and visualise human alternative splicing using TCGA data [6]. Later on, I started my PhD in the same lab and continued my work on psichomics. Nowadays, the tool also analyses gene expression and alternative splicing based on user-provided or public transcriptomic data, including those from 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 human genome assembly hg38
1.4.3	13 Jan 2018	Faster alternative splicing quantification via `Rcpp`/C++ [161]
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 [166]
	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 [126, 127] output[b] and more 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 alternative splicing quantification tool VAST-TOOLS [126, 127]. 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 Nuno Morais. The content of those publications, along with some content from my MSc Thesis [167], 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 [1–5] and promoted large consortium efforts to assemble publicly accessible splicing data. Such efforts include TCGA that catalogues clinical and molecular profiling data from multiple human tumours [6]; GTEx 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 [2–5, 168].

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 [169], MISO [170], SpliceSeq [171], VAST-TOOLS [126], rMATS [128], SUPPA [172] and Whippet [173]. Regardless of their quantification metric, alternative splicing analysis tools had at least one of the following shortcomings in 2018:

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, until 2018 no tool performed 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 [174] and DIEGO [175] 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 as a web app at `compbio.imm.medicina.ulisboa.pt/psichomics` and can 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 analyses, 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.

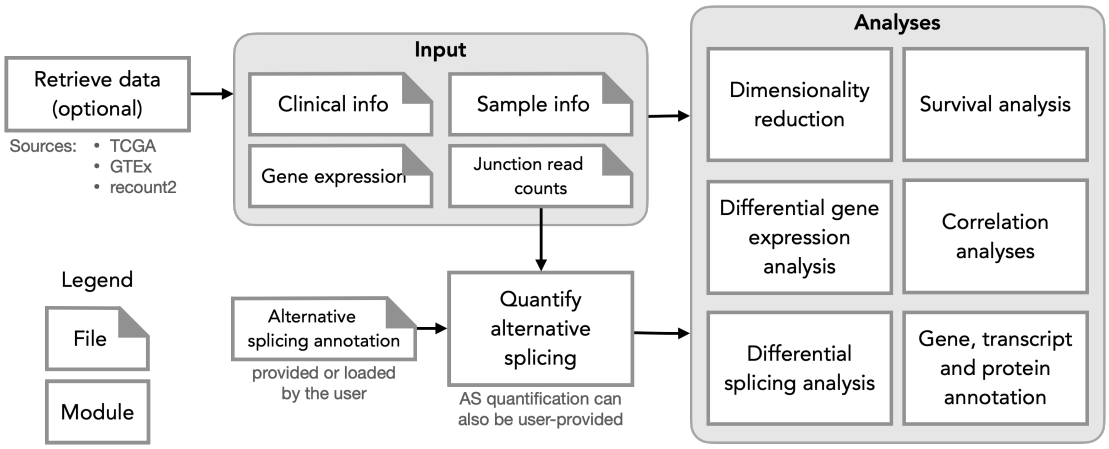


Figure 3.2: psichomics workflow. The user can provide their own input data or load data from TCGA, GTEx or recount2 to normalise gene expression data and quantify alternative splicing for downstream analyses.

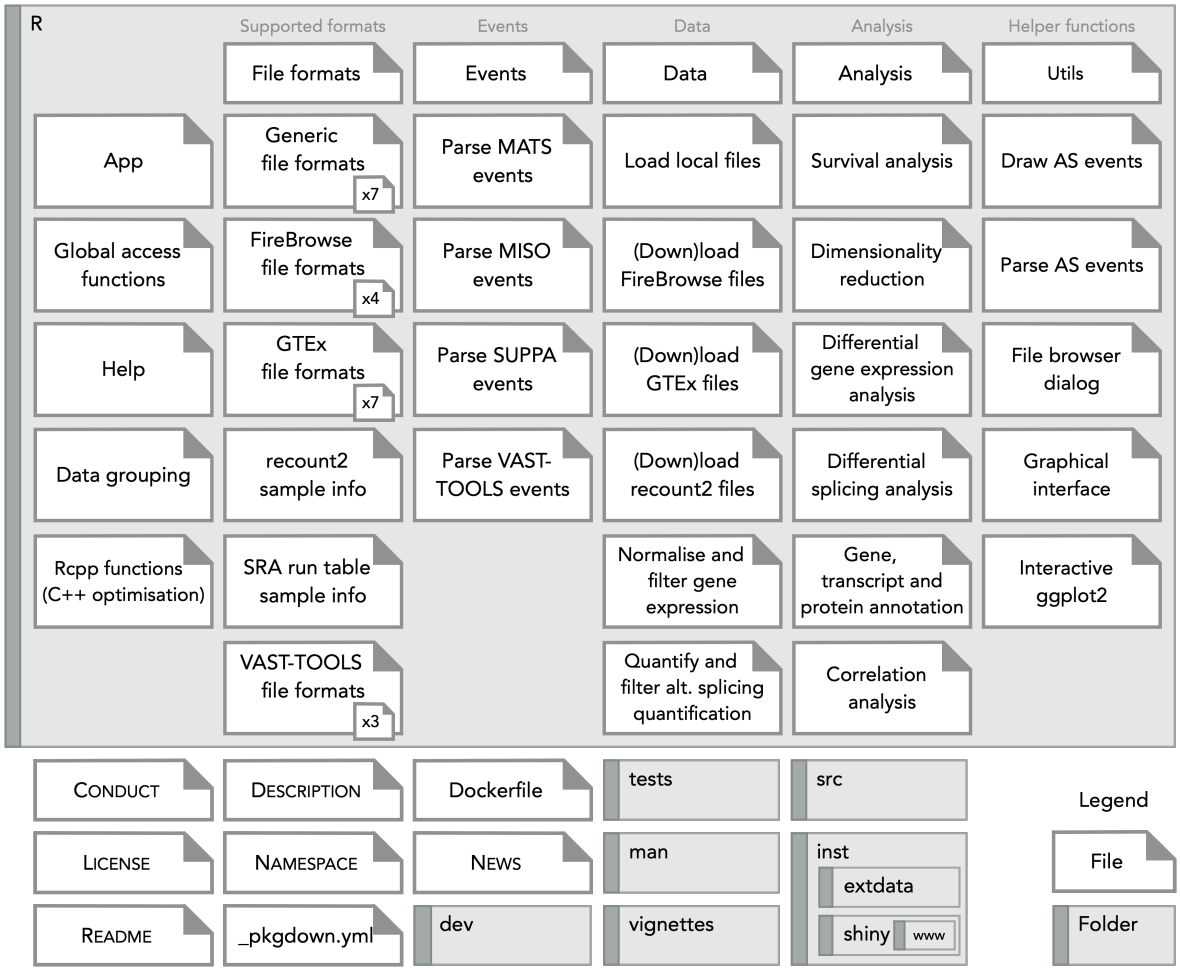


Figure 3.3: Visual representation of psichomics' file structure. psichomics is a modular program where, for instance, functions specific for different data sources and analyses can be found in different files. As usual for R packages, the `R` folder is the heart of the code and contains the main R scripts 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]Refer to 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()` [157]. 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 [170], VAST-TOOLS [126], rMATS [128] and SUPPA [172]. 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 [176] in GTF and TXT formats, so that gene identifiers in the GTF file (misleadingly identical

[3] Refer to tutorial at `nuno-agostinho.github.io/psichomics/articles/AS_events_preparation`.

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()` [177], 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.

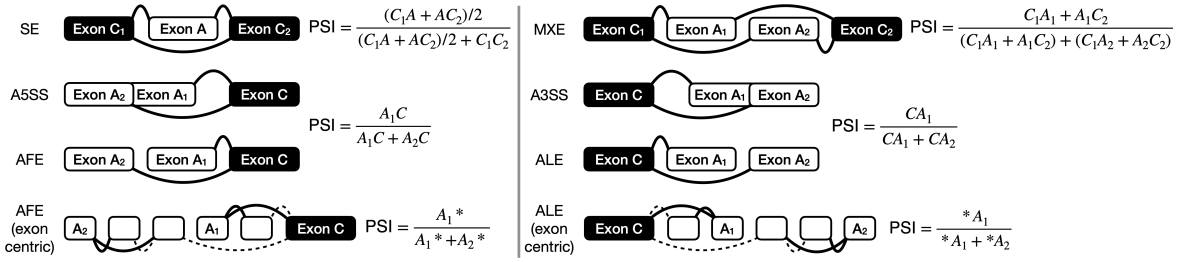


Figure 3.4: Alternative splicing quantification. Splice junctions required to quantify alternative splicing based on event type. C1A and AC2 represent read counts supporting junctions between a constitutive (C1 or C2, respectively) and an alternative (A) exon and therefore alternative exon A inclusion, 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 [178], 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 [179] 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 [180], 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) [181] and proportional hazard (PH) models [182] 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` [155].

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 [183, 184], 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 [185]. 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()` [156]) 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 PSI values

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 [166], 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 [186], UniProt [187], the Proteins API [188] and PubMed [189] 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 [190], GeneCards [191], the Human Protein Atlas [192], the UCSC Genome Browser [193], UniProt [187] and VAST-DB [127].

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()` [194]. 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 [127] were aligned with splice-aware STAR [195] 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 (function `psichomics::quantifySplicing()` with default settings) were compared with the respective RT-PCR- and VAST-TOOLS-derived PSI values, available from VastDB [127].

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 [132] provides pre-computed alternative splicing quantifications across TCGA cohorts, similarly to psichomics. As such, 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 to (1) create Docker images and store them in GitHub and Docker Hub for every psichomics release or change in the dev branch; (2) update the package documentation website via `roxygen` and `pkgdown`; and (3) build and check 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 [196–199]. 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 [200].

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 [196, 201]. 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 [196, 201, 202].

Using the graphical interface of psichomics, we analysed SRA project SRP063867 [203] 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 download 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).

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` [157] and `limma` [156] 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

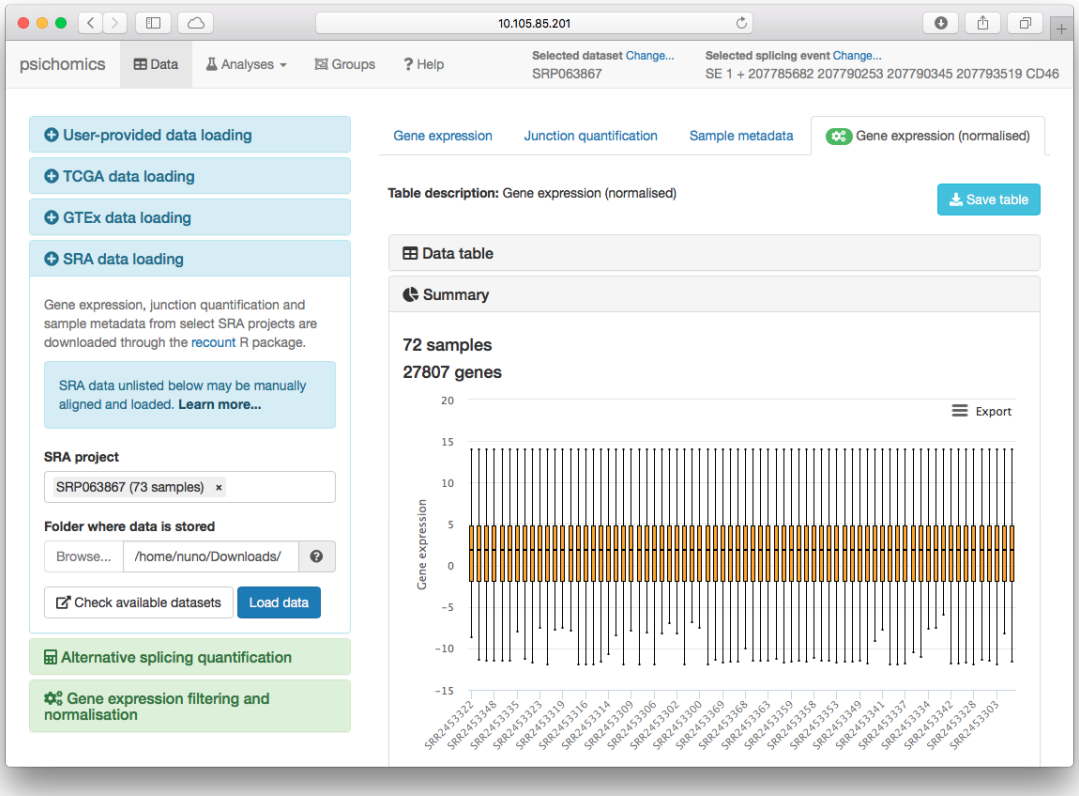


Figure 3.5: Summary information on datasets. psichomics presents information on the loaded datasets in a dedicated tab. That information includes summary statistics, like the numbers of samples and genes profiled in each dataset, and plots, like the shown boxplot to visualise the distribution of normalised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

size (i.e., the total number of mapped reads) suggests 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 values 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) [157], 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 [156]. The distributions remain heterogeneous (Figure 3.6c) based on the default weighted trimmed mean of M-values (TMM) [157], used to normalise for library sizes, so we replaced it by quantile normalisation [156]. This more vigorous normalisation of gene read counts made their distributions comparable across samples, except for SRR2453313 (Figure 3.6d) that was thus discarded. No obvious outlying gene expression distribution is apparent after discarding that sample and renormalising (Figure 3.6e). The filtered and normalised

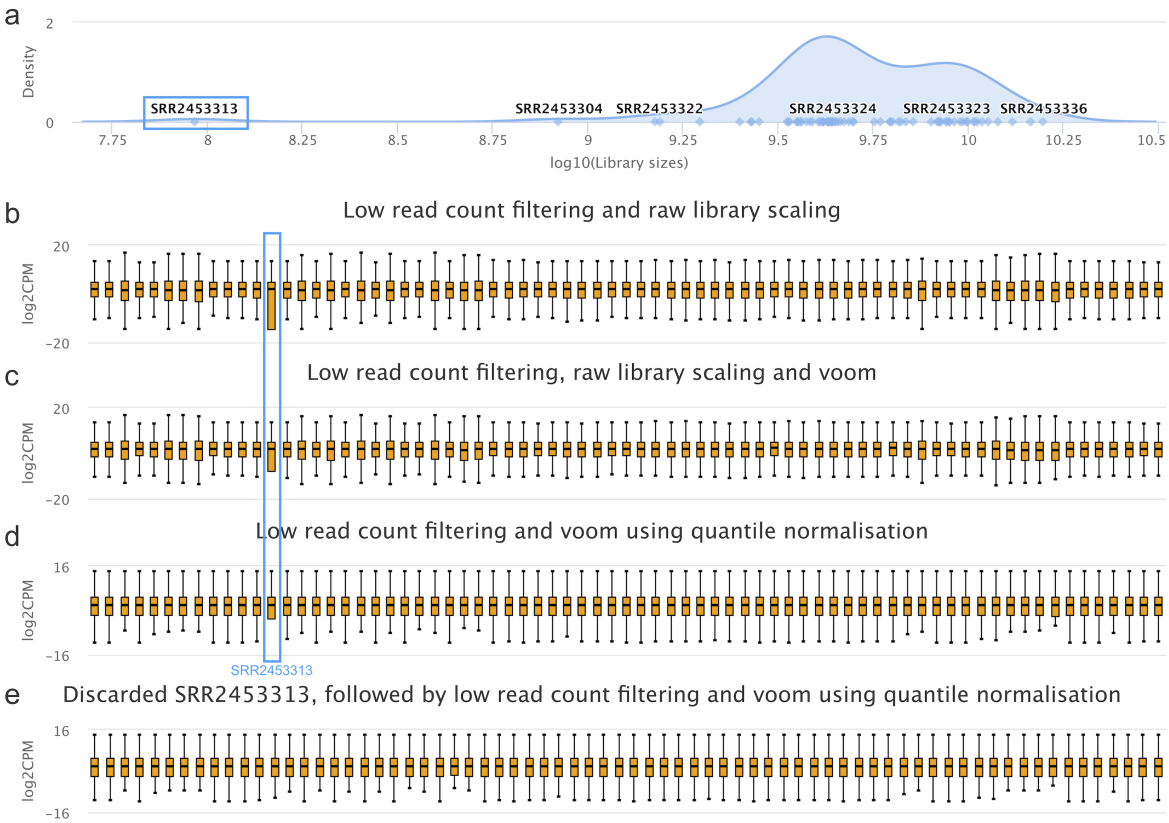


Figure 3.6: Gene expression normalisation. (a) Density plot of the distribution of raw library sizes (i.e., total number of mapped read counts) across samples. Highlighted with the blue label is the sample with the smallest library. (b–e) Boxplots of distribution of gene expression in log2-transformed 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).

