FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Towards Reproducible and Privacy-preserving Analyses Across Federated Repositories for Omics data

Rafael Lima Joia

Mestrado em Engenharia de Software

Supervisor: João Correia Lopes

Co-Supervisor: Artur Rocha

July 23, 2021

Towards Reproducible and Privacy-preserving Analyses Across Federated Repositories for Omics data

Rafael Lima Joia

Mestrado em Engenharia de Software

July 23, 2021

Abstract

Even when duly anonymized, health research data has the potential to be disclosive and therefore requires special safeguards according to the European General Data Protection Regulation (GDPR). Furthermore, the incorporation of FAIR principles (Findable, Accessible, Interoperable, Reusable) for a more favorable reuse of existing data, calls for an approach where sensitive data is kept locally and only metadata and aggregated results are shared. Additionally, since central pooling is discouraged by ethical, legal, and societal issues, it is more frequent to observe maturing data management frameworks, and platforms adopting the federated approach.

Current implementations of privacy-preserving analysis frameworks seem to be limited when data becomes very large (millions of rows, hundreds of variables). Biological samples data, collected by high-throughput technologies, such as Next Generation Sequencing (NGS), and processed by computational workflows known as bioinformatics pipelines, are examples of this kind of data (commonly known as Omics data). The reproducibility of these pipelines is hard and it is often underestimated. Nevertheless, it is important to generate trust in scientific results, and therefore, is fundamental to know how these Omics data were generated or obtained.

This work will leverage the promising results of current open-source implementations for distributed privacy-preserving analyses, while aiming at generalizing the approach and addressing some of their shortcomings, including the reproducibility concerns.

The results were promising, seeing that the privacy-preserving analysis was effective when using the DataSHIELD framework in conjunction with the "resource R" package. We also concluded that the adoption of specialized DataSHIELD packages for Omics analyses, such as dsOmics, is a viable pathway to leverage the privacy-preserving for Omics data. To address the reproducibility challenges, we defined a relational database model to represent the steps, commands and operations executed by the bioinformatics pipelines. The proposed reproducible relation model can afford the traceability of bioinformatics pipelines very well, but this model alone does not guarantee a full reproducible ecosystem, since it does not solve the platform isolation problem. It can only be guaranteed when combining reproducible tools that offer built-in support for containers, such as Nextflow or Snakemake, and a set of values and good practices.

We concluded that the proposed solution would be a viable option for privacy-preserving analysis using Omics data. In contrast, the proposed pipeline reproducibility model must be improved or incorporated into existing reproducible pipeline tools.

Keywords: health research data, federated repositories, privacy-preserving analyses, FAIR principles, TRUST principles, GDPR, Omics data, traceability, reproducibility.

Resumo

Os dados de pesquisas em saúde, mesmo quando devidamente anonimizados, têm o potencial de ser reveladores e, portanto, requerem salvaguardas especiais de acordo com o Regulamento Geral Europeu de Proteção de Dados (GDPR). Por outro lado, a incorporação dos princípios FAIR (Findable, Accessible, Interoperable, Reusable) para a reutilização mais favorável dos dados existentes exige uma abordagem em que os dados privados sejam mantidos localmente e apenas metadados e resultados agregados sejam compartilhados. Adicionalmente, como o agrupamento central de dados é desencorajado por questões éticas, legais e sociais, é mais frequente observar frameworks de gerenciamento de dados e plataformas adotando uma abordagem federada.

As implementações atuais de frameworks de análise de preservação de privacidade parecem ser limitadas quando o volume de dados se torna muito grande (milhões de linhas, centenas de variáveis). Dados de amostras biológicas, coletadas por tecnologias de alto desempenho, como as Next Generation Sequence (NGS), e processadas por workflows computacionais conhecidos como pipelines bioinformáticos, são exemplos deste tipo de dado (comumente conhecidos como dados ômicos). A reprodutibilidade desses pipelines é difícil e muitas vezes subestimada. No entanto, ela é importante para gerar confiança nos resultados científicos e, portanto, é fundamental saber como esses dados ômicos foram gerados ou obtidos.

Este trabalho aproveitará os resultados promissores das implementações atuais de código aberto para análises distribuídas de preservação de privacidade, ao mesmo tempo que visa generalizá-las, abordando algumas de suas deficiências, includindo preocupacões sobre reprodutibilidade.

Os resultados foram promissores, visto que a análise de preservação de privacidade foi eficaz ao usar a estrutura DataSHIELD em conjunto com o pacote "resource R". Também concluímos que a adoção de pacotes DataSHIELD especializados para análises que envolvam dados ômicos é um caminho viável para alavancar a preservação da privacidade deste tipo de dado. Para enfrentar os desafios de reprodutibilidade, definimos um modelo de banco de dados para representar as etapas, comandos e operações executadas pelos pipelines bioinformáticos. O modelo relacional proposto pode permitir a rastreabilidade dos pipelines muito bem, mas esse modelo sozinho não garante um ecossistema totalmente reprodutível, uma vez que não resolve o problema de isolamento das plataformas. Isto só pode ser garantido ao combinar ferramentas que oferecem suporte integrado para contêineres, como Nextflow ou Snakemake, e um conjunto de valores e boas práticas.

Concluímos que a solução proposta seria uma opção viável para análise de preservação de privacidade usando dados ômicos. Em contraste, o modelo proposto de reprodutibilidade deve ser melhorado ou incorporado às ferramentas existentes de reprodutibilidade de pipelines.

Keywords: dados de pesquisa em saúde, repositórios federados, análises com preservação de privacidade, princípios FAIR, princípios TRUST, GDPR, dados ômicos, rastreabilidade, reprodutibilidade.

Acknowledgements

First, I would like to thank my advisor, professor João Correia Lopes, and my co-advisor, researcher Artur Rocha, for the teachings, discussions, and suggestions that guided this work's preparation.

I would like to thank my friends at work, José Pedro Ornelas, Gonçalo Campos Gonçalves, and Alexandre Almeida Costa, people of high knowledge, dynamism, and who at no time refused to help me develop ideas and make suggestions for improvements.

My colleagues received me at INESC TEC very well. I feel privileged to be part of such a competent and talented team. I learn from them every day.

I would like to thank my wife, Paula Ferraz de Oliveira, for her dedication to our family while I was involved in this work. We became parents during the period of development of this work. It was challenging, but my inseparable wife was able to accommodate all our needs in all that was required.

I thank all colleagues of the Master's in Software Engineering at FEUP, companions on this journey. I wish them every success in their careers.

Author

"I have a great desire to improve always.
Improving is what makes me happy."

Ayrton Senna

Contents

1 Introduction **1**
- 1.1 Problem Description and Motivation . . . 1
- 1.2 Objectives . . . 2
- 1.3 Structure . . . 2

2 Background **3**
- 2.1 Omics Data . . . 3
- 2.2 Bioinformatics Pipelines . . . 5
- 2.3 General Data Protection Regulation . . . 6
- 2.4 FAIR Principles . . . 8
- 2.5 TRUST Principles . . . 9
- 2.6 Federated Data Repositories . . . 11
- 2.7 Non-disclosive Analysis . . . 12
- 2.8 Summary . . . 13

3 Related Work **15**
- 3.1 Privacy-preserving Analyses solutions . . . 15
  - 3.1.1 The iReceptor+ platform . . . 15
  - 3.1.2 The OBiBa solution . . . 21
  - 3.1.3 The MOLGENIS solution . . . 30
  - 3.1.4 The GA4GH solutions . . . 31
- 3.2 Reproducible Bioinformatics Pipeline Tools . . . 34
- 3.3 Summary . . . 36

4 Proposed solution **37**
- 4.1 Requirements . . . 37
- 4.2 Methodology . . . 40
- 4.3 Architecture . . . 42
- 4.4 Reproducibility: Relational model . . . 45
- 4.5 Summary . . . 47

5 Implementation **49**
- 5.1 Introduction . . . 49
- 5.2 Setup . . . 49
- 5.3 Implementation . . . 50
  - 5.3.1 Evaluating the Resources to access AIRR-Rearrangement data . . . 56
  - 5.3.2 Accessing the pipeline execution metadata . . . 57
- 5.4 Summary . . . 59

6 Conclusions and Future Work **61**
6.1 Results 61
6.2 Future Work 62

References **65**

A Reproducibility: Relational model **67**

B Registers generated in a simulated pipeline execution **69**

C Docker stack for the database pipeline **73**

List of Figures

2.1 Omics technologies. . . . 4
2.2 AIRR-Seq Pipeline example. . . . 5

3.1 The High-Level iReceptor Plus Platform Architecture. . . . 16
3.2 Immune Repertoire Sequencing. . . . 18
3.3 An example of a sequencing instrument. . . . 19
3.4 An extracted section of a raw sequencing file that follows the .FASTQ format. . . . 19
3.5 A typical MiXCR pipeline. . . . 20
3.6 From data collection to data federation via data harmonization in a study consortia. 22
3.7 VJ Gene Distribution. . . . 23
3.8 V-J Gene Distribution with Treemap visualization. . . . 24
3.9 V-J Gene Distribution with Spectratyping visualization. . . . 24
3.10 Benchmarking and Feasibility of V-J Gene Distribution. . . . 25
3.11 Clonotypes in order of CDR3 length (left), Abundance of Clonotypes (middle), Number of Unique clonotypes per Sample (right). . . . 25
3.12 Most abundant cell clonotypes (left) vs Least prolific clonotypes (right). . . . 26
3.13 Benchmarking and Feasibility of Clone Frequency. . . . 26
3.14 Repertoire overlap. . . . 27
3.15 Example of K-mers analysis based on a k size = 5, including frequency matrix. . . . 28
3.16 Benchmarking and Feasibility of Motif Extraction. . . . 28
3.17 AIRR Data Commons datasets: millions of sequences per subject. . . . 28
3.18 Resource R package architecture 29
3.19 MOLGENIS overview and its modular architecture. . . . 30
3.20 The GA4GH federated ecosystem. . . . 32
3.21 The MME data repositories. . . . 33

4.1 A mind map that represents a complex Bioinformatics pipeline. . . . 41
4.2 Coral Architecture. . . . 43
4.3 Deployment Architecture. . . . 44
4.4 Database schema to store the Pipeline's execution metadata. . . . 45

5.1 Setup configuration using pipeline outputs as resources. . . . 50
5.2 Resource properties on Opal. . . . 52
5.3 Alignments and Clones datasets registered as resources in Opal software. . . . 53
5.4 ADC API endpoints. . . . 56
5.5 An example of HTTP request to the v1/rearrangement endpoint. . . . 57
5.6 A Resource of the HTTP category. . . . 58

A.1 Reproducibility - Relational model. . . . 68

C.1 Adminer - Login page. 79

List of Tables

2.1 Comparison of the Pipeline Annotation of *Leishmania infantum* Genome Executed Across Different Platforms. 6

3.1 Diversity of Scientific Pipeline Tools. 35

4.1 Reproducibility of a Bioinformatics Pipeline 46

B.1 Data Processing Registers. 69
B.2 Processing Step Registers. 70
B.3 File Type Registers. 70
B.4 File Registers. 70
B.5 Input Files Registers. 70
B.6 Output Files Registers. 70
B.7 Tool Registers. 71
B.8 Command Registers. 71
B.9 Command Processing Step Registers. 71
B.10 Command Option Registers. 71

Listings

5.1 MiXCR pipeline basic example. 51
5.2 Basic R script to access the resources. 53
5.3 Output of the R script. 54
5.4 Example of how to access a PostgreSQL database on R. 57
C.1 Docker-Compose to deploy the suggested database model. 73
C.2 DDL script to create the database. 74

Abbreviations

ADC	AIRR Data Commons
AIRR	Adaptive Immune Receptor Repertoire
API	Application Programming Interface
BCR	B-cell receptor sequences
CDR	Complementarity-determining regions
CIHR	Canadian Institutes of Health Research
CRWG	AIRR Common Repository Working Group
CWL	Common Workflow Language
DDL	Data Definition Language
DTA	Data Transfer Agreements
DSL	Domain Specific Language
DWG	GA4GH Data Working Group
EOSC	European Open Science Cloud
FAIR	Findable, Accessible, Interoperable, Reusable
GA4GH	Global Alliance for Genomics and Health
GDPR	European General Data Protection Regulation
HPC	High-Performance Computing
HTTPS	Hypertext Transfer Protocol Secure
IMGT	international ImMunoGeneTics information system
MME	Matchmaker Exchange
MRI	Magnetic Resonance Imaging
NCBI	National Center for Biotechnology Information of the United States
NGS	Next-generation sequencing
OAIS	Open Archival Information System
OMICS	in biology, refers to the names that end in the suffix -omics, such as genomics, proteomics, metabolomics, and glycomics
PCR	Polymerase chain reaction
REST	Representational State Transfer
REWG	GA4GH Regulatory and Ethics Working Group
SDC	Statistical Disclosure Control
SDL	Statistical Disclosure Limitation
SLURM	Simple Linux Utility for Resource Management
SRA	Sequence Read Archive
SSH	Secure Shell
TCR	T-cell receptor sequences
TRUST	Transparency, Responsibility, User focus, Sustainability, Technology
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WDL	Workflow Definition Language

Chapter 1

Introduction

The exponential growth of patient data is shaking healthcare in several ways. Wearable fitness trackers, apps to manage pregnancy, calorie ingestion, mental health, medication, genealogical DNA test and drug control, all collecting health data. As a result, over the past fifteen years, personal health data has become, in some way, no longer private[1].

1.1 Problem Description and Motivation

The privacy concern obligates the government agencies to rethink their regulations, especially in this scenario, where the patients become increasingly proactive in their healthcare and private companies are interested in making these data profitable.

Personal data is defined as any information relating to an identifiable person. It can include names, identification numbers, location data, IP addresses, cookies, and any other information through which an individual can be identified, even indirectly. There is no doubt that personal health data is capable of identifying an individual. Moreover, even when health information is intentionally stripped of personal identifiers, it can often be re-identified with low effort using data science techniques[15].

By collecting data from digitally conscious citizens, it is possible to understand the dynamics of the entire population. People with resemblant profiles can be easily grouped, making it possible to apply interventions to those groups. Thus, the produced real-world data is sufficient for life sciences companies to understand the context of real-world data as evidence, and that evidence can be used as insight, for good or bad purposes. In a more pleasant scenario, these insights can help researchers make scientific discoveries and produce successful health treatments. On the other hand, in an undesired situation, these insights can be explored for advertising, unethical health-insurance valuation, and other unknown purposes.

Sequencing thousands of human genes produce a lot of data, commonly known as Omics data (see Section 2.1). These data, collected by next-generation sequencing tools and processed by

[1] https://www.idc.com/getdoc.jsp?containerId=US45415720.

Bioinformatics Pipelines[2](see Section 2.2), have enabled discoveries in many fields, among them precision medicine, the study of rare diseases, and even the development of drugs and vaccines. However, mass collection of such sensitive data introduces enormous legal and ethical risks if not protected to the highest standards and following the highest principles, such as European General Data Protection Regulation (GDPR) regulations (see Section 2.3) and FAIR principles (see Section 2.4).

Therefore, it is imperative to rethink how to make health research non-disclosive analysis (see Section 2.7), which means, that preserve privacy, including analyses that involve Omics data.

1.2 Objectives

The purpose of this dissertation is to study the existing proposals for deal with privacy-preserving analysis, examine their benefits and shortcomings, to suggest a novel architecture that can effectively and efficiently perform privacy-preserving analysis for Omics data.

In addition, the proposed solution suggest a data model that can enhance the reproducibility of the Bioinformatics Pipelines that generated the Omics data used for analysis.

1.3 Structure

The remaining chapters of this dissertation are organized as follows. In Chapter 2 we discuss relevant information for the comprehension of this dissertation, such as: what is Omics data and how these data are generated and manipulated by bioinformatics pipelines; how the European General Data Protection Regulation deal with privacy concerns; a brief description of FAIR and TRUST Principles and how it can be useful to data repositories; what are the characteristics of federated data repositories and, finally, what is non-disclosive analysis and why this technique can be useful to deal with privacy-preserving requirements.

Chapter 3 exposes a survey of the most significant proposals for privacy-preserving analyses and reproducible bioinformatics pipeline tools, with an analysis of their benefits and flaws.

In Chapter 4, we present the design proposals of a privacy-preserving analyses for Omics data, including also a relational model to deal with reproducibility. Proceeding to Chapter 5, we present the actual implementation of our proposed solution.

Finally, we conclude this dissertation with Chapter 6, by summarizing the goals that where accomplished and propose the remaining ones as future work.

[2] A Bioinformatics pipeline is composed of a wide array of software algorithms to process raw sequencing data and generate a list of annotated sequence variants[14].

Chapter 2

Background

2.1 Omics Data

The term "Omics" refers to some areas of study in biology, all of which end in the suffix -omics, such as genomics (to study genomes), transcriptomics (to study transcribed RNAs), proteomics (to study proteins), metabolomics (to study metabolites), etc. Omics are innovative, far-reaching approaches for the analysis of humans and other organisms in the point of view of genetic or molecular profiles.

Genomics is the field of science that studies the complete genome[1] of organisms and their inter-relationships, while genetics focuses on single genes. Genomics helps to determine the entire sequence of nucleotides[2] present in a DNA, analyzing and comparing them with other organisms as a way to understand their functioning and regulation. Though revolutionary and of great importance, genomics did not answer all the scientists' questions. The main curiosity is how an organism responds to different environmental conditions. DNA analysis cannot answer this because it will always be the same response for a particular individual, regardless of the environment. Therefore, to help answering these questions, other "omics" sciences were soon born, as we can see on Figure 2.1, presented by VERDI[3], such as transcriptomics, proteomics, metabolomics, lipidomics, etc.

This holistic vision can allow the researchers to understand "omics systems'" problems, i.e., how complex interactions between genes, molecules exposed to different environmental conditions, can influence the phenotype[4]. The results from a phenotype organism comes from two main factors: the genetic code expression (or gene expression[5]) of an organism or its genotype[6] and

[1] Genome is all genetic material of an organism. Available at https://en.wikipedia.org/wiki/Genome

[2] Nucleotides are organic molecules consisting of a nucleoside and a phosphate. Available at https://en.wikipedia.org/wiki/Nucleotide

[3] https://edisciplinas.usp.br.

[4] A phenotype can be defined as a set of observable characteristics or traits of an organism. Available at https://www.oxfordlearnersdictionaries.com.