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., 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). The number of potentially interesting events was reduced from 135 717 to 27 401.

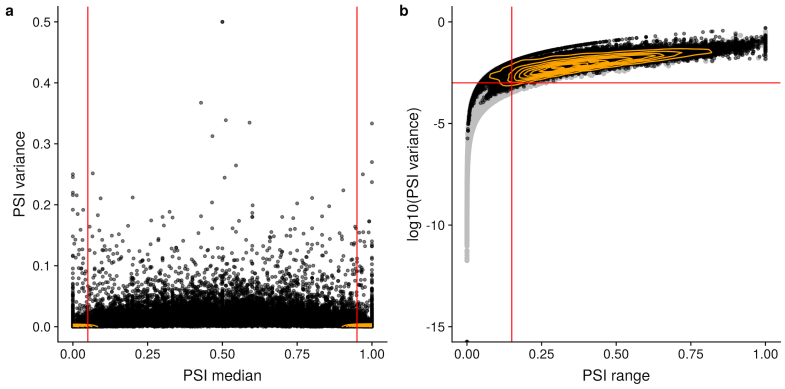


Figure 3.7: Alternative splicing quantification filtering. (a) Selection, for further analyses, of alternative splicing events with median PSI values between 0.05 and 0.95. (b) Further filtering those events with PSI range > 0.15 and $log_{10}(\text{variance}) > -3$. For illustration purposes, grey points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

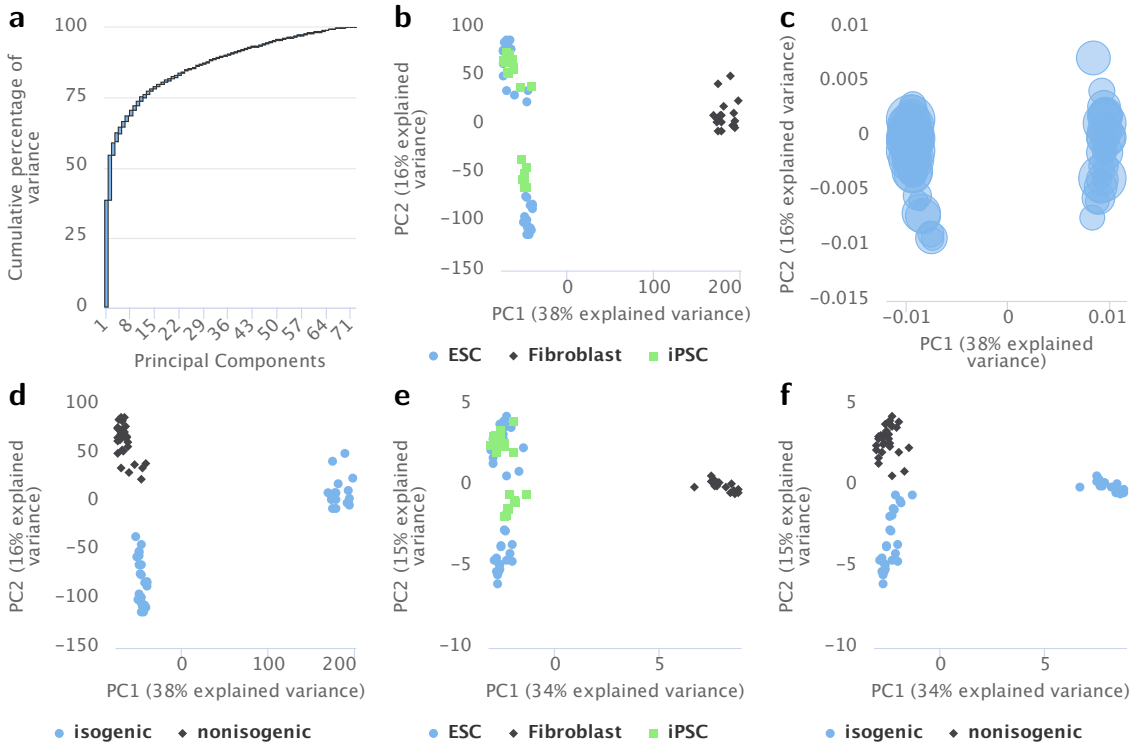
Principal component analysis (PCA)

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.

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

[4]More information in subsection 3.2.6: **Dimensionality reduction**.



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 gene expression 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). (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. For performance reasons, only the 100 most contributing variables (i.e., genes or alternative splicing events) to the selected principal components are plotted by default. (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.

genes, as displayed in the table below the loading plot, at least *DNMT3B* and *RBPMS2* have been previously associated with pluripotency [204]. 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 [204].

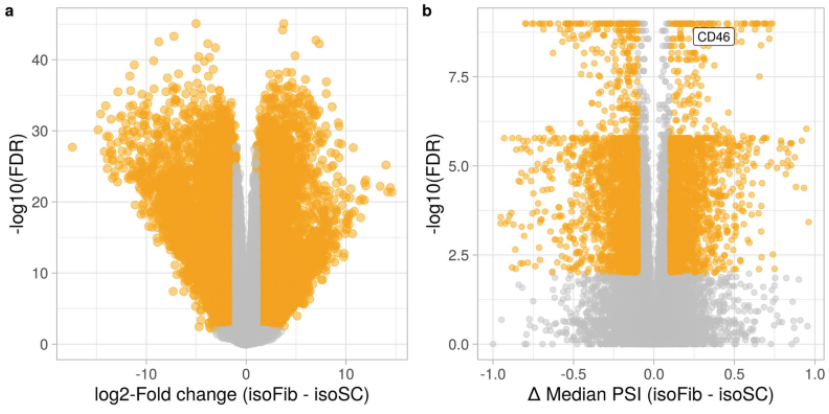
Similarly, we performed and plotted PCA on alternative splicing data[5]. Akin to the observations from PCA plots on normalised gene expression, principal component 1 appear to be associated with cell type and principal component 2 with isogenicity (Figure 3.8e-f). The table below the loading plot allows to assess which alternative

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

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* [198].

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 [156].



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 (isoSC) and green for isogenic fibroblasts (isoFib). The selected alternative splicing event (in blue) depicts the *CD46* penultimate exon inclusion.

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 significance stratum in the top 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-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 its PSI distributions 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 that may cause the respective transcript to be targeted for nonsense-mediated decay [205].

We also tested the correlation between the gene expression of a list of RNA-binging proteins [166] 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 is reportedly regulated by the ESRP1/2 proteins, involved in the epithelial–mesenchymal transition and associated with the generation of cancer stem cells [201, 205]. ESRP1 has also been reported to regulate ES cell differentiation [204].

Extending the analyses to GTEx and TCGA

To correlate *ESRP1/2* gene expression across GTEx tissues and TCGA tumour types against the PSI values of the penultimate exon of *CD46*, gene expression data was filtered and normalised using voom with quantile normalisation [156] and alternative splicing was quantified based on annotation *Human hg38 (2018-04-30)*.

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 low read counts, 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 breast cancer and most cancer types (Figures 3.12 and 3.13).

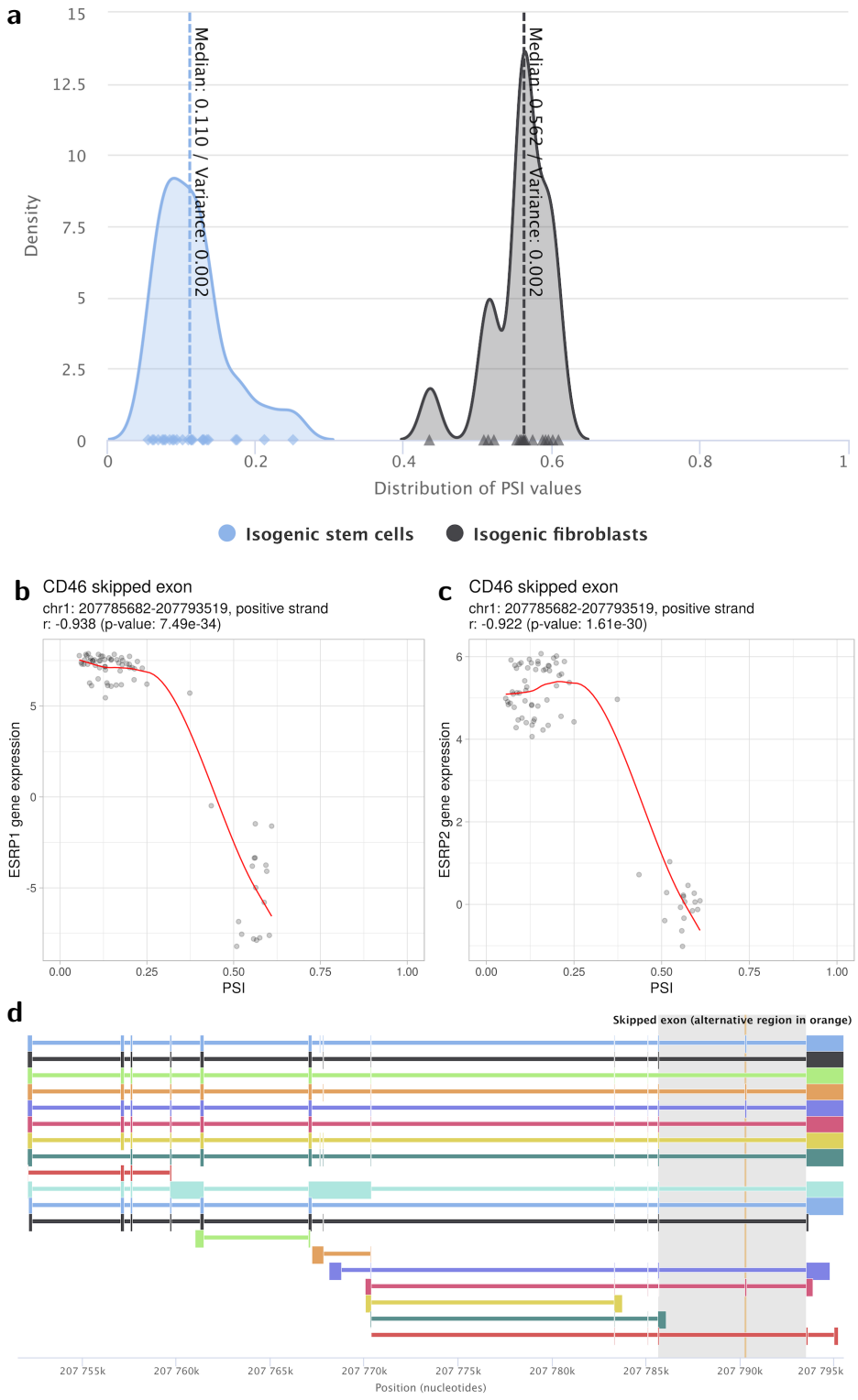


Figure 3.10: Alternative splicing of the *CD46* penultimate exon. (a) Density plots of distributions of *CD46* penultimate exon PSI values across samples, colored by isogenic stem cells (pink) and isogenic fibroblasts (green). (b-c) Scatterplots of PSI values for *CD46* penultimate exon inclusion versus 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).

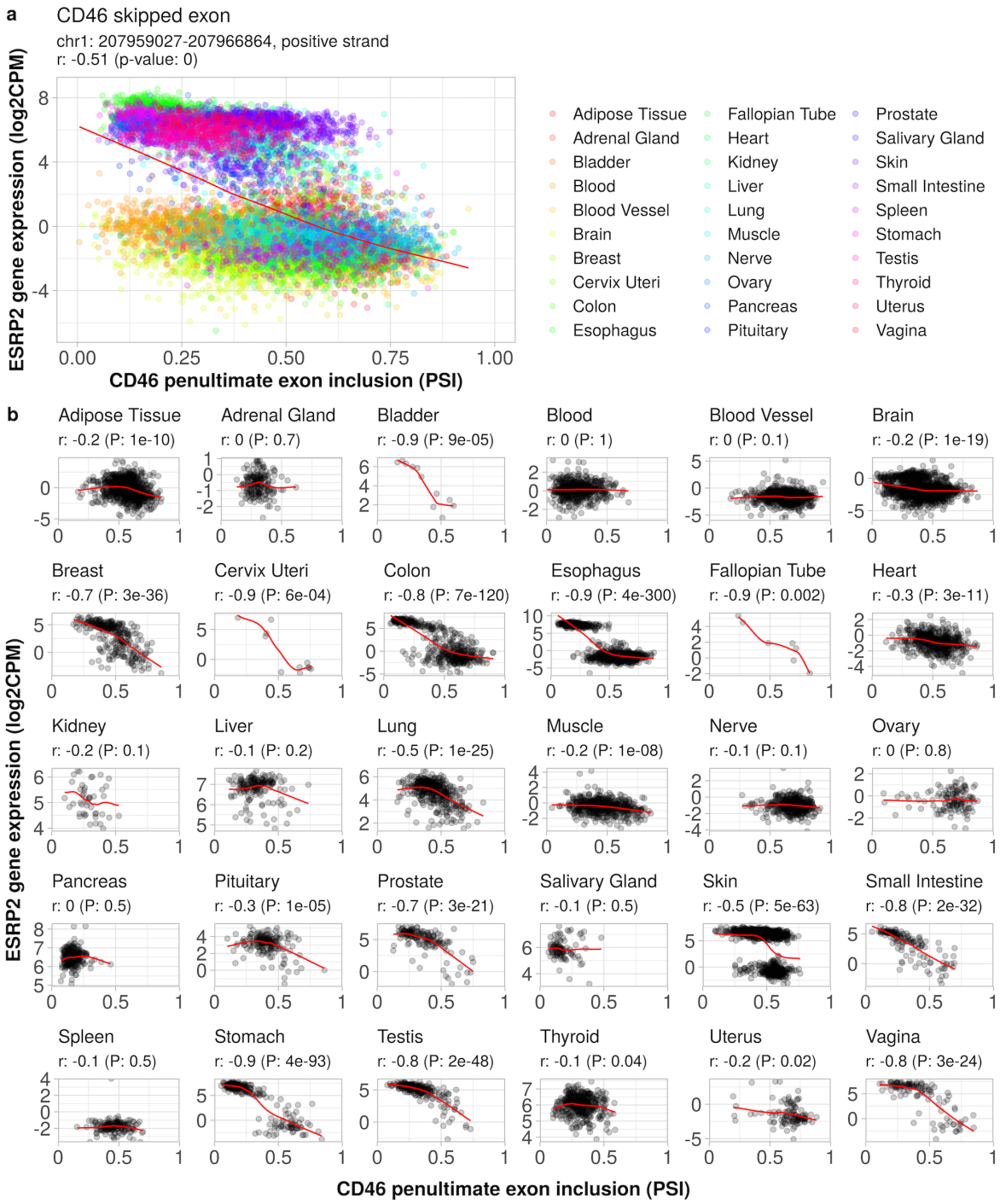


Figure 3.11: Scatterplots of normalised *ESRP2* expression versus PSI values for *CD46* penultimate exon inclusion across GTEx tissues, altogether (a) and by tissue (b). For each plot, the red line illustrates the fitted Loess regression curve. The Pearson's correlation coefficients (r) and associated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

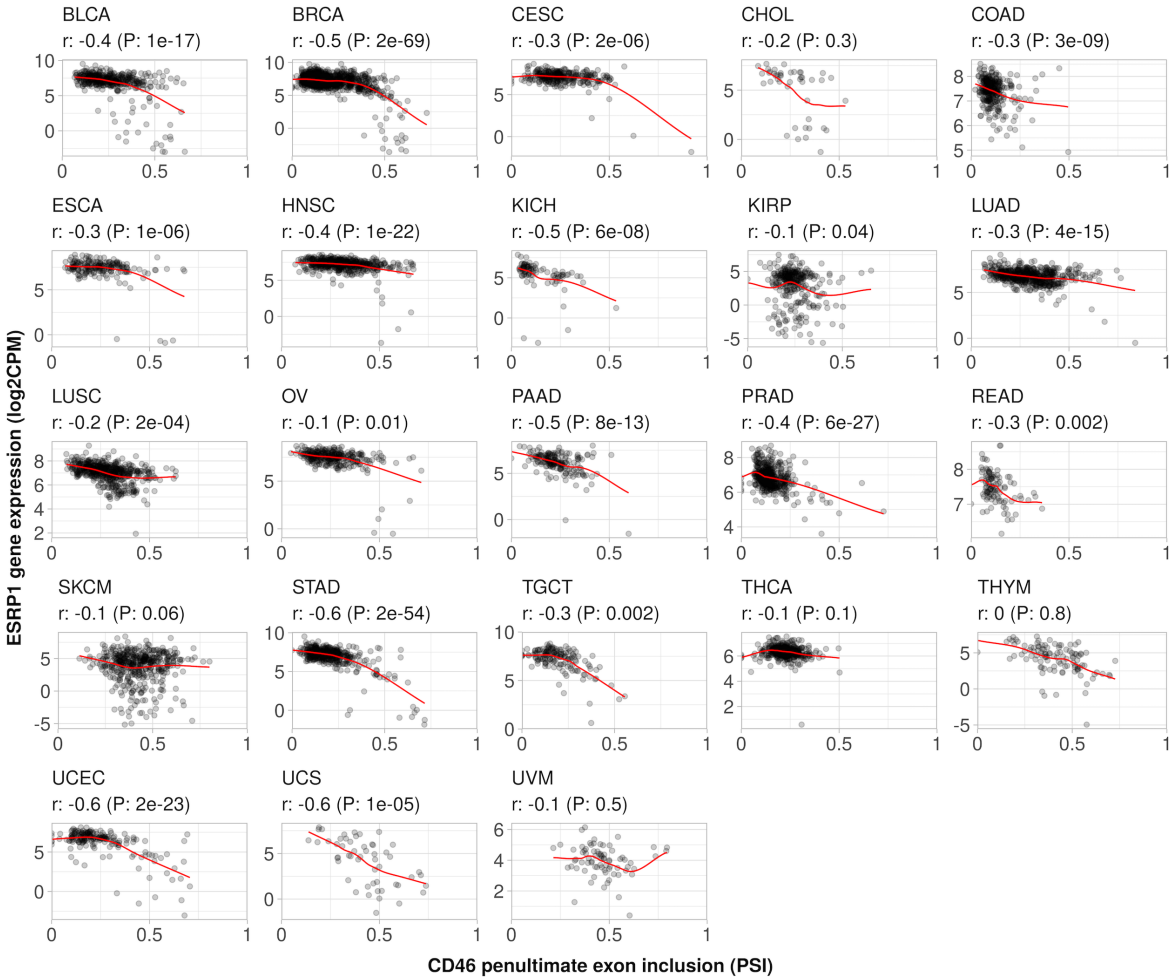


Figure 3.12: Scatterplots of normalised *ESRP1* expression versus PSI values for *CD46* penultimate exon inclusion across TCGA tumour types. For each plot, the red line illustrates the fitted Loess regression curve. The Pearson's correlation coefficients (r) and associated p-values (P) are shown.

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.

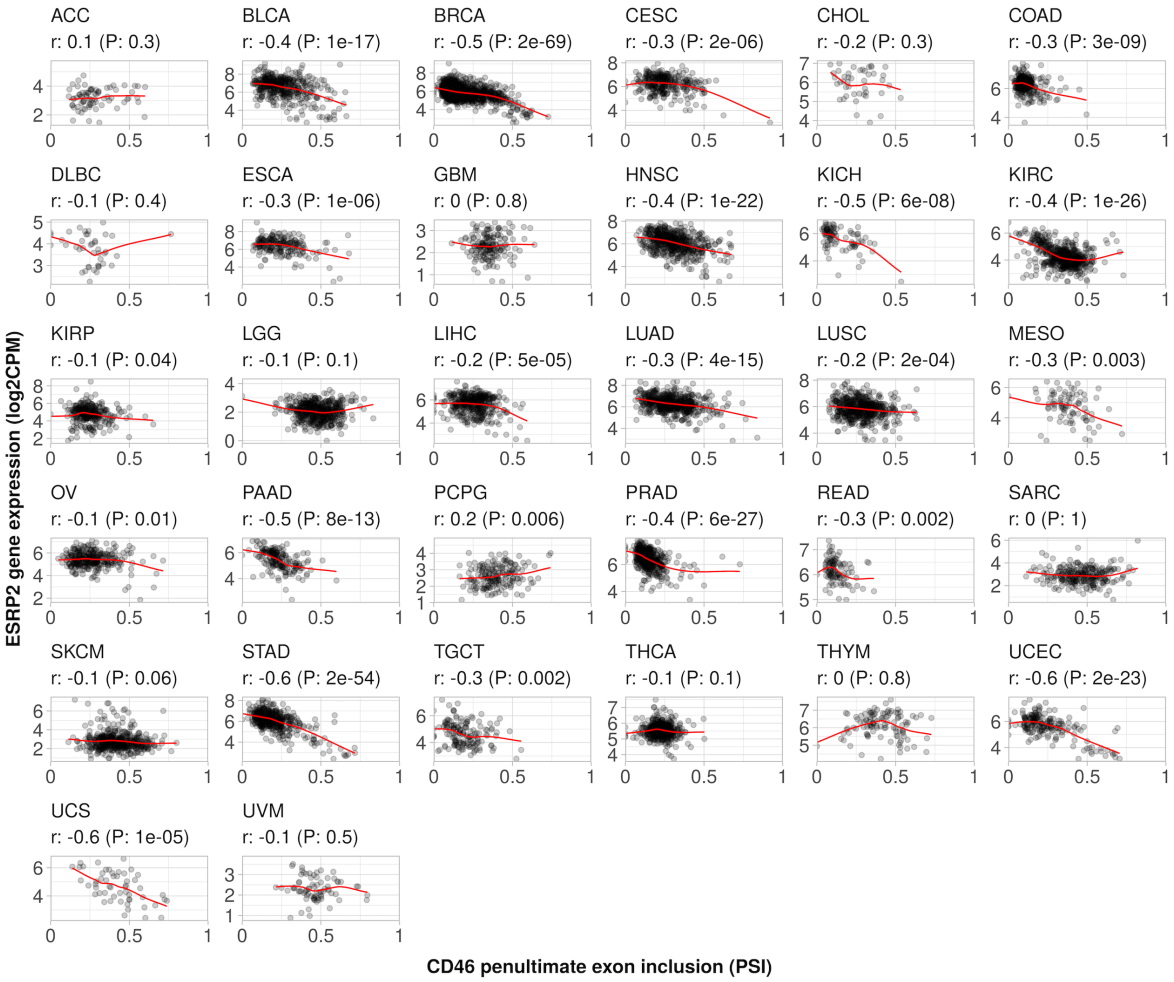


Figure 3.13: Scatterplots of normalised *ESRP2* expression versus PSI values for *CD46* penultimate exon inclusion across TCGA tumour types. For each plot, the red line illustrates the fitted Loess regression curve. The Pearson's correlation coefficients (r) and associated p-values (P) are shown. See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

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.

We performed overall survival analysis by selecting right data censoring[6], 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.

[6]psichomics supports left, right, and interval data censoring for survival analysis. Events may occur after the last observation (right censoring, such as in the case of subjects having no event reported during the study or dropping from it altogether), before an observation is performed (left censoring, for instance when events happened in an uncertain time before the start of the study) or in-between observations (interval censoring, such as when patients require periodic follow-ups and the time of an event occurrence falls between follow-ups but is not certain) [206].

For patients whose days to death are not available, we use days to last follow-up for right-censoring.

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 with low and high inclusion of the *CD46* penultimate exon. 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 maximising the significance of the survival difference is automatically selected. 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).

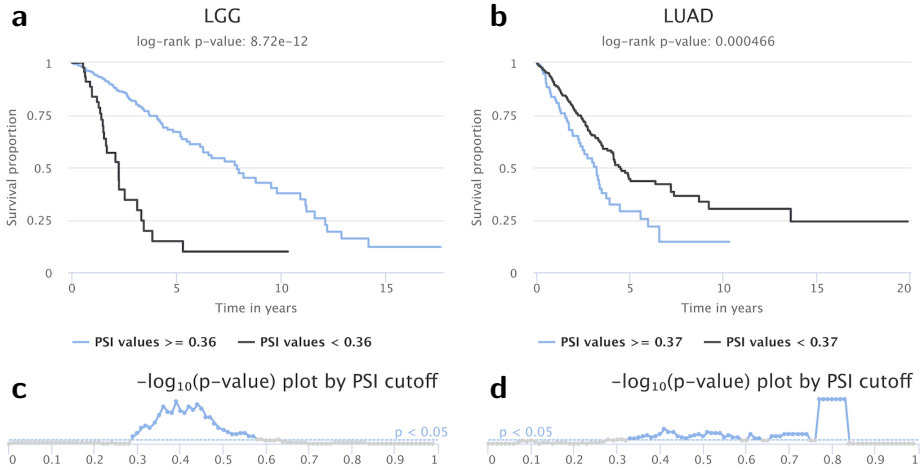


Figure 3.14: Prognostic value of *CD46* penultimate exon inclusion across select TCGA cancer types. (a, b) Kaplan-Meier plots of overall survival for all patients stratified by the respective alternative splicing event's PSI cutoff that maximised the significance of differences in survival between patient 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.

3.3.2 Time benchmarking

The runtimes 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 for which takes more time to load, quantify and analyse alternative splicing and gene expression data. Contrastingly, processed data from GTEx come bundled in files containing all tissues. Although only data from specified tissues are loaded, scanning though the large GTEx file still delays data loading. Tissues from GTEx were loaded in pairs for subsequent differential splicing analyses (Figure 3.15A).

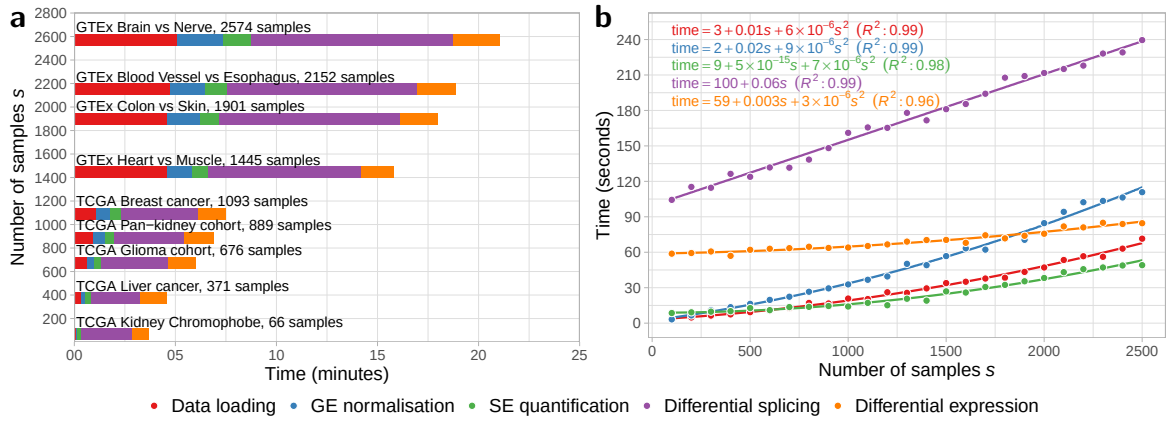


Figure 3.15: Performance benchmark for alternative splicing analysis using RNA-seq data from multiple TCGA and GTEx sample types. (a) Median times of 10 runs of data loading, gene expression (GE) normalisation, skipped exon (SE) event quantification and differential expression and splicing analysis (normal 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 multiple sample sizes were generated, based on TCGA data distributions, to determine the time complexity of each step in psichomics as a function of the number of input samples s (Figure 3.15B). 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 [174] and DIEGO's [175] 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 [126] across multiple tissue and cell line samples from human, mouse and chicken [127]. VAST-TOOLS fol-

lows 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 (Figure 3.16), 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).

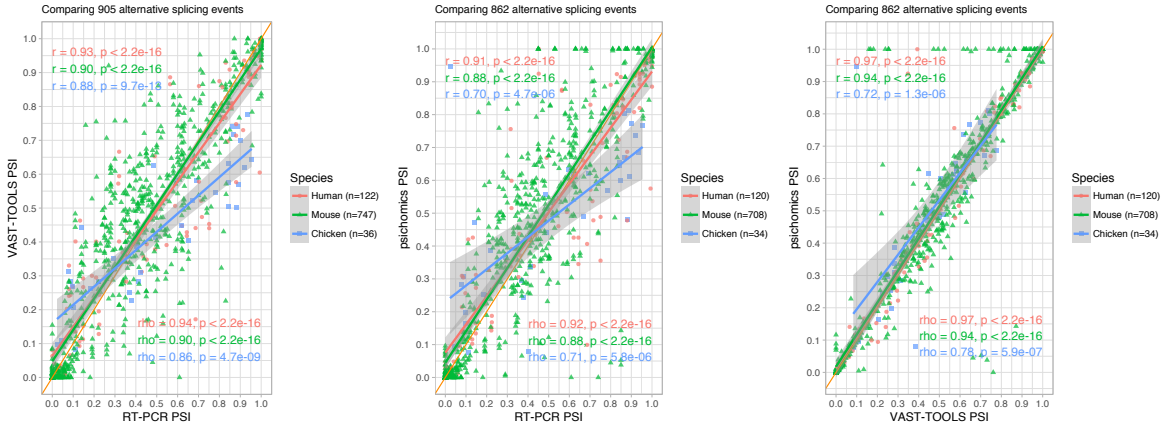


Figure 3.16: Comparison between PSI values estimated by psichomics, VAST-TOOLS and RT-PCR across multiple tissue and cell line samples from human, mouse and chicken. Pearson's (r) and Spearman's (rho) correlation coefficients and respective p-values are shown. Linear regression lines are coloured per species with the respective 95% confidence interval represented as their shades. Identity line in orange.

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 (10 000 times for each combination). Figure 3.17 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 [132], 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 $< 10^{15}$; Figure 3.18).

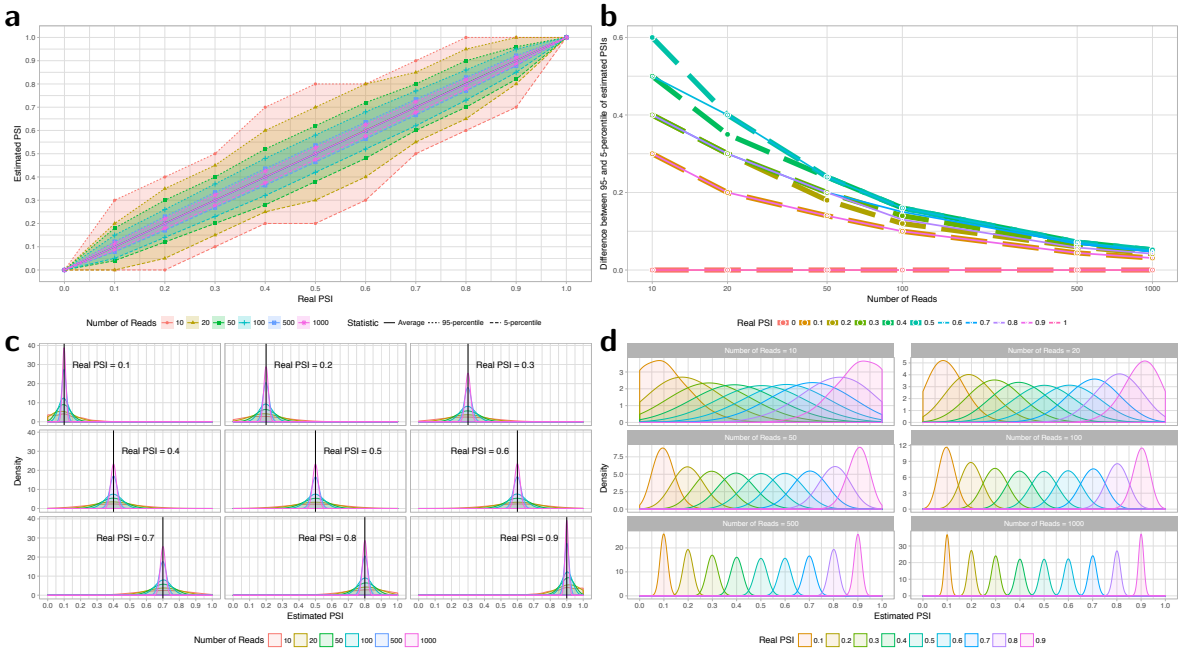


Figure 3.17: Dependence of accuracy of psichomics PSI quantification on event read coverage. (a) Comparison between simulated "real" PSI values and the mean, 95th percentile and 5th percentile of their corresponding PSI estimates for different simulated numbers of junction reads. (b) 90% prediction interval (i.e., difference between the 95th and the 5th percentiles) across different number of junction reads for simulated "real" PSI values. (c,d) Density plots of PSI estimates for each simulated "real" PSI and junction read coverage combination (except those involving PSI = 0 and PSI = 1). Each "real" PSI and junction read coverage combination was simulated 10 000 times.