[5] Gene expression is the process the cell uses to produce the molecule it needs by reading the genetic code written in the DNA. Available at https://www.genome.gov/genetics-glossary/Gene-Expression.

[6] The genotype of an organism is its complete set of genetic material. The term is often used to refer to a single gene or set of genes, such as the genotype for eye color. Available at https://en.wikipedia.org/wiki/Genotype.

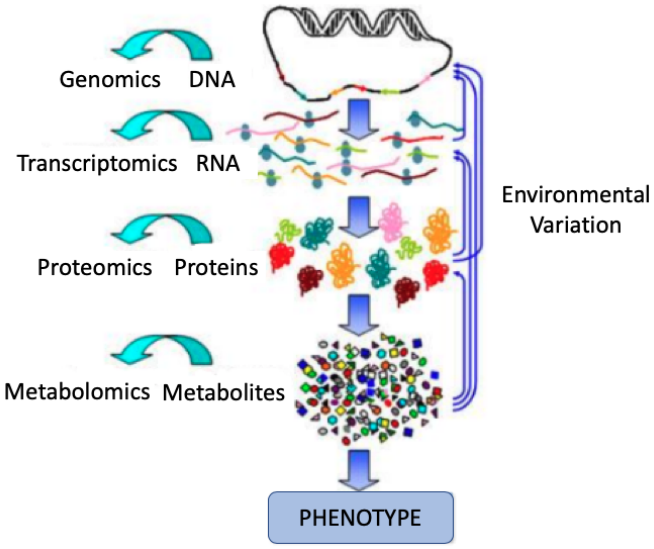

Figure 2.1: Omics technologies. Adapted from VERDI et al., 2019.

the influence of environmental conditions. Understanding these factors helps to understand the disease symptoms of a patient.

Currently, there is no single approach for processing, analyzing and interpreting all kinds of Omics data. Each organism poses different challenges, facing the uniqueness of metabolite abundance, gene expression bias, epigenetic regulation and cell-type specificity of a given Omics dataset.

The development of high-throughput techniques, such as Next Generation Sequencing (NGS)[7], allows the sequencing of entire genomes. The analysis of this kind of data is revolutionizing scientific research and holds great potential for improving treatments. Hence, the importance of these technologies is fundamental to expand and create new perspectives for advanced diagnoses, such as how to understand the effect of variations with the human genome on response to drugs.

There is a real necessity of the health research community to access these data (or, ideally, a summarized, non-disclosive version of them, see Section 2.7), preferentially from repositories that follow the FAIR (see Section 2.4) and TRUST principles (see Section 2.5). Sharing these data across researchers may hugely increase sample sizes, strengthen statistical conclusions and unquestionably relevant to search and discover for the patterns that enable personalized treatments.

Nevertheless, sharing these large datasets between researchers to improve analysis results can be challenging, especially when there is a need to address data privacy concerns, whose theme is covered by regulations like the GDPR (see Section 2.3). The adoption of a federated architecture for data repositories helps to address some of these aspects (see Section 2.6).

Additionally, these data are usually unstructured, with millions of registers. Several tools and commands pre-process them before ready to make an analysis. These processing steps are

[7]Next Generation Sequencing is a term used to describe a number of different modern sequencing technologies. These technologies allow for sequencing of DNA and RNA. Available at https://www.ebi.ac.uk/.

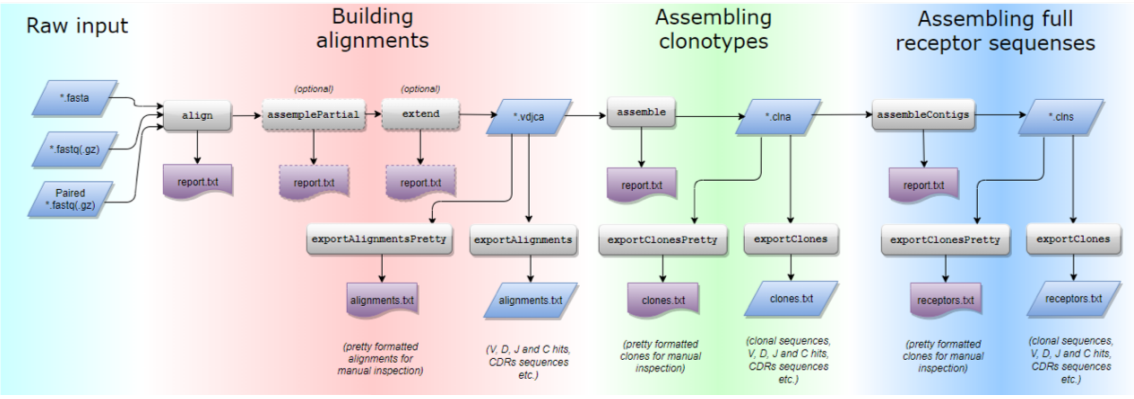

Figure 2.2: AIRR-Seq Pipeline example. Image by MiLaboratory.com, 2018.

executed by Bioinformatics Pipelines (see Section 2.2). Understand how these data were generated and processed is fundamental to validate the reliability of the analysis, generating trust in scientific results. Reproducibility, therefore, is an essential aspect to consider for sustainable Omics data analysis.

2.2 Bioinformatics Pipelines

Researchers and Bioinformatics use algorithms, statistical methods, and tools to extract, organize, and analyze large and complex Omics data. When executed in a set of predefined steps, these elements, combined, are usually referred to as a Bioinformatics Pipeline.

Thus, Bioinformatics Pipelines, sometimes referred to as Genomics workflows, is a mash-up of many different tools and scripts to process raw sequencing data (see an example in Figure 3.4) and generate a list of annotated sequence variants[8].

A Pipeline, in the context of Bioinformatics, receives a set of parameters. These parameters, used through each processing step, can include the software tools, algorithms, quality thresholds, the Germline[9] reference database, the Primers[10], data processing protocols, etc.

A Pipeline can build alignments[11], assemble clonotypes[12], apply several error-correction algorithms to eliminate artificial diversity arisen from sequencing errors, assemble complete receptor sequences, etc. You can see a visual representation of a simplified pipeline in Figure 2.2.

[8] Sequence variants is a surrogate term covering any unintentional amino acid substitutions, omissions, or insertions during protein biosynthesis. Available at https://pubs.acs.org/doi/abs/10.1021/bk-2015-1201.ch002

[9] The Germline is the population of a multicellular organism's cells that pass on their genetic material to the progeny (offspring). In other words, they are the cells that form the egg, sperm and the fertilised egg. Available at https://en.wikipedia.org/wiki/Germline

[10] A Primer is a short single-stranded nucleic acid used by all living organisms in the initiation of DNA synthesis. Available at https://en.wikipedia.org/wiki/Primer_(molecular_biology)

[11] In Bioinformatics, a sequence alignment is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. Available at https://en.wikipedia.org/wiki/Sequence_alignment

[12] Clonotype is a unique nucleotide sequence that arises during the gene rearrangement process for that receptor. The combination of nucleotide sequences for the surface expressed receptor pair would define the T cell clonotype[19].

Table 2.1: Comparison of the Pipeline Annotation of *Leishmania infantum* Genome Executed Across Different Platforms. From Di Tommaso[3].

Platform	**Amazon Linux**	**Debian Linux**	**Mac OSX**
Number of chromosomes	36	36	36
Overal length (bp)	32.032.223	32.032.223	32.032.223
Number of genes	7.781	7.783	7.771
Gene density	236,64	236,64	236,32
Number of coding genes	7.580	7.580	7.570
Average coding length (bp)	1.764	1.764	1.762
Number of genes with multiple CDS	113	113	111
Number of genes with known function	4.147	4.147	4.142
Number of t-RNAs	88	90	88

The researchers use the final outputs to make analyses, but these outputs are sensitive to the parameters used in the Pipeline. For example, different tool versions can produce different results. For instance, if the version of a Germline reference database differs, two pipelines can produce different results.

A Germline reference database is being updated over time[13], as new research develops and identifies new alleles. (e.g., several news alleles[14] were aggregated last year in the international ImMunoGeneTics information system (IMGT) databases)[15]. Additionally, the Germline reference database has population characteristics.

Pipelines are usually executed in a complex parallelization, spawning hundreds of jobs over a distributed cluster. In this way, a typical Bioinformatics Pipeline exhibits complex dependency trees and configuration, resulting in a very fragile ecosystem.

To emphasize this fragility, the same pipeline deployed in different environments can produce different results. To complicate even more, some commands and tools use non-deterministic algorithms[16]. For example, in Table 2.1, presented in this article[3], we can see the results of a pipeline executed in different platforms, where it is possible to identify different small results (underlined) in some attributes.

Since different platforms show different results, isolating the pipeline tools using a platform independence strategy, such as container technologies[17], is highly recommended.

2.3 General Data Protection Regulation

The digital transformation changes the paradigm of technology use in many areas, such as medicine and healthcare. This phenomenon collaborates to every type of data is generated, col-

[13] https://www.antibodysociety.org/the-airr-community/airr-working-groups/germline_database/.

[14] An allele is one of two, or more, forms of a given gene variant. Available at https://en.wikipedia.org/wiki/Allele.

[15] http://www.imgt.org/.

[16] A non-deterministic algorithm is an algorithm that, even for the same input, can exhibit different behaviors on different runs. Available at https://en.wikipedia.org/wiki/Nondeterministic_algorithm.