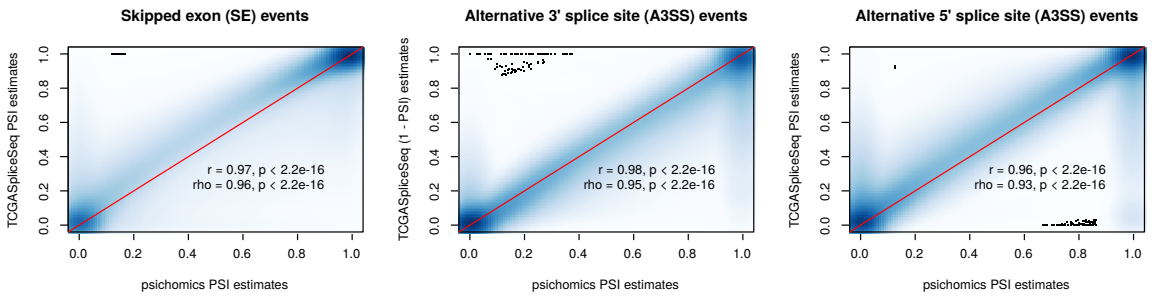


Figure 3.18: Correlation of PSI estimates for TCGA samples between TCGASpliceSeq and psichomics. For A3SS events, PSI values from psichomics correspond to 1–PSI from TCGASpliceSeq (the splice site deemed as alternative in A3SS events by TCGASpliceSeq is constitutive in psichomics and vice-versa). Pearson's (r) and Spearman's (rho) correlation coefficients and respective p-values are shown. Identity line in red.

3.4 Conclusion

Alternative splicing is a regulated molecular mechanism involved in multiple cellular processes and its dysregulation has been associated with diverse pathologies [1, 207–209]. 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 in-

deed playing crucial role in the improvement of our insights into the role of alternative splicing in both physiological and pathological contexts [1–3, 74, 208]).

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`[7] 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 [178]. 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

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

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 [128], which is able to generate *de novo* annotations[8].

Since its publication, psichomics has been used to analyse alternative splicing in multiple scientific articles, such as [11–14]. 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 `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 the results of a custom differential gene expression analysis 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 of 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).

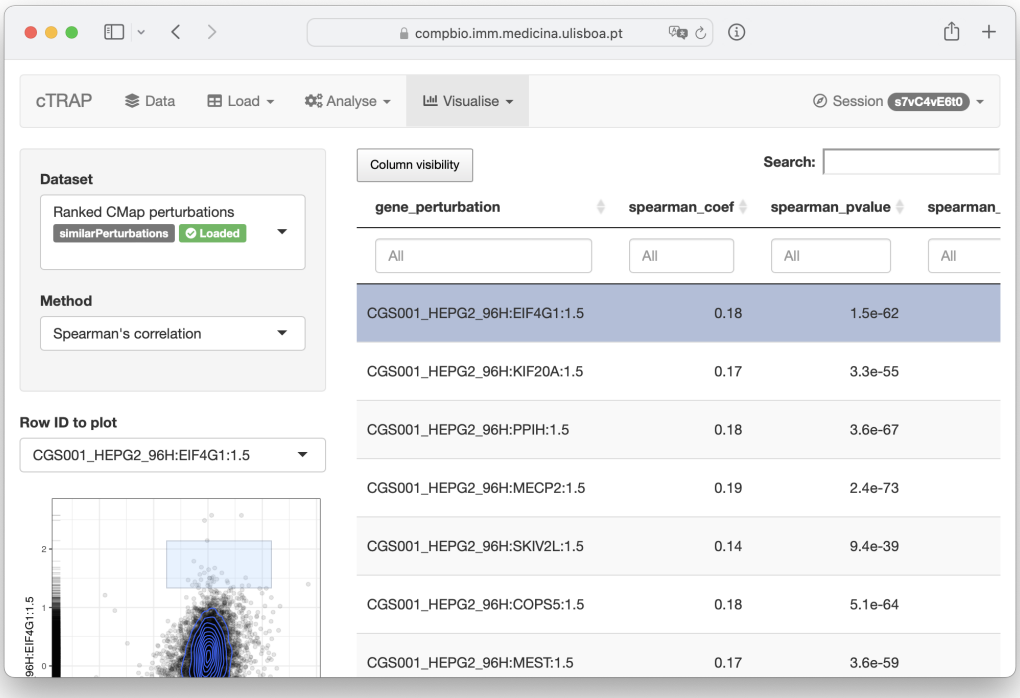


Figure 4.1: cTRAP global interface screenshot (21 Dec 2021).

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 drugs targeting altered genes by using datasets that featured both drug sensitivity and gene expression data for many cell lines. Additionally, cTRAP also allows to analyse the enrichment of molecular descriptor sets for compounds from NCI-60 and CMap. More recently, we developed a Shiny-based 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 NCI-60, CTRP and GDSC data Analyse enrichment for molecular descriptors of compounds 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

Understanding the biological mechanisms underlying uncharacterised phenotypes is crucial to unravel the physiological role of genes and the mechanism of action of compounds, alongside their therapeutic potential [15, 135–137, 210, 211]. Comprehending novel genetic and pharmacological perturbations – including the modulation of the expression of a gene or induction of an unknown compound in a cellular system – can be achieved by profiling genome-wide gene expression, allowing to analyse the whole cellular transcriptional response [210].

The gene expression profile of experimental phenotypic changes can be compared with a database of characterised transcriptomic signatures of known perturbagens, in order to infer putative molecular causes of the observed phenotype [15, 210]. This approach requires a large, heterogeneous and representative dataset containing gene expression profiles associated with known perturbagens across multiple cell lines [210].

Such is the case of the Connectivity Map (CMap), a repository of transcriptomic signatures of thousands of genetic and pharmacological perturbations of human cancer cell lines [15].

CMap data can be explored and compared with user-provided data via a collection of user-friendly web apps from the CMap and LINCS Unified Environment (`clue.io`) [15]. However, `clue.io` limits the maximum number of input genes for CMap queries (150 up-regulated and 150 down-regulated genes), is difficult to automate for downstream analyses and cannot be run using local computing resources. Furthermore, `clue.io` does not currently integrate with drug sensitivity datasets to further assist in pinpointing compounds that selectively target cells [211].

We thus developed cTRAP (Figure 4.2), an R package and web app that identifies potentially causal molecular perturbations by seamlessly comparing user-provided differential gene expression results with those available from CMap. cTRAP also sup-

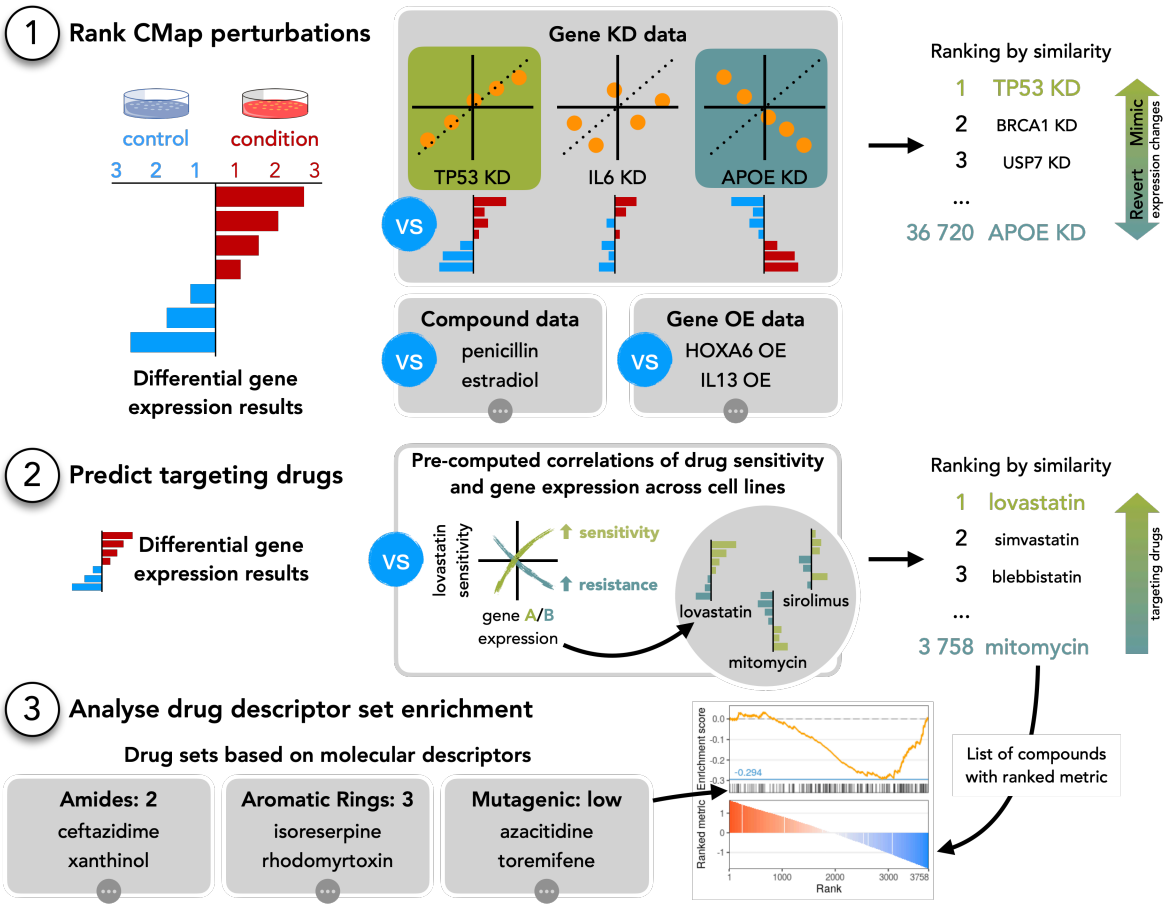


Figure 4.2: cTRAP analyses. cTRAP allows to perform three types of analyses: **(1) rank CMap perturbations** based on the similarity between their associated gene expression alterations and user-provided differential gene expression values, **(2) predict targeting drugs** by comparing user-provided differential gene expression values with matrices of correlation between gene expression and drug sensitivity data across human cell lines and **(3) analyse drug descriptor set enrichment** (v. main text) using the compound list from either the first or second analysis.

ports 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 [211]. In cTRAP, similarity between differential gene expression results is measured by gene set enrichment [15, 19] and correlation scores. Finally, cTRAP can also analyse a list of compounds resulting from previous analyses for the enrichment in sets of computed molecular descriptors (e.g., number of oxygen atoms or aromatic rings) from CMap and NCI-60 compounds, allowing to identify common chemical properties of compounds of interest.

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 list of CMap perturbations ranked by similarity or predict candidate drugs for targeting the associated phenotype. Moreover, cTRAP can also analyse the enrichment of drug sets in an ordered vector of compounds to identify common chemical characteristics (Figures 4.3 and 4.4).

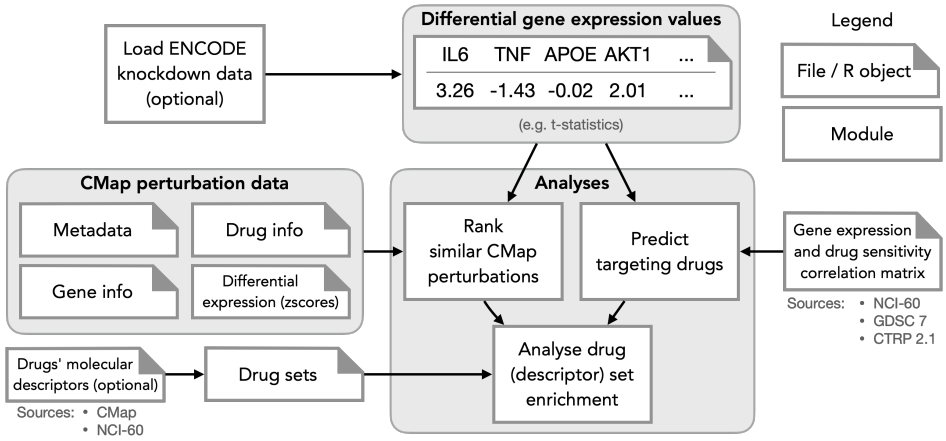


Figure 4.3: cTRAP workflow. A vector of differential gene expression values containing gene names is required to rank similar perturbations and to predict targeting drugs. Drug descriptor set enrichment analysis can then be performed based on those results. CMap perturbagen data, gene expression/drug sensitivity correlation matrices and molecular descriptors for drug sets can be automatically downloaded by cTRAP.

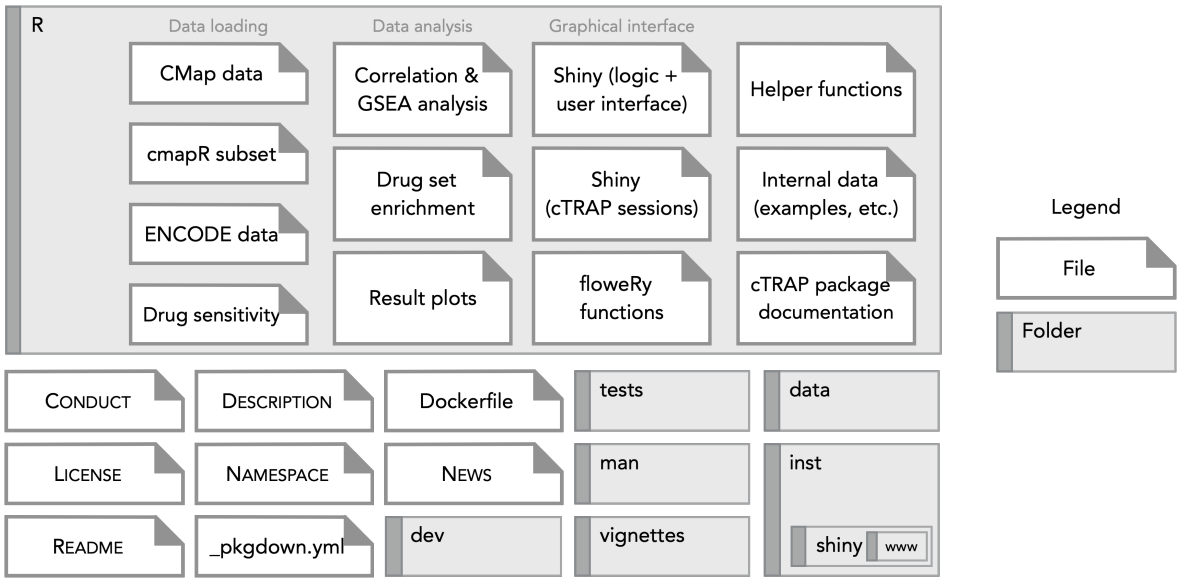


Figure 4.4: Visual representation of cTRAP's file structure. As usual in an R package, the `R` folder contains the scripts with cTRAP functions and data. `dev` is a custom folder that stores supporting scripts (e.g., test workflows and benchmarks); its contents are not included when building the R package.