[17] https://containerjournal.com.

lected, and replicated in several places. Therefore, controlling the data, i.e., defining permissions and access rules and regulations to protect citizen privacy, is challenging. On the other hand, controlling the data controllers instead of controlling the data is not only feasible, but also vital. Recent consumer custody laws, like GDPR, address this issue.

GDPR takes a conceptual approach to the permitted and prohibited uses of personal information, individuals' rights of access and control, and companies' obligations to respect the limits and rights of the owners. The biggest concern in GDPR is "what" the data is, not "who" may be holding it.

In the core of its structure, GDPR adopts the notion of primary data custodian, or in other words, a Controller. In general, a Controller is "the natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data". Another relevant stakeholder is the Processor, which is "a natural or legal person, public authority, agency or other body which processes personal data on behalf of the Controller"[18].

Controllers specify the steps of processing personal data, and may or may not be responsible for directly collect the data from data subjects. An example in health area would be: a diagnosis laboratory (Controller) can collect the data of its patients while they do an exam, but there is another institution (Processor) that curates, loads, digitizes, catalogs, and indexes the information produced by the Controller (diagnosis laboratory). These institutions can be data centers or document management holders. Both institutions are responsible for treating the patients' personal data.

A building block for good data protection practice is to follow the guiding principles for the processing of personal data defined by the GDPR. We take a look at each principle below[19]:

- **Lawfulness, fairness, and transparency**: This kind of data must be processed transparently, legally, adequately.
- **Purpose limitation**: Data that can be collected and used only for those purposes that have been clearly defined to the data subject. The consent from the data owner is required.
- **Data minimization**: The purpose for which data is processed will limit the amount of collected data.
- **Accuracy**: Data must be kept up to date, i.e. old or outdated data and contacts must be erased.
- **Storage limitation**: Data which have public interest, for historical and scientific research may be archived. Otherwise, should not be stored for longer than is necessary.
- **Integrity and confidentiality (i.e. data security)**: Data must be curated with appropriate security requirements, to avoid accidental loss or unlawful processing. To protect the identity of the clients, some techniques, like pseudonymization, must be applied.

[18] https://edpb.europa.eu/.
[19] https://crsreports.congress.gov/product/pdf/R/R45631.

- **Accountability**: The GDPR's principles must be followed by Controllers.

When we look at the Controller and Processor stakeholders, the technology constraints and Regulation rules find an intersection. In this intersection, emerges another set of principles: the FAIR principles (see Section 2.4).

2.4 FAIR Principles

The reuse of research data needs improvement, including not only better infrastructure support, but a set of techniques, tools, practices, and last, but not least, principles.

The principles precede implementation, working as guidelines: they affect how the tools will be produced, how the relationship between different stakeholders will proceed, how the data governance must be accomplished, and which types of guarantees must be feasible and agreed with data owners.

The FAIR Principles improve data management and stewardship, defining guidelines that help Controllers and Processors.

The FAIR Guiding Principles can be summarized as follows[16]:

- **Findable**: Humans and computers must be capable to find the data and metadata. Metadata must include tags and identifiers of the data it describes. Data and metadata must be indexed.
- **Accessible**: Data and metadata are retrievable using tags or identifiers, using open and universally implementable communications protocols, possibly including authentication and authorization.
- **Interoperable**: The data usually needs to be interoperable with other systems. Its knowledge representation must use broadly applicable language.
- **Reusable**: To optimise the reuse, data and metadata should be richly described, versioned and associated with detailed provenance, following community standards.

The FAIR principles help to describe a path in the direction of "machine-actionability", i.e., the ability of software, or bots, to find, access, interoperate, and reuse data without manual intervention. This process enables the agent to:

1. identify the object's structure, including its type;
2. analyze the semantic context of the task interested in that object to determine if it is useful;
3. determine if it is usable by checking the license, consent, accessibility or other use constraint;
4. take the expected action, like a human would.

The semantic relationships of the data, its metadata, curation, and harmonization are useful aspects to assist algorithms in their discovery and data inquiry through interoperability. These characteristics are attributes of a good data management and stewardship.

There is a real interest of governing bodies and funding institutions in following the FAIR Principles for open data publishing, as we can see within the Horizon 2020, a Work Programme 2018-2020 from European Commission, call topic SC1-BHC-05-2018: "International flagship collaboration with Canada for human data storage, integration and sharing to enable personalised medicine approaches"[20]. In parallel, many portals are emerging, where it is possible to find useful information and data repositories that are FAIR compliant[21].

The **FAIRsFAIR**[22] initiative facilitates sharing of knowledge, guidelines, courses, education and expertise needed to turn the FAIR principles into reality. FAIRsFAIR is producing examples to support the use of the FAIR data principles, generating recommendations for the European Open Science Cloud (EOSC)[23], an intergovernmental cloud service to support EU science.

The **GO FAIR**[24] is a bottom-up, self-governed initiative and focused on culture changing, training, technology standards and infrastructure components. GO FAIR developed and disseminated the FAIRification process[25], a schema to transform non-FAIR data repositories into FAIR repositories. It is focused on data, and indicates the required changes for metadata. The FAIRification process is particularly useful for organizations that have a lot of useful datasets that are not ready to be shared in a secure and federated way.

The **FAIRsharing.org**[26] is a FAIR resource hosted at the University of Oxford since 2011, focused on producing standards, policies and educational content about how to curate data and metadata following the FAIR principles. In FAIRSharing it is possible to find FAIR maturity indicators, metrics, models, guidance to stakeholders, training material, and so on.

Many known repositories are already implementing the FAIR principles, such as Dataverse[27], FAIRDOM[28] and wwPDB[29].

2.5 TRUST Principles

The emergence of FAIR implies a related question: "Who can we trust to enable FAIR?". Transparency (T), responsibility (R), user community (U), sustainability (S), and technology (T), define the TRUST principles[8], and help to keep FAIR data over time. TRUST provides a way to think about data life-cycle management and preservation within repositories aligned with FAIR

[20] https://ec.europa.eu/info/funding-tenders/.
[21] https://ec.europa.eu/research/participants/.
[22] https://www.fairsfair.eu/.
[23] https://ec.europa.eu/info/research-and-innovation/.
[24] https://www.go-fair.org/.
[25] https://www.go-fair.org/fair-principles/fairification-process/.
[26] https://fairsharing.org/.
[27] https://dataverse.org/.
[28] https://fair-dom.org/.
[29] https://www.wwpdb.org/.

principles. In other words, a successful application of the TRUST principles ensure the alignment of data management standard practices for managing and preserving FAIR data in repositories.

It is not a good strategy to delegate to individuals the responsibility of managing data repositories. Frequently, individuals do not have career stability, e.g. they may change careers or move departments/institutions, or necessary technological resources to ensure trustworthiness. FAIR data does not guarantee that data will be preserved, remain available, and become usable for independent verification of results at any time in the future. Operationalizing FAIR to preserve FAIR data objects requires digital repositories to be trustworthy for the longer term.

The fundamental difference between FAIR and TRUST is that FAIR defines the desirable characteristics of digital objects (data and associated metadata) at a certain time point, while TRUST describes the characteristics of repositories that are needed for responsibly managing and disseminating digital objects and maintaining their FAIR alignment over a long period.

TRUST works like guidelines to ensure that repositories have transparent policies, organizational capabilities, and guardians behind the websites, infrastructures and databases, who understand what enabling FAIR data entails on a practical level. We can define every characteristic of the TRUST principles as follows:

- **Transparency** is achieved by providing publicly accessible evidence of the services that a repository does and does not offer.
- **Responsibility** is a commitment to provide reliable data services together with clarity as to where responsibility for those services resides.
- **User focus** is a commitment to be clear which community is being served, and implement and enforce the standards and norms of the user community.
- **Sustainability** is a measure to identify if a data system is ready to support long-term preservation and use.
- **Technology** is the infrastructure and capability of supporting repository operations.

The adoption of TRUST Principles offers the following advantages:

- The opportunity to bring different certification standards, such as CoreTrustSeal[30], ISO 16363[31], DIN 31644[32] and others.
- An easily understandable manner to apply conceptual frameworks, like OAIS ("Open Archival Information System", ISO 14721[33]), to generalize trustworthiness beyond disciplinary data repositories.

[30] https://www.coretrustseal.org/.
[31] https://www.iso.org/standard/56510.html.
[32] https://www.beuth.de/en/standard/din-31644/147058907.
[33] https://www.iso.org/standard/57284.html.

- Guidance of repositories' operation, in concert with other principles, such as the FAIR principles.
- The opportunity to communicate and describe how they are working toward each component of the TRUST principles and share practices.

Quality data are the fuel that powers the research enterprise. The TRUST principles can help support the infrastructure required to manage these valuable resources for research communities.

2.6 Federated Data Repositories

A federated data repository is part of a federated system. A federated system is composed of a collective agreement upon many standards, at the same time admits independence and autonomy of each component of the federation. Heterogeneity in terms of hardware, operating systems and network are also characteristics of a federated system. Most importantly, the analysis is executed distributively, accessing the data distributed over multiple databases.

Federated data repositories are necessary in order to facilitate cross-border collaborations between research centers. In a typical federated data repository, each data holder is responsible for maintaining the control over their data, including curation and security aspects while sharing samples of this data to other researchers or data holders. Researchers can produce queries using a central portal that then searches to all participants of the federated system. Records are matched to fulfill the requests.

The characteristics of the federated data repositories can be briefly listed as follows:

- **Data ownership**: Each institution is the owner of their data and responsible for your own staff.
- **Technical requirements**: All the required hardware and network bandwidth must have been provided for each data holder.
- **System Performance**: The data delivery is influenced by delays during the loading of each source systems.
- **Privacy/Security**: Each source data system provider is responsible for your security process.
- **Data updates/corrections**: Data resides within each data provider. Each data provider is responsible for communicating all corrections, updates or data processes changes.
- **Data availability**: Access to data is determined by the source data provider.
- **Data quality**: Dependent on processes implemented at each data provider.
- **Implementation**: Usually requires less time than a central approach, but the data provision can be more complex.

- **Scalability**: Each source data system is responsible to include any additional required hardware or other resources.
- **Production of standard reports**: Dependent on a data provider's responsibility.
- **Sustainability**: Partners of the system are responsible. Engagement is necessary, but a funding formula can help to increase the uniformity of resources.
- **Usability**: Partners will query a large amount of data, which requires assurance of comparability.

The federated approach for data repositories mitigates the turf battles/gets around trust issues. Moreover, tailored protection of data is based more on the sensitivity of each data holder. The federated approach, combined with non-disclosive analysis mechanisms (see Section 2.7), can also minimize the necessity of Data Transfer Agreements (DTA)[34].

2.7 Non-disclosive Analysis

The non-disclosive analysis is a technique that provides information that respects all of the legal, contractual, or ethical undertakings about the usage of the data. The data owner has agreed with third parties, and no individual data should be inferred. It makes use of data with those agreements under which sensitive or personal information can be shared.

The purpose of the non-disclosive analysis is to protect the privacy of the participants and subjects of the research, respecting the legal restrictions imposed by governance rules (see Section 2.3).

Researchers are usually interested in statistics and trends among larger groups of individuals or samples instead of single individuals. The non-disclosive analysis uses statistical disclosure control (SDC), also known as statistical disclosure limitation (SDL) or disclosure avoidance.

The non-disclosive analysis is feasible in several research situations. In essence, a central analysis computer may coordinate a parallelized simultaneous analysis of the individual-level data on a spread, federated set of data repositories (see Section 2.6). These request each server to undertake a particular analysis and return non-disclosive summary statistics to the analysis computer, that is, data that cannot possibly lead to the identification of the individuals to which they relate.

For instance, to know an average value from a specific variable, the central analysis computer can send a command block for all data repository participants asking for the average in your dataset. Then, each participant response with the average value without exposing the individual value of the requested variable.

In this way, it is possible to fit a mathematical model as if the individual-level data from all studies were pooled centrally on the analysis computer while, in reality, the data never leave their studies of the origin, and all that does leave are the non-disclosive summary statistics.

[34] A DTA is a contract between the providing and recipient institutions that governs the legal obligations and restrictions, as well as compliance with applicable laws and regulations, related to the transfer of such data between the parties. Available at http://www.ott.emory.edu.

2.8 Summary

This chapter presented vital concepts to provide the necessary background to understand the related work and the proposed solution. Most of these concepts are not directly related to Software Engineer, corroborating the idea that the presented work involves a diversity of topics, such as Legal, Health, and Software.

Chapter 3

Related Work

In this chapter, we will take a look at some of the most prominent initiatives for privacy-preserving analyses across federated health data repositories. Whenever possible, the current limitations of each one will be cited, especially when involving Omics data. We will also present an overview of Reproducible Bioinformatics Pipeline tools, which aim to provide a general schema and an infrastructure to distribute reproducible workflows.

3.1 Privacy-preserving Analyses solutions

In this section, we will present four known solutions: iRecetor+ platform, OBiBa, MOLGENIS, and GA4GH solutions. The selection criteria were: the direct involvement of INESC TEC, and the relevance of related scientific papers.

3.1.1 The iReceptor+ platform

The iReceptor[1] platform, developed at Simon Fraser University, Canada, is a distributed data management system and a scientific gateway for mining sequence data (a large collection of computerized nucleic acid sequences, protein sequences, or other polymer sequences) from immune responses.

iReceptor smooths curating, analyzing and sharing of antibody/B-cell and T-cell receptor repertoires (Adaptive Immune Receptor Repertoire or AIRR-sequencing data). AIRR-sequencing has enormous potential for understanding the dynamics of the immune repertoire[2] in vaccinology, infectious disease, autoimmunity, and cancer biology, but also poses substantial challenges[2].

The iReceptor+[3] is a project, funded by the European Union's H2020 Research and Innovation Programme and Canadian Institutes of Health Research (CIHR), will expand the analysis tools

[1] http://ireceptor.irmacs.sfu.ca/.

[2] Immune repertoire is defined as the sum of T cell receptors and B cell receptors (also named immunoglobulin) that makes the organism's adaptive immune system. Available at https://en.wikipedia.org/wiki/Immune_repertoire.

[3] https://www.ireceptor-plus.com/.

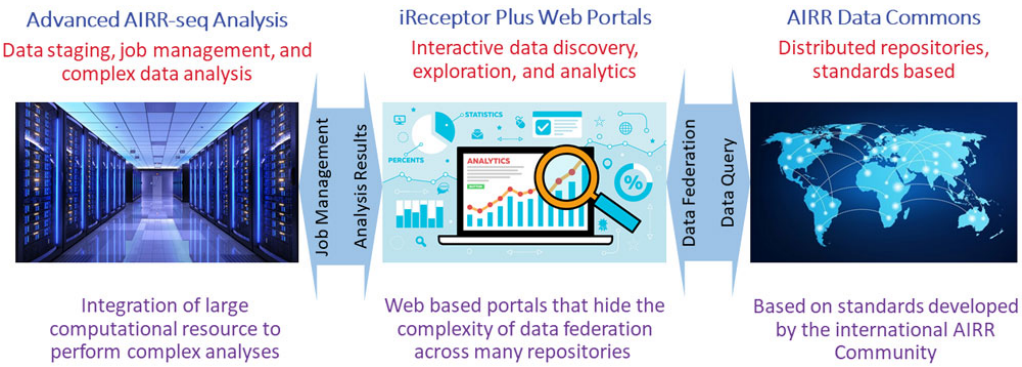

Figure 3.1: The High-Level iReceptor Plus Platform Architecture. From ireceptor-plus.com.

available on the iReceptor Gateway, including for single cell or systems immunology approaches, and add security for iReceptor+ repositories. Figure 3.1 is a high-level iReceptor+ platform architecture:

The iReceptor+ is designed as a network of federated repositories, following standards for sharing and interoperability developed by the AIRR Community, known as AIRR Data Commons.

The AIRR Data Commons aims to facilitate data queries and advanced analysis across multiple projects, labs, institutions, and countries, through web portals (e.g., the iReceptor+ Scientific Gateway).

The data privacy and ownership are concerns addressed by the distributed federated architecture that was adopted in this project. It allows each institution to maintain control over its data and stay compliant with local legislation. There are plans for iReceptor+ to integrate with relevant non AIRR-seq clinical and biological. It is useful because allows the analysis of global interactions within the immune system and within its environment.

The AIRR Common Repository Working Group (CRWG) has developed a set of recommendations that promote the storage, sharing, and use of AIRR sequence data. In May 2020, was released the first version of the AIRR Data Commons API (ADC API). AIRR Data Commons can be viewed as a set of repositories that adhere to the CRWG recommendations, that implement the ADC API as a mechanism to access that data.

The iReceptor+ Web Portals work like a scientific gateway that integrates these large, distributed, AIRR-seq data repositories. Present functionalities include:

- Search for repertoires satisfying certain metadata (e.g. find all AIRR-seq repertoires from ovarian cancer studies);
- Search for all repertoires that contain specific complementarity-determining regions (CDRs) sequences;
- Search identified repertoires for sequences derived from particular V, D, and J genes and alleles (alternative forms, created by mutation, from the same gene);
- Download sequences from these repertoires in AIRR.tsv format that can be imported by other AIRR-seq analysis tools.

Some repositories are currently adhering with AIRR Data Commons (ADC), such as the iReceptor Turnkey, Immune DB, and sciReptor.

The iReceptor Turnkey[4] is a mechanism for researchers to create their own AIRR Data Commons repository. The iReceptor Turnkey Repository consists of a database software stack (based on MongoDB), a API that allows external users to query that repository through the API (ADC adherent), and a set of services that help users curate data in the repository. By using the iReceptor Turnkey Repository, a research lab will have access to both a local repository for their data as well as the ability to share that data by integrating their repository node into the AIRR Data Commons. Such integration would allows the iReceptor Scientific Gateway to perform queries across all of the data in the AIRR Data Commons, including the data in their own repository.

ImmuneDB[5] is a tool to analyze and store B-cell receptor (BCR) and T-cell receptor (TCR) data. In its more common use, ImmuneDB excels at acting as a central data storage and as interface between other tools such as IgBlast[6], MiXCR[7], and VDJtools[8] via AIRR compliant importing and exporting routines. ImmuneDB uses an optimized MySQL database to store raw sequencing data as input. The list of possible analyses the researcher can do include:

- annotate receptor gene usage;
- calculate selection pressure;
- infer clonotypes;
- aggregate results;
- construct clonal lineages[9].