4.2.1 ENCODE knockdown data

Using cTRAP, we can query and download ENCODE knockdown (and respective control) samples [212] for multiple cell lines, filter genes/samples with low coverage from gene expression data, convert from ENSEMBL gene identifiers to gene symbols, and perform differential gene expression analysis using `voom()`, `lmFit()` and `eBayes()` from the `limma` R package [156]. 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) associated with *EIF4G1* knockdown in HepG2 cells (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
```

```r
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, `filterCMapMetadata()` allows to filter the metadata based on selected perturbations types, cell lines, dosages and time points, allowing to specifically load only the desired data in downstream analyses. This information is passed to `prepareCMapPerturbations()` to download (if file

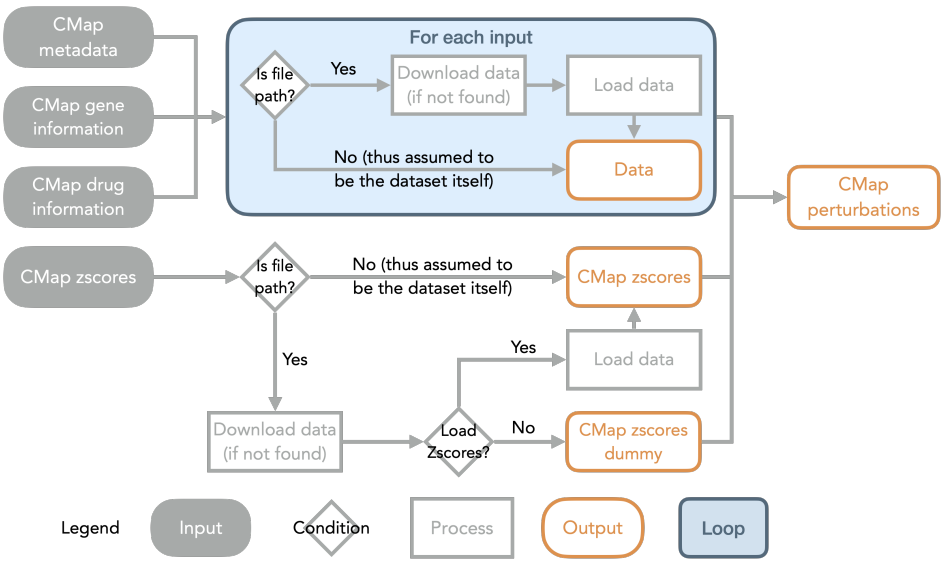


Figure 4.5: Loading data from CMap perturbations. Input arguments support either the data themselves (as data frames) or their respective file path. If the file path directs to a non-existing file, data are first downloaded and then saved to the given file path. To avoid high memory usage, CMap perturbations' differential expression z-scores (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.

is not found) and process CMap differential expression normalised z-scores (GCTX file) and gene and compound information (Figure 4.5). 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 normalised z-scores from select CMap perturbations against user-provided differential expression results, `rankSimilarPerturbations()` returns a table with ranked CMap perturbations. Ranks closer to the top 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 (described below). All three methods are run by default. For each method, the similarity scores are averaged across multiple cell lines for the same conditions (i.e., same exposure time, dose and induced compound or gene target) 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 [213], ultimately used to sort the CMap perturbations.

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

1. Order genes by 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), a composite and bi-directional version of the weighted Kolmogorov-Smirnov enrichment statistic (ES) [15] where GSEA is run for the most up- and down-regulated genes from the user's differential expression profile. The WTCS is the mean between ES_{top} and ES_{bottom}; however, WTCS = 0 if both sets have the same sign [15].

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 ex-

pression 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 4.5). 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 4.6). 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.

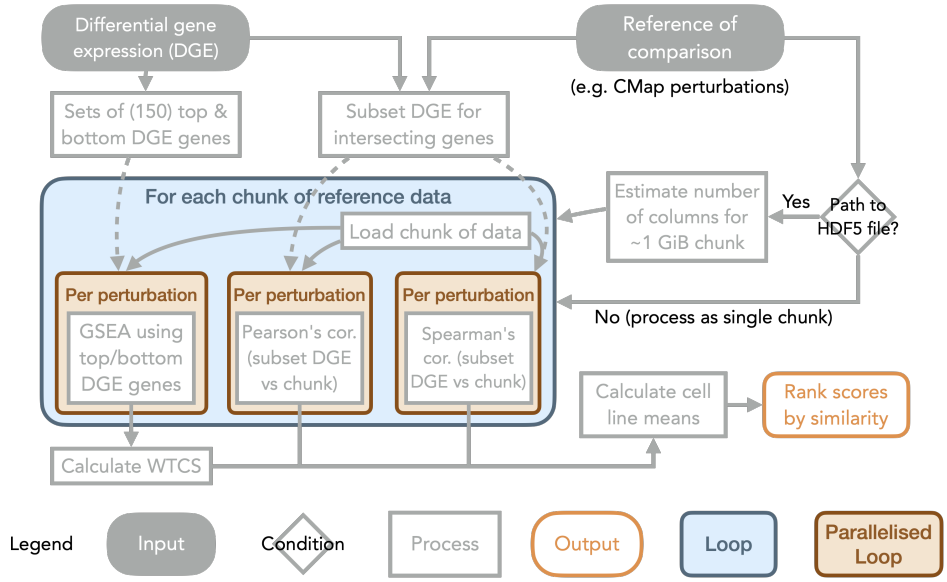


Figure 4.6: cTRAP similarity analysis. User-provided differential gene expression statistics are compared with reference data (e.g., differential expression z-scores of CMap perturbations) and ranked by similarity. If the reference is contained in an HDF5 file, the file is processed in 1 GiB chunks (by default) 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 of the results for a single CMap perturbation (examples shown in subsection 4.3.1: **Case study**).

[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.

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] (Table 4.2). For each source, the `prepareExpressionDrugSensitivityAssociation()` function performs the following:

1. download all the necessary data depending on given source;
2. perform Spearman's correlation (by default) across cell lines between the expression of each gene and the sensitivity 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.

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()`.

A positive correlation coefficient for a given gene and drug suggests a gene whose expression is associated with sensitivity to that drug across multiple cell lines. By calculating the susceptibility of genes for each annotated drug, we can then correlate this information with a given phenotype to rank compounds based on their potential to selectively target the queried (or similar) phenotypes, i.e., to selectively target phenotypes characterised by the overexpression of genes conferring susceptibility to the compounds.

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 WTCS (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 WTCS

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

Source	Compounds	Cell lines
NCI-60	21 738	60
GDSC 7	266	983
CTRP 2.1	545	823

(Figure 4.6). The ranks closer to the top 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 plots for a single 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 (examples shown in subsection 4.3.1: **Case study**).

4.2.4 Drug descriptor set enrichment analysis

Juan Carlos, a former member of the lab, computed drug descriptors (e.g., molecular weight and number of aromatic rings) for compounds from CMap and NCI-60 based on their three-dimensional (3D) and two-dimensional (2D) characteristics. These descriptors were uploaded to `compbio.imm.medicina.ulisboa.pt/public/cTRAP/` and the resulting files can be automatically downloaded and processed to R using `loadDrugDescriptors()`.

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

The `analyseDrugSetEnrichment()` function analyses the enrichment of the created drug descriptor sets in a named numeric vector or an object returned from `rankSimilarPerturbations()` or `predictTargetingDrugs()`. The GSEA-based enrichment analysis is internally performed using `fgsea::fgsea()` [214]. The resulting object can be plotted with `plot()`, showing a list of all results ordered by a given score or plots for a single 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 NCI-60 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 for 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 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` [145] and `pkgdown` [215]; and check for errors and warnins 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, 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.

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

4.3.1 Case study

To showcase cTRAP, we used RNA-seq data from *EIF4G1* shRNA knockdown experiments in the HepG2 cell line from the ENCODE project [212]. *EIF4G1* (Eukaryotic Translation Initiation Factor 4 Gamma 1) encodes for the EIF4G1 scaffolding protein that contains binding sites for subunits of the EIF4F protein complex, required to initiate cap-dependent translation [212, 216, 217]. EIF4G1 is involved in cancer cell proliferation and migration and *EIF4G1* knockdown has been suggested to impair tumourigenicity in prostate and epithelial cancer cells [216, 217].

Using cTRAP functions, we downloaded pre-processed gene quantification data for *EIF4G1* knockdown (ENCFF955TXI and ENCFF049UZV, two replicates) and controls (ENCFF657KMW and ENCFF726AVT) from ENCODE [212]. Afterwards, we quantilise-normalised the gene expression data with `voom` [156] and performed differential expression analysis with `limma` [156].

Comparison with CMap perturbations

We compared ENCODE's t-statistic for the *EIF4G1* knockdown expression changes against the normalised z-scores associated with CMap's knockdown and small molecule perturbations in HepG2[4]. The comparisons were performed using Spearman's correlation, Pearson's correlation and WTCS (using the default 150 up-regulated and 150 down-regulated genes). All results were ordered based on the rank product's rank.

Within CMap knockdown perturbations, the top result is expectedly the knockdown of *EIF4G1* in CMap data, working as a positive control (Table 4.3 and Figure 4.7).

Table 4.3: Top 10 most similar CMap HepG2 gene knockdown perturbations compared to *EIF4G1* knockdown profile. Results ordered by rank product's rank. Legend: *rho* is Spearman's coefficient and *r* is Pearson's coefficient.

Gene	*rho*	*r*	WTCS	Gene description
EIF4G1	0.18	0.19	0.49	eukaryotic translation initiation factor 4 gamma, 1
SKIV2L	0.14	0.18	0.51	Ski2 like RNA helicase
MECP2	0.19	0.18	0.39	methyl-CpG binding protein 2
KIF20A	0.17	0.18	0.46	kinesin family member 20A
MEST	0.17	0.19	0.42	mesoderm specific transcript
COPS5	0.18	0.18	0.42	COP9 signalosome subunit 5
PPIH	0.19	0.17	0.44	peptidylprolyl isomerase H
STAT1	0.18	0.19	0.38	signal transducer and activator of transcription 1
KIAA0196	0.19	0.18	0.35	KIAA0196
SQRDL	0.19	0.17	0.40	sulfide quinone reductase-like (yeast)

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

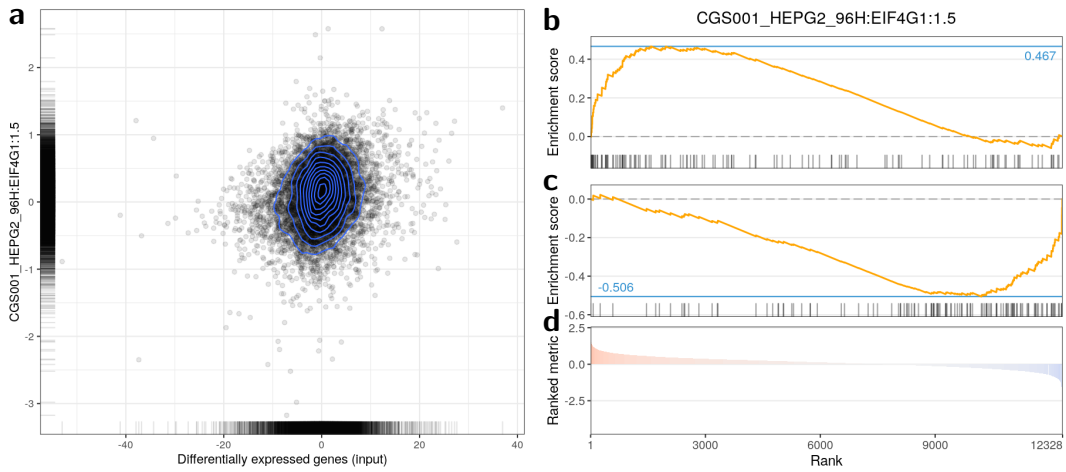


Figure 4.7: Comparison of the *EIF4G1* knockdown experiments in HepG2 cells between ENCODE and CMap. (a) Scatterplot of differential gene expression values of *EIF4G1* knockdown for common genes between ENCODE (t-statistics; X-axis) and CMap (normalised z-scores; Y-axis). (b-d) GSEA plot of the enrichment of the most up- (b) and down-regulated (c) genes from ENCODE *EIF4G1* knockdown relative to the ranked genes from the CMap perturbation (d). The genes are ranked (X-axis) based on the CMap normalised z-scores (Y-axis).

Afterwards, we compared the *EIF4G1* knockdown gene expression changes against those associated with CMap compound perturabtions in HepG2 (Table 4.4). The knockdown of *EIF4G1*, a gene that encodes for a translation initiation factor subunit relevant in the assembly of the EIF4F complex for initiation of cap-dependent translation [212, 216, 217], shows a similar expression profile to the perturbation caused by homoharringtonine, a translation elongation inhibitor.

Table 4.4: Top 10 most similar CMap HepG2 compound perturbations compared to the differential expression profile of *EIF4G1* knockdown in HepG2 cells from ENCODE. Results ordered by rank product's rank. Legend: *rho* is Spearman's coefficient, *r* is Pearson's coefficient and Time is the exposure time.

Compound	*rho*	*r*	WTCS	Dose	Time	Mechanism of action
homoharringtonine	0.23	0.21	0.44	10 µM	6 h	protein synthesis inhibitor
homoharringtonine	0.22	0.21	0.42	10 µM	24 h	protein synthesis inhibitor
BRD-K51592837	0.21	0.21	0.42	30 µM	24 h	
wortmannin	0.13	0.22	0.50	10 µM	24 h	PI3K inhibitor
AKT-inhibitor-IV	0.19	0.21	0.48	500 nM	24 h	
thioridazine	0.15	0.21	0.50	10 µM	24 h	dopamine receptor antagonist
BRD-K45534781	0.20	0.21	0.46	10 µM	24 h	
BRD-K79826210	0.18	0.21	0.46	30 µM	24 h	
anisomycin	0.22	0.21	0.40	10 µM	24 h	DNA synthesis inhibitor
puromycin	0.23	0.23	0.27	10 µM	24 h	protein synthesis inhibitor

Predicted targeting drugs

We can infer compounds targeting the phenotypes associated with the input differential expression profile by comparing them against gene expression and drug sensitivity

associations. Gene expression alteration profiles associated with increased drug sensitivity suggest the susceptibility of that (and similar) phenotypes for that compound, whereas profiles associated with decreased drug sensitivity suggest their resistance. We used the gene expression and drug sensitivity association based on CTRP 2.1 to infer targeting drugs for the *EIF4G1* knockdown's differential expression profile (Table 4.5). Compounds were 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 4.5: Top 10 CTRP targeting drugs compared to the differential expression profile of *EIF4G1* knockdown in HepG2 cells from ENCODE. Results ordered by rank product's rank. Legend: *rho* is Spearman's coefficient and *r* is Pearson's coefficient.

Compound	*rho*	*r*	WTCS	Mechanism of action
BRD-K75293299	0.14	0.13	0.33	product of diversity oriented synthesis
palmostatin B	0.12	0.12	0.36	inhibitor of acyl-protein thioesterase 1
FGIN-1-27	0.13	0.11	0.27	TSPO activator
niclosamide	0.054	0.067	0.39	inhibitor of STAT3 signaling
JW-55	0.094	0.091	0.29	inhibitor of tankyrase
968	0.10	0.080	0.21	inhibitor of glutaminase
bafilomycin A1	0.071	0.072	0.32	inhibitor of the vacuolar-type H+-ATPase
LY-2157299	0.13	0.12	0.0	TGFBR1 inhibitor
BRD-K49290616	0.084	0.079	0.26	product of diversity oriented synthesis
BRD8899	0.088	0.073	0.25	inhibitor of serine/threonine kinasase STK33

For drugs in common between the two analyses (i.e., between CTRP and CMap), we also compared the ranks of targeting potential and similarity with CMap perturbations towards the *EIF4G1* knockdown differential expression profile (Table 4.6 and Figure 4.8). This strategy highlights compounds that may both revert expression changes and target cells with a similar phenotype to the input differential expres-

Table 4.6: Drugs that may target the *EIF4G1* knockdown phenotype and revert the associated expression changes. Table of predicted CTRP 2.1 targeting drugs against similarity scores from CMap compound perturbations. Legend: Time is the exposure time.

Compound	Dose	Time	Mechanism of action
FGIN-1-27	10 µM	6 h	TSPO activator
simvastatin	10 µM	6 h	inhibitor of HMG-CoA reductase
lovastatin	40 µM	6 h	inhibitor of HMG-CoA reductase
blebbistatin	40 µM	6 h	inhibitor of myosin II ATPases
CAY10576	5 µM	6 h	inhibitor of IKK-epsilon
Compound 1541A	10 µM	6 h	activators of caspases 3, 6 and 7
NSC 74859	100 µM	6 h	inhibitor of STAT3
ML334 diastereomer	500 nM	24 h	inhibitor of KEAP1-NFE2L2 interaction
GMX-1778	100 nM	6 h	inhibitor of NAMPT
isonicotinohydroxamic acid	0.04 µM	24 h	inhibitor of HDAC6
pevonedistat	10 µM	6 h	inhibitor of Nedd-8 activating enzyme
GW-843682X	10 µM	6 h	inhibitor of PLK1 and PLK3
purmorphamine	40 µM	6 h	activator of smoothened receptor

sion [211]. However, the association between those ranks is not statistically significant according to Fisher's exact test. Nevertheless, *EIF4G1* depletion is associated with cell proliferation impairment by increasing levels of cell cycle inhibitor p27, thereby promoting autophagy [218], whereas top candidate targeting drugs purmorphamine [219] and blebbistatin [220] have been reported to inhibit autophagy.