Pre-annotated data can be imported as well, and analysis outcomes can be exported in several file formats.

sciReptor[6] is a tool to process, store and analyse single-cell level immunoglobulin (Ig) and TCR sequence data. With sciReptor it is possible to analyse and compare Ig sequencing data originating from various experimental protocols. sciReptor has a relational database, which stores all sequences, annotations and metadata in a standardized format. sciReptor support sequences or genomic annotations of reference sequences. The parameters and reference sequences are customizable for individual projects. sciReptor can analyze data from a previously published single-cell matrix PCR platform, and from data generated by alternative experimental procedures.

The concept of iReceptor+ Web Portal finds similarity with TCRdb Web Portal. TCRdb[10] is a comprehensive human TCR sequences database. TCRdb categorizes sample metadata, enables

[4] https://github.com/sfu-ireceptor/turnkey-service-php.
[5] https://immunedb.readthedocs.io/en/latest/.
[6] https://www.ncbi.nlm.nih.gov/igblast/.
[7] https://mixcr.readthedocs.io/en/master/.
[8] https://vdjtools-doc.readthedocs.io/en/master/.
[9] Clonal lineage represents a set of B cells that are related by descent, arising from the same VDJ rearrangement event. Available at https://ionreporter.thermofisher.com.
[10] http://bioinfo.life.hust.edu.cn/TCRdb.

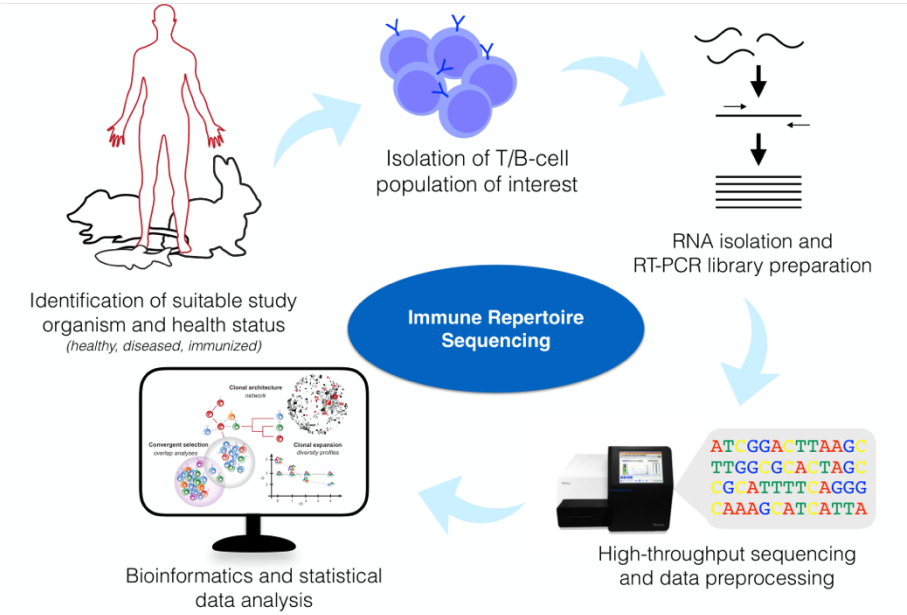

Figure 3.2: Immune Repertoire Sequencing. By Victor Greiff (greifflab.org).

comparison of TCRs in different sample types, shows several data visualization charts, describes the TCR in diversity, length distribution and V-J gene utilization. TCRdb collects and curates TCR-Seq data and TCR sequences from public TCR-Seq datasets from the National Center for Biotechnology Information of the United States (NCBI) Sequence Read Archive (SRA), and from other TCR sequences databases, including iReceptor, VDJServer[11] and immuneACCESS[12].

3.1.1.1 AIRR-seq: Adaptive immune receptor repertoire sequencing

AIRR-seq has enormous promise for understanding the dynamics of the immune repertoire in vaccinology, infectious diseases, autoimmunity, and cancer biology, but also poses substantial challenges. The AIRR sequencing is described on Figure 3.2.

The first step is the isolation of the T/B-cell population of interest, followed by the RNA isolation. After that, raw sequence data of B-cell/T-cell are extracted using sequencing instruments like Illumina Genome Analyzer or Illumina NextSeq 550 Series[13], as we can see on Figure 3.3.

Once these data are extracted, they are stored in files with .FASTA or .FASTQ extensions (see Figure 3.4), that are text files that contain the sequence data. These files, at this point, are known as "raw data" or "raw sequences", and can contain up to millions of entries and can have several megabytes or even gigabytes in size, which often makes them too large to open in a normal text editor. Viewing these files is not necessary as they are intermediate output files used as input for

[11] https://vdjserver.org/.
[12] https://clients.adaptivebiotech.com/immuneaccess.
[13] https://www.illumina.com/systems/sequencing-platforms.html.

Figure 3.3: An example of a sequencing instrument (Illumina NextSeq 550 Series).

tools that perform downstream analysis, such as alignment to a reference or *de novo* assembly[14]. Examples of these tools are iGBlast and MiXCR.

MiXCR is an universal framework for processing big immunome data[15] from raw sequences to quantitated clonotypes. MiXCR assembles clonotypes by applying several error-correction algorithms. It is necessary to eliminate artificial diversity arising from PCR (polymerase chain reaction) and sequencing errors, handle paired- and single-end reads, consider sequence quality, and identify germline hypermutations. For example, the workflow from T-cell receptor data sets to final clonotypes, following a typical MiXCR pipeline (commonly used in bioinformatics pipelines), is shown on Figure 3.5.

Once clonotypes are assembled, they become ready for exploring targeted and advanced data analysis.

[14] *de novo* assembly is a method for constructing genomes from a large number of DNA fragments, with no *a priori* knowledge of the correct sequence or order of those fragments. Available at https://thesequencingcenter.com/knowledge-base/de-novo-assembly/.

[15] immunome data is a set of genes and proteins that constitute the immune system. Available at https://en.wikipedia.org/wiki/Immunome.