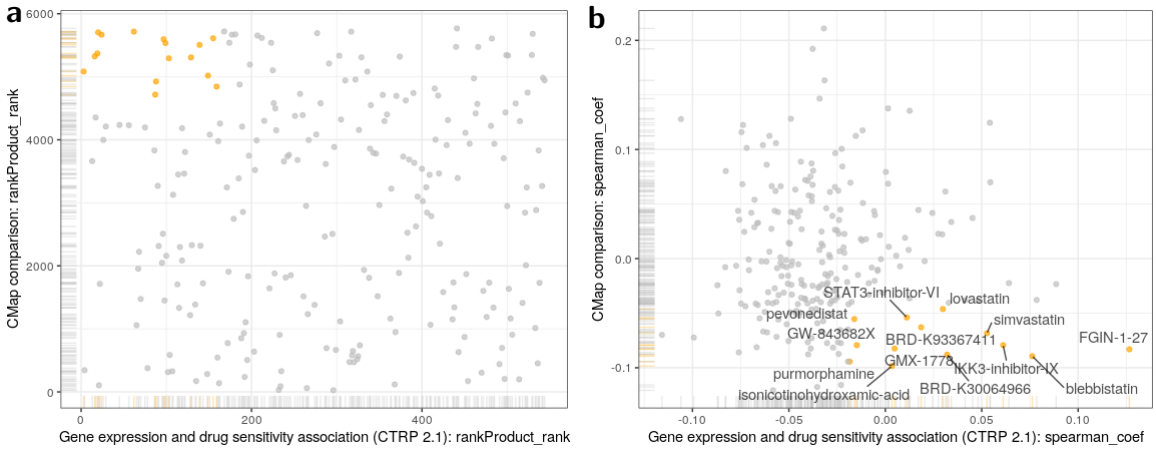


Figure 4.8: Drugs that may target the *EIF4G1* knockdown phenotype and revert the associated expression changes. Scatter plot of predicted CTRP 2.1 targeting drugs against similarity scores from CMap compound perturbations based on rank product's rank (a) and Spearman's coefficient (b) for both datasets. The Spearman's coefficient results are displayed for illustrative purposes. The 13 compounds amongst both the top 25% compounds that may revert *EIF4G1* knockdown and the top 25% targeting drugs are highlighted.

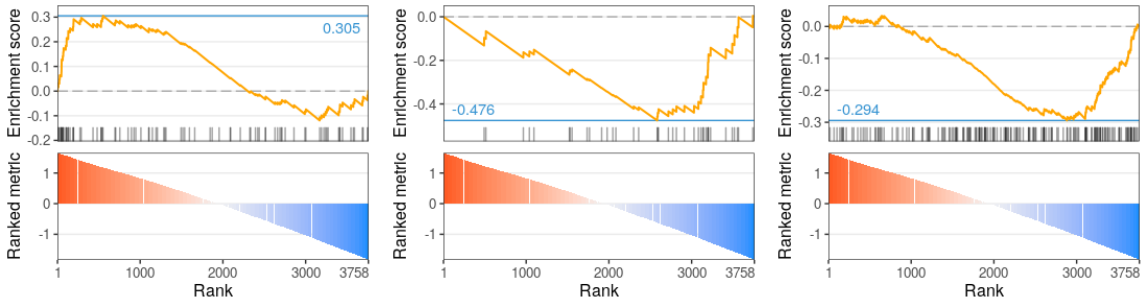
Drug set enrichment analysis

In order to elucidate a potential consensus in the modes of action of the most promising CMap chemical perturbations, we analysed their enrichment in 2D molecular descriptor sets (computed from all CMap compounds by collaborator Juan Carlos Gómez Verjan, INGER Mexico).

Enrichment scores are provided based on the signal of the ranked metric, yet ranks are only positive. As such, we centre the ranking so that the top-ranked compounds' values are positive, the bottom-ranked compounds' are negative and the middle-ranked compound is 0 (the additive inverse of the ranking is used so that the top-ranked compounds are positive after scaling). The values are also scaled to hint to the user that the values were transformed.

Based on these results (Figure 4.9), the number of rings closures associated with CMap chemical perturbations that mimic the *EIF4G1* knockdown phenotype seems lower (between 0 and 2) than for those perturbations that revert the phenotype (between 6 and 14), potentially indicating a chemical property associated with the phenotype that could be interesting to further study. The semblance between the drug sets associated with 2 rings closures and 0 or 1 is due to how cTRAP generates drug

sets (described in subsection 4.2.4: **Drug descriptor set enrichment analysis**). If these (or other) sets are usually associated in different contexts, this may suggest the need to further optimise the default generation of drug sets.



(a) Rings Closures: 6 to 14 **(b)** Rings Closures: 0 to 1 **(c)** Rings Closures: 2

Figure 4.9: Drug set enrichment analysis on CMap compound perturbations ranked by similarity to *EIF4G1* knockdown phenotype. GSEA plots displayed for select molecular descriptors with adjusted p-value < 0.05. The rank metric is the scaled and centred additive inverse of the rank product's rank.

4.3.2 Time and memory optimisation

cTRAP allows comparing user-provided differential expression results against the CMap perturbation z-scores that are contained in a 21GB GCTx file. 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 [221]. Instead of loading the whole GCTx file into memory, cTRAP 1.4 and newer versions minimise peak RAM usage by loading and processing user-selected subsets of perturbation z-scores (e.g., only compound perturbations) in chunks of 1 GiB (default).

Although processing data 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 WTCS 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 implement 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 NCI-60 gene expression and drug sensitivity correlation matrix is also large (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 NCI-60 data.

We benchmarked time and memory when ranking CMap perturbations based on Spearman's correlation, Pearson's correlation and WTCS using a development version of cTRAP 1.10 (commit `296f9b2`) and using as query the t-statistics for differential expression of *EIF4G1* knockdown in HepG2 cells from ENCODE. The number of CMap compound perturbations (241 258) is much higher than those of 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 4.10). For each chunk, we parallelised the calculations performed per perturbation and this can help to speed up runtime, 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 while consuming more resources.

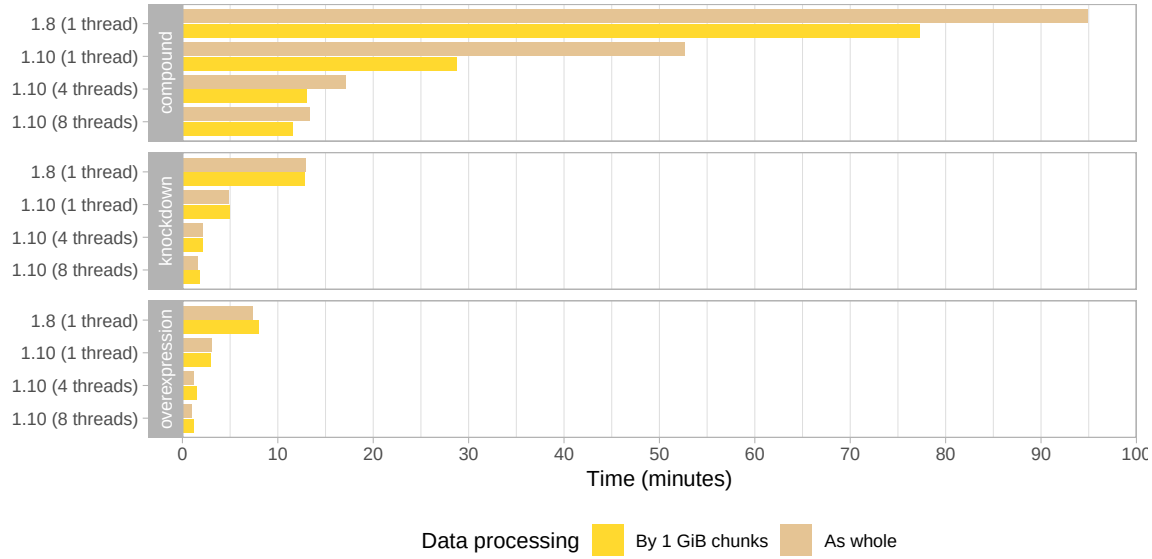


Figure 4.10: Time benchmark of CMap perturbation ranking. Times were measured for cTRAP 1.8 (1 thread only) and 1.10 (1, 4 and 8 threads) for over-expression, knockdown and compound perturbations whose data was loaded by 1 GiB chunks or as whole.

Benchmarked memory was annotated based on the different internal steps performed by cTRAP (Figure 4.11). 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.

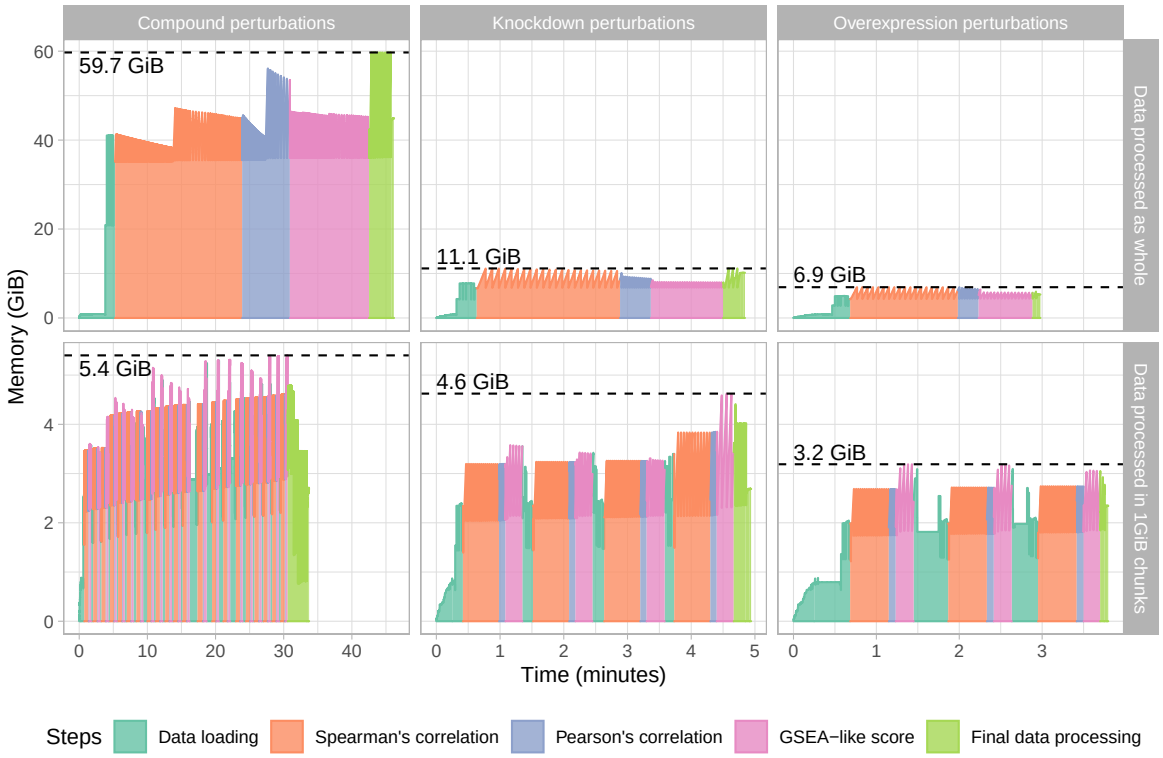


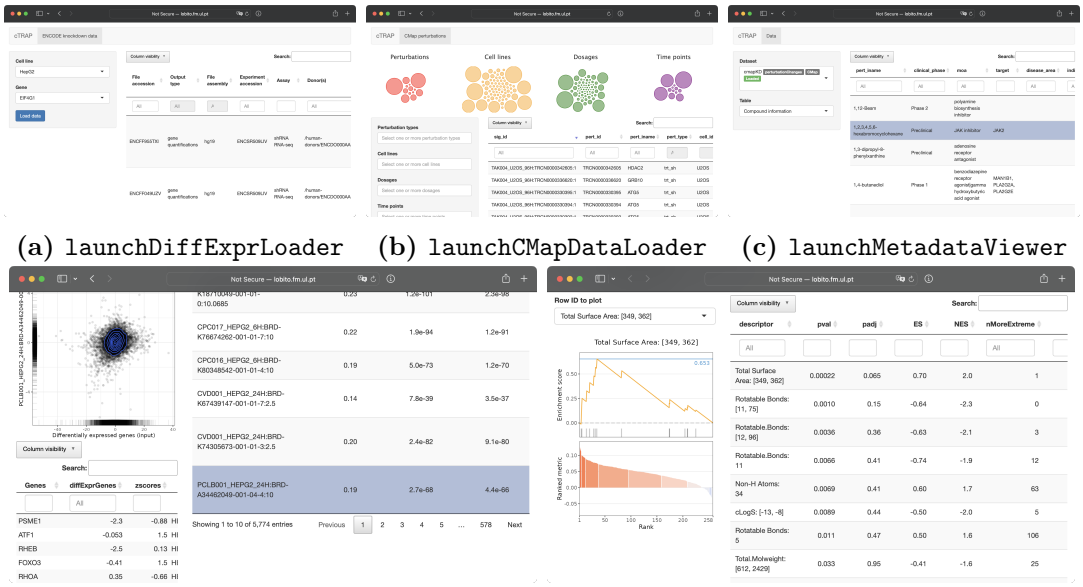
Figure 4.11: Memory benchmark of CMap perturbation ranking, annotated using the different steps performed by cTRAP. Memory was benchmarked when ranking compound, knockdown and overexpression perturbations against differential expression results. The maximum memory used for each condition is labelled.

4.3.3 Graphical interface

To assist users that may prefer graphical interfaces, most cTRAP functionality is exposed via 5 modular and independent Shiny-based functions (Figure 4.12):

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

Like usual R functions, these graphical interface 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



(a) `launchDiffExprLoader` (b) `launchCMapDataLoader` (c) `launchMetadataViewer`

(d) `launchResultPlotter` (e) `launchDrugSetEnrichmentAnalyser`

Figure 4.12: Screenshots of the modular cTRAP graphical interface functions.

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.

```
library(cTRAP)

# 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 Shiny-based 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, that implementation for a single interface would mean that each cTRAP session needed to be live and consume useful resources during the long-running cTRAP analyses, which does not properly scale in a web server with multiple users requiring heavy memory resources simultaneously. To circumvent this issue, long-running tasks can be optionally managed via job queues running in the background. In order for users to redeem their results once they finish calculating, we also implemented storage and retrieval of user session data in cTRAP.

User session data

Session data are saved in folders named after a random alphanumeric string (token) that uniquely identifies each session and can be downloaded as an RDS file – a list containing data for 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 to optimise resources and scalability

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

with multiple users[6].

cTRAP visitors are greeted with a welcome screen that allows them to create a new session or restore previous ones (Figure 4.13), a dialog that can be opened at any time from the session menu.

When creating a new session, a unique token is created. As soon as session-specific data are loaded, a new folder is created in the working directory and named after the session token (Figure 4.14). 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.

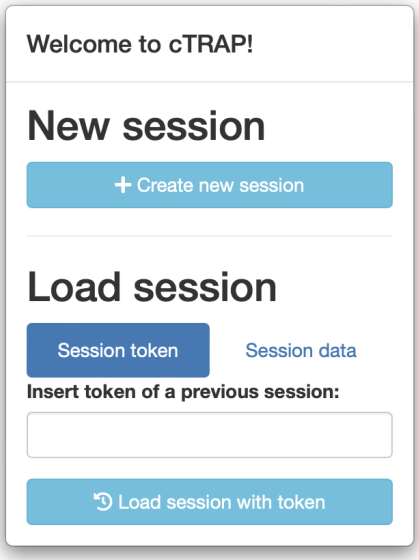


Figure 4.13: Welcome modal.

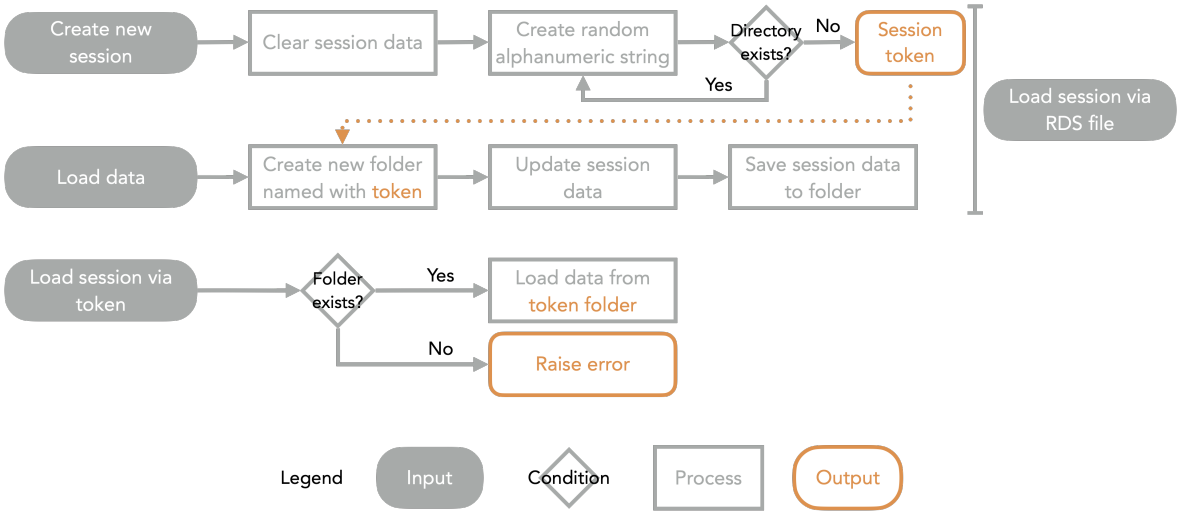


Figure 4.14: User session workflow. cTRAP allows to create new sessions or load a previous one via a given token or RDS file. When loading session via a RDS file, a new cTRAP session is created before loading the data from RDS file. Sessions can also be loaded using a token if any folder with such token exists in cTRAP's working directory.

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 4.14). In case the user uploads the RDS file of a previous session, cTRAP will load its contents into a new session (Figure 4.14).

[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.

Background tasks

When running a Shiny app, the user has to wait for all foreground tasks to finish before the app responds to user's commands. This issue can be mitigated by using 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 to take advantage of those resources until the whole session is terminated.

Alternatively, we can use light-weight job schedulers to manage and run large tasks in the background, such as **Celery**, a Python-based task queue manager that is complemented by **Flower**, a monitoring app that provides an HTTP API and graphical interface to manage Celery jobs.

Celery requires a `tasks.py` file detailing the tasks to run. As we intend to run R code submitted by cTRAP, we set up a Celery task that runs arbitrary R code by running the `Rscript` command via the `subprocess` Python module (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 assist job submission 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 the `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 [154]. Users can monitor the status of background tasks in their cTRAP session (Figure 4.15)[7].

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 4.15: Progress of Celery jobs in cTRAP, updated every 5 seconds. Job 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 removed). When job results are loaded, the respective dataset name is a link to access them (in blue).

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 processes finish, their output is saved into the session folder. If the user is actively using that session in the cTRAP website, the data are automatically loaded – replacing the previous *dummy* objects – and the user is informed of such via a notification

[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**.

in cTRAP (Figure 4.16). Otherwise, the 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.

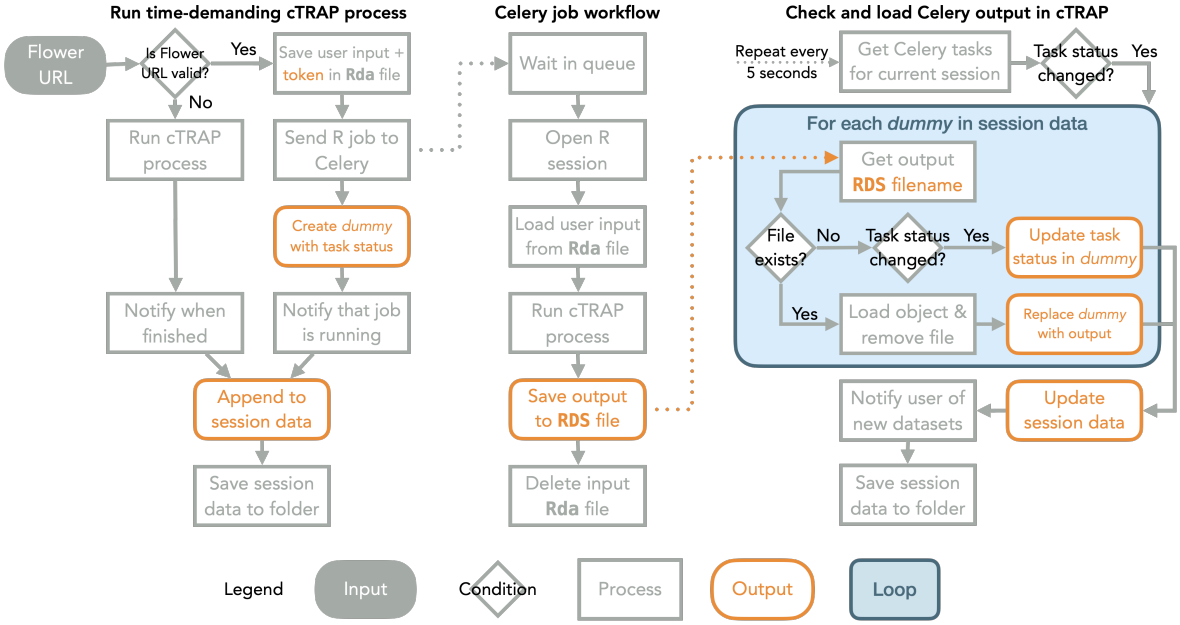


Figure 4.16: cTRAP process running in Celery. Time-demanding cTRAP processes can be run in the background using Celery/Flower. While running in Celery, the output of the cTRAP process is saved to the folder associated with the token of the user's session. When that specific session is active, all finished files are automatically loaded as part of the data session and the user is notified.

4.4 Conclusion

The analysis of uncharacterised phenotypes may help understanding relevant biological insights and developing novel therapies by comparing changes in gene expression against a large reference database – such as CMap – containing differential expression data associated with known perturbagens [15, 210]. The `clue.io` website is a collection of web apps to explore CMap data and to compare user-provided differential expression results against those from genetic and pharmacological CMap perturbations [15]. However, its shortcomings include limited queries with a maximum of 150 up-regulated and 150 down-regulated genes, poor automation with downstream analyses and lack of support to run with local computing resources.

We thus present cTRAP as a R package and web app to identify causal molecular perturbations from differential expression data, as well as pinpoint compounds that may promote or revert observed differences in gene expression. Besides the WTCS used by `clue.io` that only considers the top 150 up- and top 150 down-regulated genes, cTRAP also allows to measure similarity based on Pearson's and Spearman's correlations, thus

considering the expression of all common genes between datasets. Moreover, cTRAP allows to customise the number of up- and down-regulated genes in the set (by default, 150 like `clue.io`). When performing multiple comparisons, cTRAP also returns the rank product's rank as a ranked summary of selected comparison methods.

Unlike `clue.io`, cTRAP also uses publicly available drug sensitivity and gene expression data from NCI-60, GDSC and CTRP. The comparison of these data with user-provided differential expression profiles may help unravel compounds that selectively target cells. Furthermore, integrating such results with those from CMap comparisons allow to identify putative compounds associated with the queried expression changes and that may target cells with a similar profile to the user input [211].

Inspired by gene set enrichment analysis, cTRAP allows to analyse the enrichment of drug sets based on 2D and 3D molecular descriptors computed from NCI-60 and CMap compounds. This feature may assist in identifying common characteristics across ordered lists of compounds, therefore discerning candidate chemical properties to guide researchers in finding sets of similar compounds associated with the observed phenotype from previous results. However, the biological insights of (gene) set enrichment analysis is dependent on the sets used as input [222]. Although cTRAP allows to customise the drug set generation from the molecular descriptor data[8], the default drug sets could be further optimised and benchmarked for biological relevance.

Given the large input data from CMap (21GB of differential expression data), 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 usage of 5.4 GiB. This reduced memory demand is due to loading and processing CMap data in 1 GiB chunks by default. The speed and memory optimisations also apply when predicting targeting drugs based on the 3.3GB pre-processed dataset from NCI-60.

Besides its command-line interface, most of cTRAP's functionality can be accessed via multiple, modular graphical user interface functions that can be intertwined with R code. cTRAP also features a global user interface that is available as a web app at `compbio.imm.medicina.ulisboa.pt/cTRAP`. The web app allows users to download a RDS file to load cTRAP output into a local R session (e.g., to use with cTRAP locally or to perform other downstream analyses) or into the web app at a later time.

From our experience with psichomics, we expect the graphical interfaces of cTRAP to be popular among users that are less comfortable with coding in R. Nevertheless, the web app could benefit from 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. Unfortunately, Celery does not have built-in support for emails and this is not trivial to implement.

[8] As described in subsection 4.2.4: **Drug descriptor set enrichment analysis**.

In 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.

We hope that users will be able to successfully employ cTRAP in identifying candidate causal molecular perturbations and compounds to better understand the biological mechanisms underlying differences in gene expression alterations, as well as in prioritising 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 fingertips, 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 from big datasets. We want our tools to be intelligibly 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`.

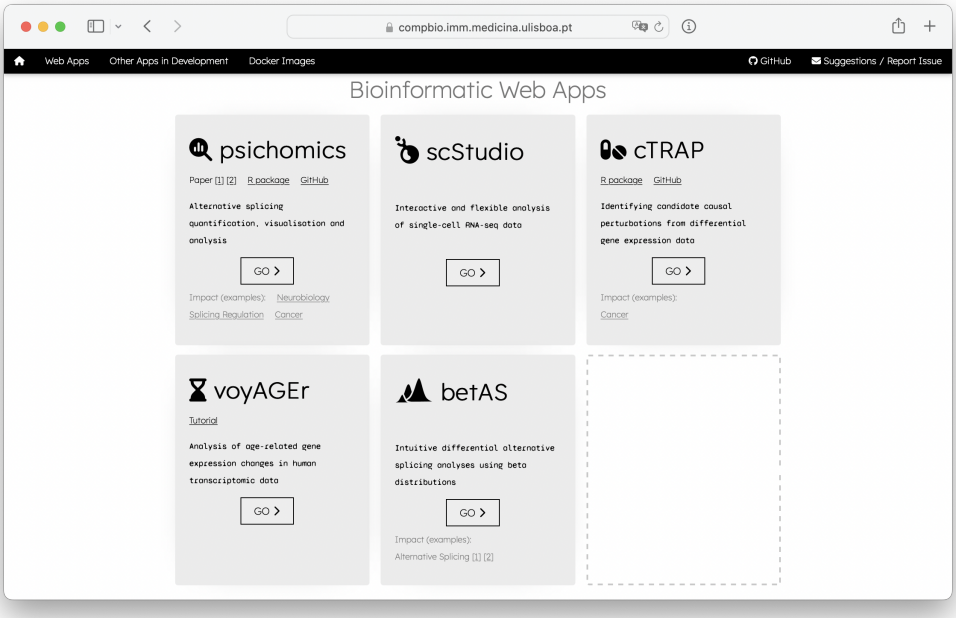


Figure 5.1: CompBio's homepage screenshot. List of hosted web apps (1 Sep 2022).

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 creating exploratory dashboards using the Shiny R package [160]. 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 [154,160]. 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 underlying web app as 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 [140]. 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.

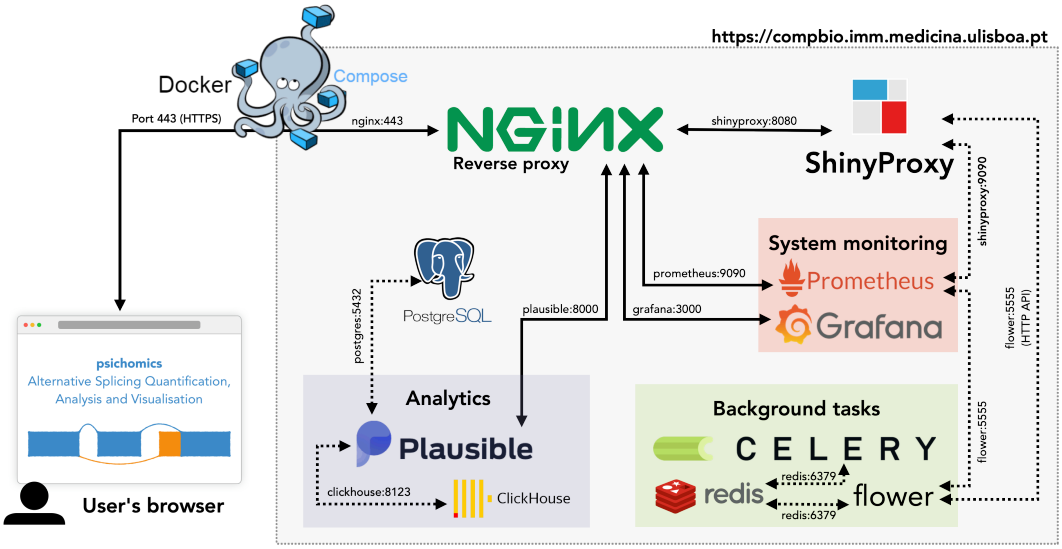


Figure 5.2: App server architecture is based on Docker Compose. All services are provided via Docker images and communicate with each other via a Docker-created network using the name of the service and a specific port (e.g., Nginx communicates with ShinyProxy via `shinyproxy:8080`). The groups (analytics,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

[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**.

resources depending on system usage.

The code can be run in Linux and macOS[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 or 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

[6]CompBio was not tested in Windows.

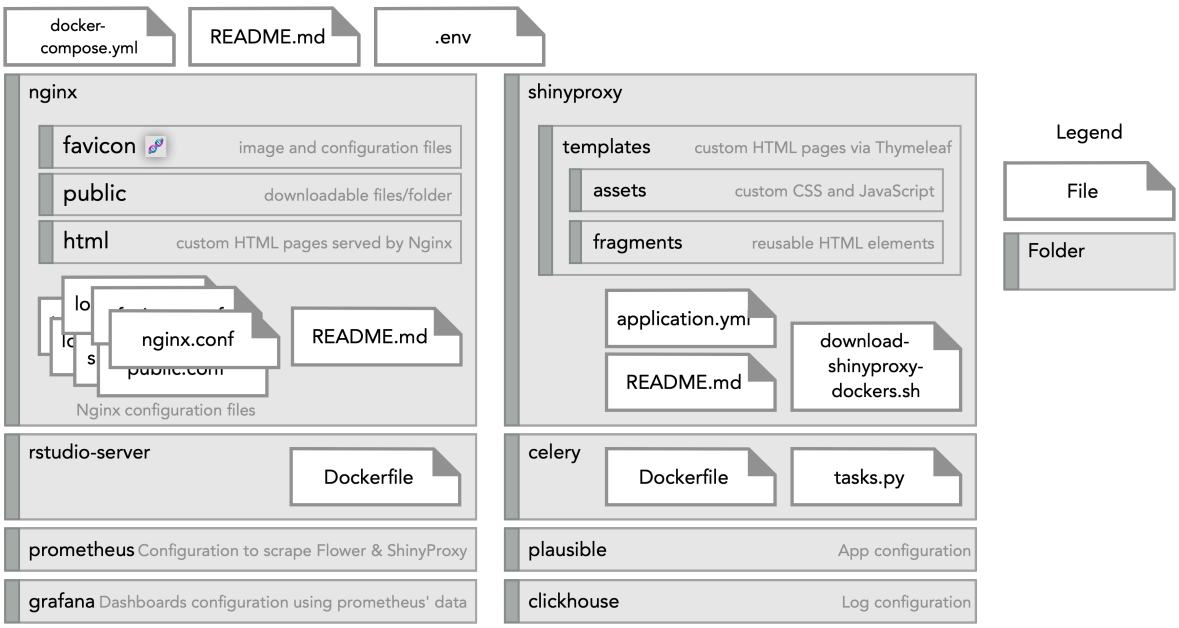


Figure 5.3: Visual representation of the file structure of the CompBio app server. Each folder contains files associated with a specific service: `rstudio-server` and `celery` contain Dockerfiles for building custom Docker images of the respective services; multiple `README.md` document the usage of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

available and portable. All services spawned in Docker Compose are based on Docker 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 are 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.

```
1  RSTUDIO_PASSWORD=rstudio_pass
2
3  POSTGRES_USER=postgres_user
4  POSTGRES_PASSWORD=postgres_pass
5
6  GRAFANA_USER=grafana_user
7  GRAFANA_PASSWORD=grafana_pass
8
9  PLAUSIBLE_EMAIL=someone@email.com
10 PLAUSIBLE_USER=plausible_user
11 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 [223, 224].

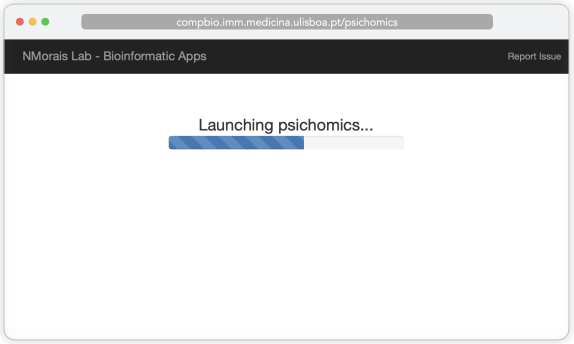


Figure 5.4: Progress bar displayed while psichomics loads (11 Nov 2021).