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
```

Figure 3.4: An extracted section of a raw sequencing file that follows the .FASTQ format.

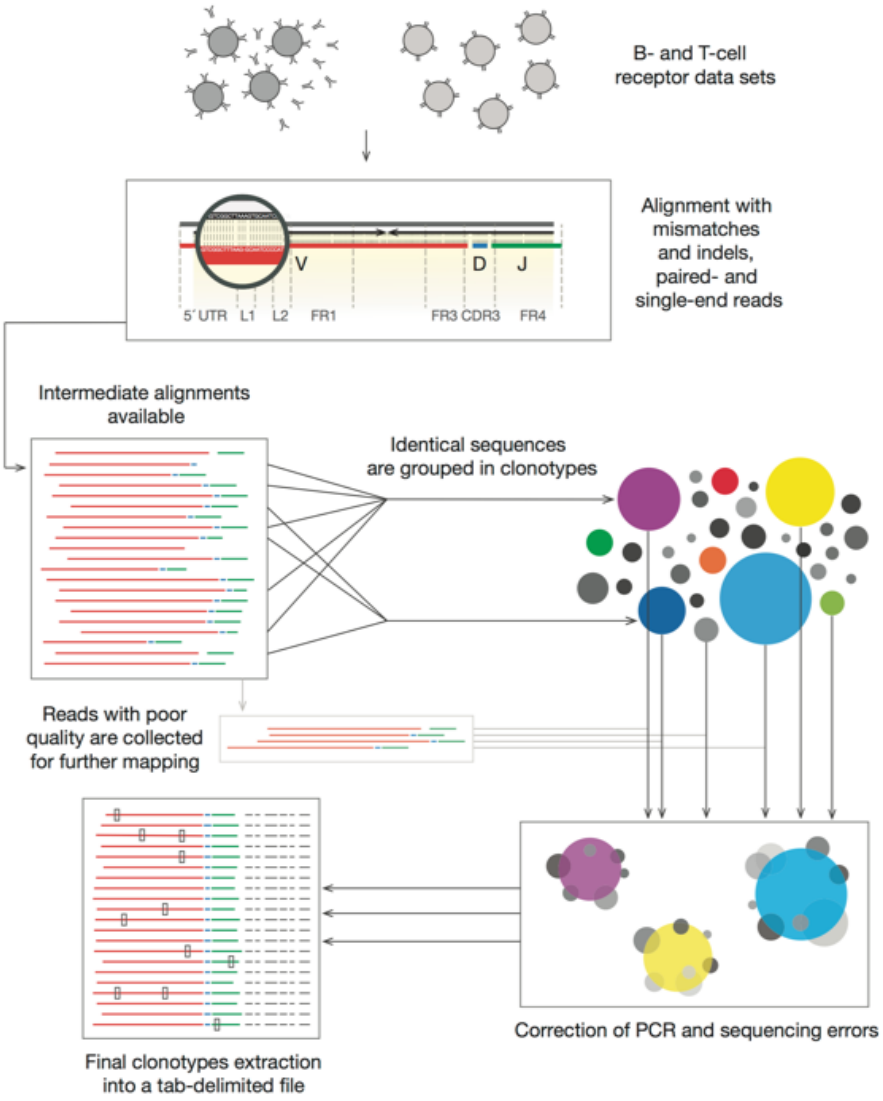

Figure 3.5: A typical MiXCR pipeline. Image by MiLaboratory.com.

3.1.2 The OBiBa solution

OBiBa[16] is a project developed as part of the Maelstrom Research[17] program, in collaboration with Canada, United Kingdom, and the European Union. Its aim is to build and maintain open-source software solutions for epidemiological studies and support the entire data management lifecycle including data collection, integration, harmonization, sharing and analysis.

OBiBa solution offer a comprehensive and integrated software solution for both individual studies and networks of studies, giving the possibility to build a federated infrastructure.

OBiBa solution has extensible interfaces to enable interoperability and secure data transfer from one OBiBa application to another. The OBiBa support almost all data management activities, and can be used for both individual studies and study consortia. Each application of the OBiBa stack is responsible for a domain-specific problem:

- **Mica**: is the web portal for report and summarizes **published** data dictionaries and aggregated results. Mica has advanced search capabilities that allow researchers to explore the study's variables and data profiles. Data access requests can also be submitted and evaluated, making it for other researcher to evaluate the published data and aggregated results.

- **Opal**: is able to **store** data from several data sources, like CSV or SPSS files. These data can be imported into study-specific Opal databases. In Opal it is possible to create views and derived variables to implement processing algorithms that transform data collected by each study into a common (i.e. harmonized) dataset. Opal is extensible and offers data transformations such as curation, data cleaning and quality checks. The studies' data can also be enriched with metadata. In Opal you can compartmentalize the data for better privacy.

- **Agate**: is a web application that offers users related services to the OBiBa software stack: user **central authentication**, user profile management, user notifications. These services are offered to the remaining applications: Opal, and Mica.

OBiBa's solution is promoted by several health research consortium in Europe and Canada, such as RECAP Preterm (Research on European Children and Adults Born Preterm)[18], and EucanConnect (Connecting Europe and Canada in personalized & preventive health care)[19].

3.1.2.1 DataSHIELD

DataSHIELD[20] is a solution developed and maintained by the University of Newcastle, composed by a client and a server R Package. There are planning to develop a Python version of the client and server packages.

[16] https://www.obiba.org/.
[17] https://www.maelstrom-research.org/.
[18] https://recap-preterm.eu/.
[19] https://eucanconnect.com/.
[20] https://www.datashield.ac.uk/.

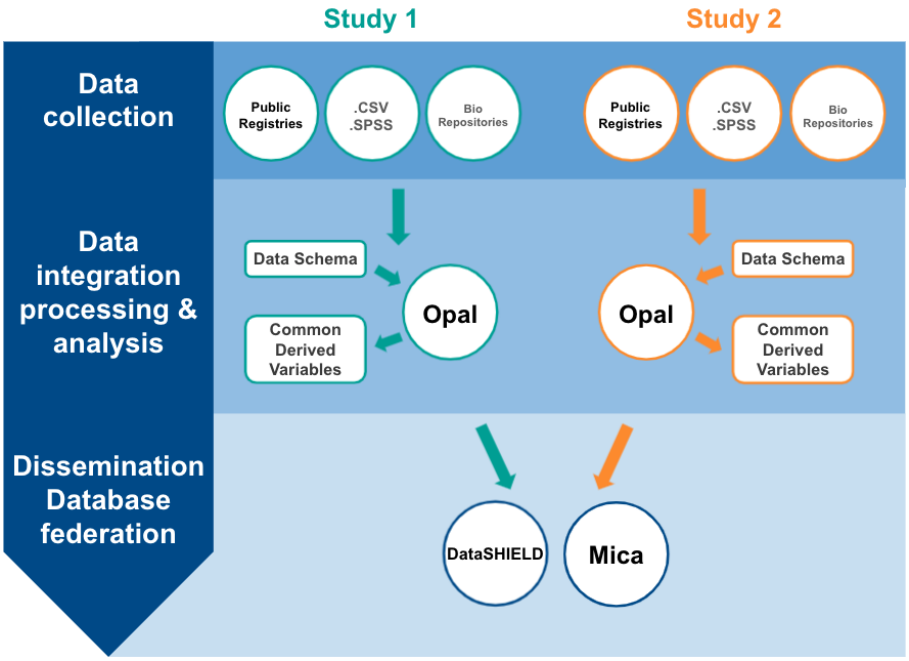

Figure 3.6: From data collection to data federation via data harmonization in a study consortia. Adapted from OBiBa.org.

DataSHIELD is a powerful tool that enables advanced and federated statistical analysis across a network of data, including Opal databases, without interposing ethical and legal constraints when the central pooling of individual-level data is prohibited.

DataSHIELD helps the researchers in situations where:

- It is scientifically necessary to perform co-analysis of individual-level data from several studies, but there are regulations or governance rules that prevent the release, the sharing, or the copying of some of the required data.
- A researcher wishes to share the information held in its data with others researchers actively but would like to maintain the governance control or the intellectual property of these data.
- A dataset that is to be remotely analyzed or included in another study contains data objects that are too large to be physically transferred to the analysis site.

The OBiBa team built the native integration required to run DataSHIELD analyses on data stored in Opal. Such integration, as we can see on Figure 3.6, studies can run advanced analysis such as linear regressions, logistic regressions, in a controlled environment where the fine permission for authorization and authentication can be set. OBiBa, in conjunction with DataSHIELD, can offer a federated software solution to support privacy-preserving analyses.

The data transformed in a common dataset can be queried and analyzed with the DataSHIELD approach through a federated database system without sending individual-level study data outside of host institutions. DataSHIELD allows the data owner to set (and control) a variety of optional privacy levels to deal with disclosure control[17].