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 [223, 224].

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`.

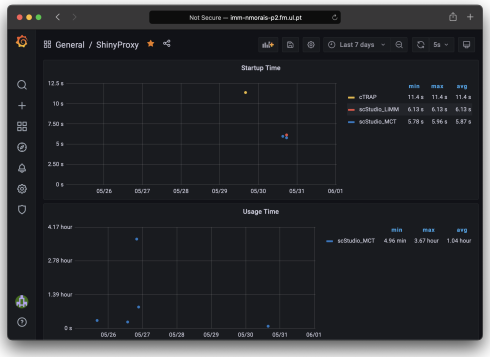
```
1  celery-ctrap:
2    container_name: celery-ctrap
3    build: ./celery
4    command: celery -A tasks worker -c5 -l info -E -n ctrap
5    volumes:
6      - ./celery:/celery:ro
7      - ../apps/cTRAP/sessions:/data
8    depends_on:
9      - redis
```

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).

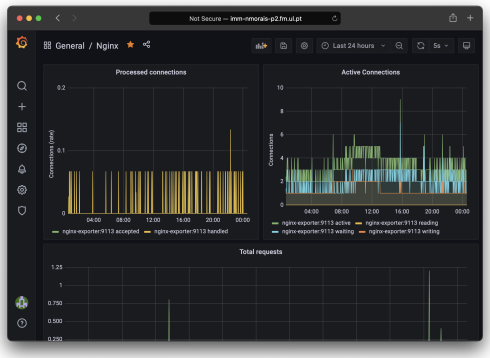
[10]More information in section 4.3.3: **Background tasks**.

(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 as the server's SQL database system, although currently no Shiny web apps in the server use it.

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 created and the project of my lifetime. It allowed me to develop an app based on multiple topics of my interest (e.g., transcriptomics, data visualisation, web technologies and user interface design), while providing a tool with a graphical interface for alternative splicing and gene expression analysis based on user-provided and public data.

psichomics is limited in the types of alternative splicing events profiled given that it only supports exon-exon junction read counts when quantifying alternative splicing. Support for other types of reads (exon–intron junction, exon body and intron body

counts) would be required to calculate intron retention and alternative 5′ and 3′ UTR events [178]. This issue is partially mitigated by allowing to import alternative splicing quantifications from user-provided tables and VAST-TOOLS [126, 127]. For future iterations of psichomics, users have also been requesting support to process the output from other popular alternative splicing quantification tools, including Whippet [173] and rMATS [128].

psichomics was employed in multiple peer-reviewed scientific articles, being used to perform genome-wide identification of differentially spliced events in TCGA data for a review article on alternative splicing in lung cancer [11], to assess mutation effects associated with the splicing of specific exons between GTEx brain and skin tissue data alongside VAST-TOOLS and experimental data [12], and to study how *IGF2* over-expression and reduced *SRSF3* splicing activity are negatively associated with the overall survival of liver cancer patients using TCGA data [14]. The use of psichomics in these publications demonstrates the usefulness of allowing users to access public, pre-processed data to analyse clinical, gene expression and alternative splicing data to complement other bioinformatic and experimental analyses.

6.2 cTRAP

Current `clue.io` tools allow to compare user-provided data with those from CMap, but have some limitations: they only consider a restricted input set of 150 up-regulated and 150 down-regulated genes, are difficult to automate, offer no option to run using local computing resources and have no integration with data from relevant sources that can benefit from similar analyses (e.g., drug sensitivity data from NCI-60, CTRP and GDSC). By overcoming such issues, we expect cTRAP will make it easier for users to identify candidate causal molecular perturbations of phenotypes and compounds, as well as in prioritising targeted therapeutic agents for disease-associated queries.

Thanks to working on psichomics, my knowledge of R was more mature and allowed me to flourish my creativity while developing cTRAP. In the optimisation department, it was challenging 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. Although their practical usefulness may be limited, their modular design made it relatively easy to create the intuitive global interface that culminated into the cTRAP web app. To further test command-line and graphical interfaces of cTRAP, we asked fellow scientists to use them while under cTRAP development and tweaked the interface and internal logic according to their feedback.

Creating the web app 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 still 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.

We are currently preparing a manuscript for publication in a peer-review journal. Nonetheless, the early availability of cTRAP in GitHub and Bioconductor led to users asking for the integration of differential gene expression results from single-cell analyses tools, such as Seurat [225]. Such integration with results from popular R packages could promote the use of cTRAP as a downstream analysis tool to further investigate their molecular data.

6.3 CompBio

Creating the app server to host web apps was an enormous challenge, but I am satisfied with the final result. With the exception of R/RStudio and Docker, I had to learn about all of the software stack 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. It was also interesting to configure the server to work in a testing environment so it is easy to test it on any machine before deploying to production.

Our app server currently hosts psichomics and cTRAP, among other web apps from my colleagues. In the future, we intend to host new web apps developed by the lab. In case user demand rises, we may need to increase the available RAM memory and disk space available to the server, which the current iMM infrastructure contemplates.

Although outside of the scope of this work, we decided to make CompBio codebase open-source, allowing other laboratories to repurpose this project, set up their own computing servers and deploy their favourite R and Python web apps. This could make it easier for users to quickly start using the developed web apps without the need to install R packages, Python libraries and their dependencies, as well as avoiding potential installation issues by leveraging Docker images.

[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.

6.4 PanAShé

There are many other projects that I have been part of during my PhD but unfortunately did not progress 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 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 presented, 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 for a PhD, psichomics helped me complete one of them: to create a useful tool to others' 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. Interactive Alternative Splicing Analysis of Human Stem Cells Using psichomics. In: Kidder BL, editor. Stem Cell Transcriptional Networks: Methods and Protocols. 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: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: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] Kumar D, Das M, Oberg A, Sahoo D, Wu P, Sauceda C, et al. Hepatocyte Deletion of IGF2 Prevents DNA Damage and Tumor Formation in Hepatocellular Carcinoma. Advanced Science. 2022;9(21):2105120. doi:https://doi.org/10.1002/advs.202105120.

[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. Proc Natl Acad Sci U S A. 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] Dalrymple GB. The age of the Earth in the twentieth century: a problem (mostly) solved. Geological Society, London, Special Publications. 2001;190(1):205–221. doi:10.1144/GSL.SP.2001.190.01.14.

[22] Dodd MS, Papineau D, Grenne T, Slack JF, Rittner M, Pirajno F, et al. Evidence for early life in Earth's oldest hydrothermal vent precipitates. Nature. 2017;543(7643):60–64. doi:10.1038/nature21377.

[23] 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.

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

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

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

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

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

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

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

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

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

[33] 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.

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

[35] Calabrese EJ. Muller's nobel prize research and peer review. Philos Ethics Humanit Med. 2018;13(1):15–15. doi:10.1186/s13010-018-0066-z.

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

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

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

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

[40] Gayon J. From Mendel to epigenetics: History of genetics. Comptes Rendus Biologies. 2016;339(7):225–230. doi:https://doi.org/10.1016/j.crvi.2016.05.009.

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

[42] Beadle GW, Tatum EL. Genetic Control of Biochemical Reactions in Neurospora. Proc Natl Acad Sci U S A. 1941;27(11):499–506. doi:10.1073/pnas.27.11.499.

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

[44] 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.

[45] 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.

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

[47] 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.

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

[49] 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:10.1016/S0021-9258(19)77302-5.

[50] Warner JR, Knopf PM, Rich A. A Multiple Ribosomal Structure In Protein Synthesis. Proc Natl Acad Sci U S A. 1963;49(1):122–129. doi:10.1073/pnas.49.1.122.

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

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

[53] 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.

[54] 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.

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

[56] 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:10.1016/0006-291X(60)90094-2.

[57] 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.

[58] 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:10.1016/0006-291X(60)90110-8.

[59] 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.

[60] 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.

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

[62] 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.

[63] 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.

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

[65] 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.

[66] Berget SM, Moore C, Sharp PA. Spliced segments at the 5′ terminus of adenovirus 2 late mRNA. Proc Natl Acad Sci U S A. 1977;74(8):3171–3175. doi:10.1073/pnas.74.8.3171.

[67] 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.

[68] 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.

[69] 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.

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

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

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

[77] 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.

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

[79] 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.

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

[81] 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.

[82] 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.

[83] 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.

[84] Lee Y, Rio DC. Mechanisms and Regulation of Alternative Pre-mRNA Splicing. Annu Rev Biochem. 2015;84:291–323. doi:10.1146/annurev-biochem-060614-034316.

[85] 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.

[86] 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.

[87] 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.

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

[89] 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.

[90] 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.

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

[92] 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.

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

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

[95] 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.

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

[97] 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.

[98] 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:10.1016/0022-2836(70)90057-4.

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

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

[101] Sanger F, Nicklen S, Coulson AR. DNA sequencing with chain-terminating inhibitors. Proc Natl Acad Sci U S A. 1977;74(12):5463–5467. doi:10.1073/pnas.74.12.5463.

[102] 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.

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

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

[105] 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.

[106] Higgins DG, Sharp PM. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. Gene.

1988;73(1):237–244. doi:10.1016/0378-1119(88)90330-7.

[107] 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:10.1016/S0022-2836(05)80360-2.

[108] 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.

[109] 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.

[110] 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.

[111] 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.

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

[113] 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.

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

[115] Kuska B. Beer, Bethesda, and Biology: How Genomics Came Into Being. J Natl Cancer Inst. 1998;90(2):93–93. doi:10.1093/jnci/90.2.93.

[116] 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.

[117] 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.

[118] 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.

[119] 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.

[120] 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. Proc Natl Acad Sci U S A. 1997;94(24):13057–13062. doi:10.1073/pnas.94.24.13057.

[121] 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. 2008;320(5881):1344–1349. doi:10.1126/science.1158441.

[122] 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.

[123] Andrews S, Krueger F, Segonds-Pichon A, Biggins L, Krueger C, Wingett S. FastQC: A quality control tool for high throughput sequence data.; 2019. Babraham Institute.

[124] 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.

[125] 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.

[126] 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.

[127] 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.

[128] 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.

[129] 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.

[130] 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.

[131] 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.

[132] 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.

[133] 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.

[134] 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.

[135] 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.

[136] 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.

[137] 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.

[138] 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.

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

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

[141] 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.

[142] Hewitt E. Semantic Software Design: A New Theory and Practical Guide for Modern Architects. O'Reilly Media; 2019.

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

[144] 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.

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

[146] 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.

[147] 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.

[148] 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.

[149] 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.

[150] 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.

[151] 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.

[152] Tidwell J, Brewer C, Valencia A. Designing Interfaces: Patterns for Effective Interaction Design. O'Reilly Media; 2019.

[153] 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:10.1016/j.paid.2019.109709.

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

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

[156] 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.

[157] 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.

[158] Wickham H. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York; 2016. Available from: https://ggplot2.tidyverse.org.

[159] Kunst J. highcharter: A Wrapper for the 'Highcharts' Library; 2022. Available from: https://CRAN.R-project.org/package=highcharter.

[160] 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.

[161] Eddelbuettel D, Balamuta JJ. Extending extitR with extitC++: A Brief Introduction to extitRcpp. The American Statistician. 2018;72(1):28–36. doi:10.1080/00031305.2017.1375990.

[162] Chang W, Luraschi J, Mastny T. profvis: Interactive Visualizations for Profiling R Code; 2020. Available from: https://CRAN.R-project.org/package=profvis.

[163] Mersmann O. microbenchmark: Accurate Timing Functions; 2021. Available from: https://CRAN.R-project.org/package=microbenchmark.

[164] Wickham H, Hester J, Chang W, Bryan J. devtools: Tools to Make Developing R Packages Easier; 2021. Available from: https://CRAN.R-project.org/package=devtools.

[165] Wickham H, Danenberg P, Csárdi G, Eugster M. roxygen2: In-Line Documentation for R; 2022. Available from: https://CRAN.R-project.org/package=roxygen2.

[166] 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.

[167] Saraiva-Agostinho N. Bioinformatics application for analysis and visualisation of alternative splicing in cancer. Faculdade de Ciências, Universidade de Lisboa; 2016.

[168] 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.

[169] 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.

[170] 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.

[171] 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.

[172] 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.

[173] 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.

[174] 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.

[175] 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.

[176] 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.

[177] 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.

[178] 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.

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

[180] 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.

[181] 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.

[182] 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.

[183] 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.

[184] 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.

[185] Caravela BSR. Alternative splicing analysis for finding novel molecular signatures in renal carcinomas. Instituto Superior Técnico, Universidade de Lisboa; 2015.

[186] 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.

[187] 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.

[188] 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.

[189] 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.

[190] 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.

[191] 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.

[192] 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.

[193] 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.

[194] 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.

[195] 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.

[196] 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:10.1016/j.ceb.2017.12.002.

[197] 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.

[198] 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.

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

[200] 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.

[201] 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.

[202] 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.

[203] 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.

[204] 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.

[205] 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:10.1038/emboj.2010.195.

[206] Zhang Z, Sun J. Interval censoring. Statistical Methods in Medical Research. 2010;19(1):53–70. doi:10.1177/0962280209105023.

[207] 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.

[208] 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.

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

[210] Hughes TR, Marton MJ, Jones AR, Roberts CJ, Stoughton R, Armour CD, et al. Functional Discovery via a Compendium of Expression Profiles. Cell. 2000;102(1):109–126. doi:10.1016/S0092-8674(00)00015-5.

[211] de Almeida BP, Vieira AF, Paredes J, Bettencourt-Dias M, Barbosa-Morais NL. Pan-cancer association of a centrosome amplification gene expression signature with genomic alterations and clinical outcome. PLOS Computational Biology. 2019;15(3):1–31. doi:10.1371/journal.pcbi.1006832.

[212] Luo Y, Hitz BC, Gabdank I, Hilton JA, Kagda MS, Lam B, et al. New developments on the Encyclopedia of DNA Elements (ENCODE) data portal. Nucleic Acids Research. 2019;48(D1):D882–D889. doi:10.1093/nar/gkz1062.

[213] Breitling R, Armengaud P, Amtmann A, Herzyk P. Rank products: a simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments. FEBS Letters. 2004;573(1-3):83–92.

[214] Sergushichev AA. An algorithm for fast preranked gene set enrichment analysis using cumulative statistic calculation. bioRxiv. 2016;doi:10.1101/060012.

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

[216] Tu L, Liu Z, He X, He Y, Yang H, Jiang Q, et al. Over-expression of eukaryotic translation initiation factor 4 gamma 1 correlates with tumor progression and poor prognosis in nasopharyngeal carcinoma. Molecular Cancer. 2010;9(1):78. doi:10.1186/1476-4598-9-78.

[217] Jaiswal PK, Koul S, Shanmugam PST, Koul HK. Eukaryotic Translation Initiation Factor 4 Gamma 1 (eIF4G1) is upregulated during Prostate cancer progression and modulates cell growth and metastasis. Scientific Reports. 2018;8(1):7459. doi:10.1038/s41598-018-25798-7.

[218] Ramírez-Valle F, Braunstein S, Zavadil J, Formenti SC, Schneider RJ. eIF4GI links nutrient sensing by mTOR to cell proliferation and inhibition of autophagy. J Cell Biol. 2008;181(2):293–307. doi:10.1083/jcb.200710215.

[219] Jimenez-Sanchez M, Menzies FM, Chang YY, Simecek N, Neufeld TP, Rubinsztein DC. The Hedgehog signalling pathway regulates autophagy. Nature Communications. 2012;3(1):1200. doi:10.1038/ncomms2212.

[220] Wang G, Wang T, Hu Y, Wang J, Wang Y, Zhang Y, et al. NMMHC IIA triggers neuronal autophagic cell death by promoting F-actin-dependent ATG9A trafficking in cerebral ischemia/reperfusion. Cell Death Dis. 2020;11(6):428. doi:10.1038/s41419-020-2639-1.

[221] 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.

[222] Davies MN, Meaburn EL, Schalkwyk LC. Gene set enrichment; a problem of pathways. Brief Funct Genomics. 2010;9(5-6):385–390. doi:10.1093/bfgp/elq021.

[223] 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.

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

[225] Hao Y, Hao S, Andersen-Nissen E, Mauck I William M, Zheng S, Butler A, et al. Integrated analysis of multimodal single-cell data. Cell. 2021;184(13):3573–3587.e29. doi:10.1016/j.cell.2021.04.048.