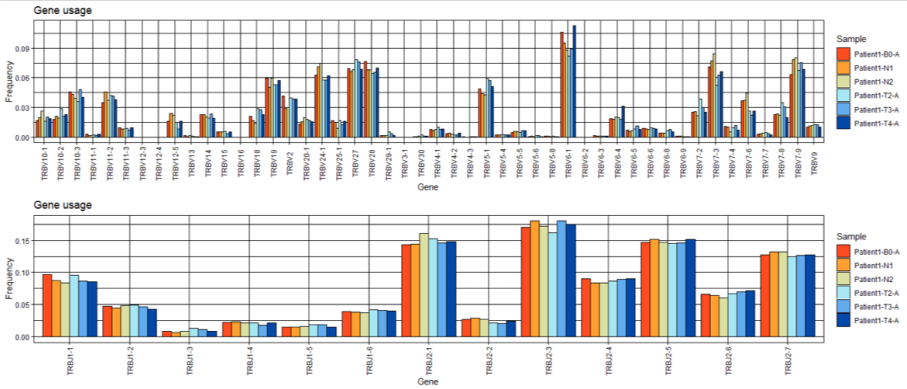

Figure 3.7: VJ Gene Distribution.

The DataSHIELD analyses are executed in R Sessions, which are long-running tasks that contain the current working environment. At the end of an R session, the researcher can save an image of the working environment that can be automatically reloaded the next time R is started.

3.1.2.2 Using "DataSHIELD like" approach for non-disclosive analysis of AIRR-seq data

To test the feasibility of "DataSHIELD like" approach for AIRR-seq data, a setup was deployed using the OBiBa stack, with Opal for data storage and R Server, DataSHIELD and Immunarch[11] for analysis.

Immunarch makes immune sequencing data analysis as effortless as possible and helps users to focus on research instead of coding. Immunarch works with any type of data: single-cell, bulk, data tables, databases, and offers automatic format detection and parsing for several popular immunosequencing formats: MiXCR, ImmunoSEQ[21], 10XGenomics[22], and ArcherDX[23].

In this setup, the dsImmunarch package, deployed on server, runs like a wrapper for Immunarch functions, and clients may query for specific analyses using dsImmnuarchClient, including VJ gene distribution, clone frequencies, repertoire similarity, and extraction of motif features (motifs are short, recurring patterns in DNA that are presumed to have a biological function). This Immunarch customization was not fully adapted to deal with the DataSHIELD disclosure control rules, since it was a proof of concept. An Immunarch version full DataSHIELD compliant requires additional development.

In the VJ Gene distribution, to compute the distributions of genes, immunarch includes the geneUsage()[24] function. It receives a repertoire or a list of repertoires as input and genes for which the user wants to get statistics (see Figure 3.7 to see in gene usage computation format, see Figure 3.8 to see in a TreeMap format, see Figure 3.9 to see in a Spectratype format).

When dealing distributions, data may be plotted in multiple ways. Spectratype is a useful way to represent distributions of genes per sequence length.

[21] https://www.adaptivebiotech.com/products-services/immunoseq/.
[22] https://www.10xgenomics.com/.
[23] https://archerdx.com/.
[24] https://immunarch.com/articles/web_only/v5_gene_usage.html.

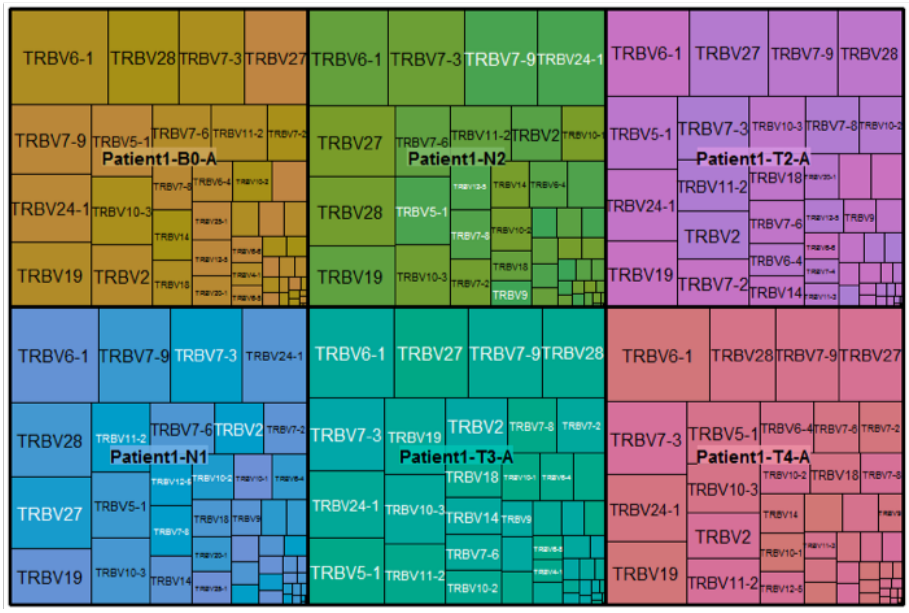

Figure 3.8: V-J Gene Distribution with Treemap visualization.

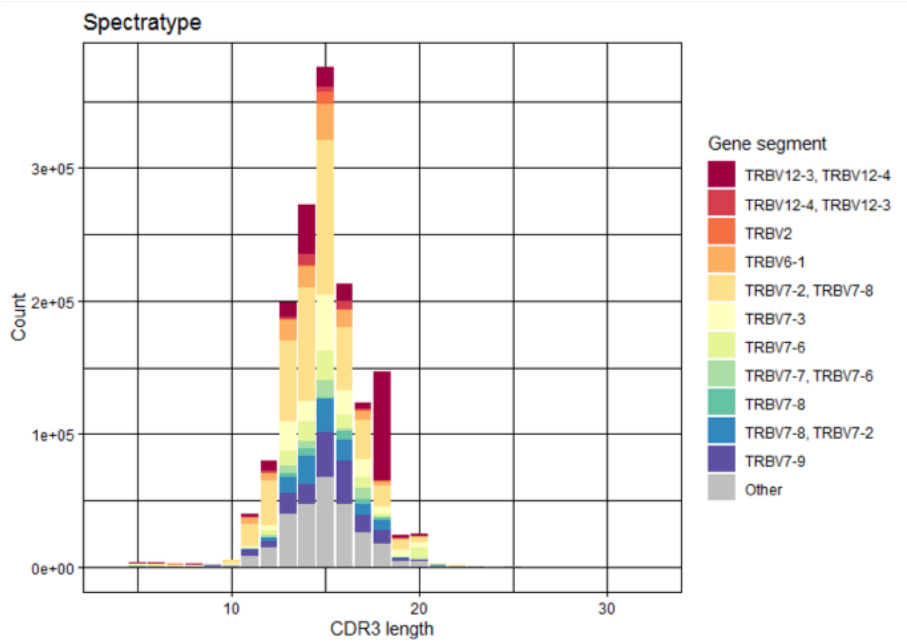

Figure 3.9: V-J Gene Distribution with Spectratyping visualization.

Number of Samples	Number of Clonotypes	Process duration (avg)
1	69,402	0.91 mins
6	321,602	1.33 mins
6 (*)	1,140,494	1.8 mins
12	763,607	3.11 mins
18	1,904,101	5.45 mins

(*) – Different data sequencing project

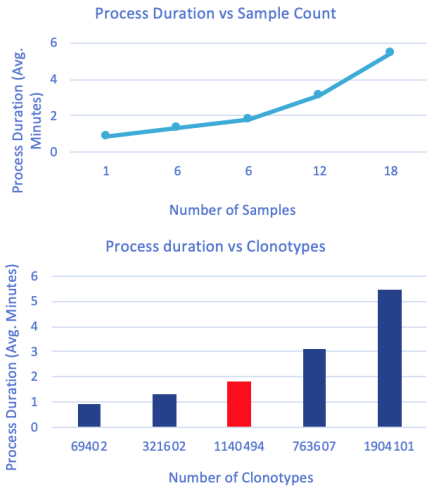

Figure 3.10: Benchmarking and Feasibility of V-J Gene Distribution.

Each experiment was performed at least three times for averaging. It was noticed a non-proportional behavior between clonotypes and time to process (see Figure 3.10).

In the Clone Frequency analysis, for determining clone frequencies, immunarch implements the function repExplore()[25] to extract statistics from repertoires (see Figure 3.11), and repClonality()[26] to assess clonal proportions of repertoires (see Figure 3.12):

In terms of feasibility and benchmarking (see Figure 3.13), the Clone Frequency analysis was slightly more resource-demanding than Gene Usage, and the non-proportional behavior between samples and time to process was noticed as well.

In the Repertoire Similarity analysis, repertoire overlap is a common approach to measure similarity. It was achieved by comparing clonotypes shared between give repertoires, also called "public" clonotypes (see Figure 3.14). It was based on Morisita's index, that is a statistical measure of dispersion of "individuals" in a population.

[25] https://immunarch.com/reference/repExplore.html?q=repexplore.
[26] https://immunarch.com/reference/repClonality.html?q=repclonality.

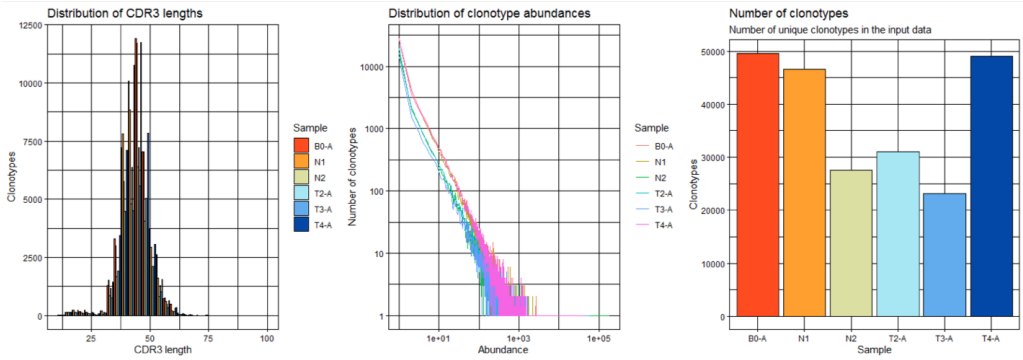

Figure 3.11: Clonotypes in order of CDR3 length (left), Abundance of Clonotypes (middle), Number of Unique clonotypes per Sample (right).

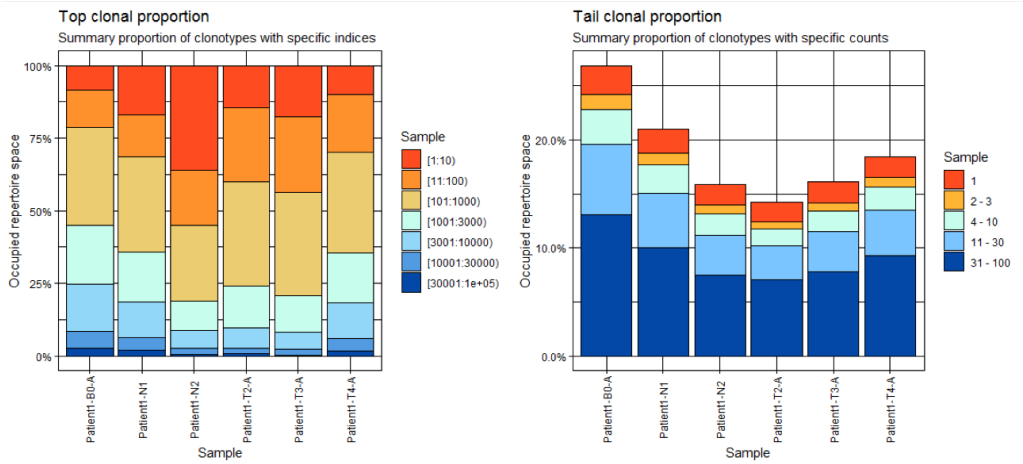

Figure 3.12: Most abundant cell clonotypes (left) vs Least prolific clonotypes (right).

Number of Samples	Number of Clonotypes	Process duration (avg)
1	69,402	0.49 mins
6	321,602	1.52 mins
6 (*)	1,140,494	3.83 mins
12	763,607	3.08 mins
18	1,904,101	7.24 mins

(*) – Different data sequencing project

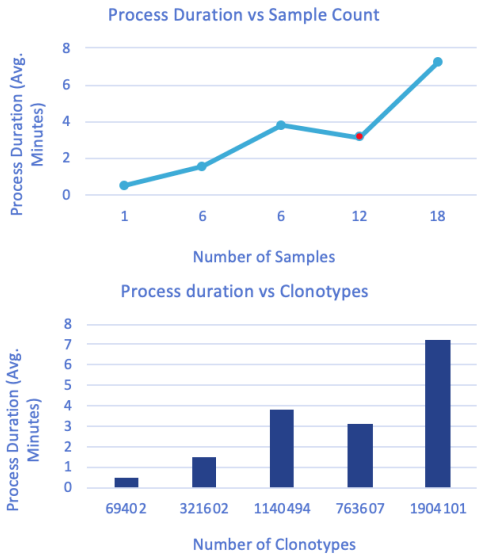

Figure 3.13: Benchmarking and Feasibility of Clone Frequency.

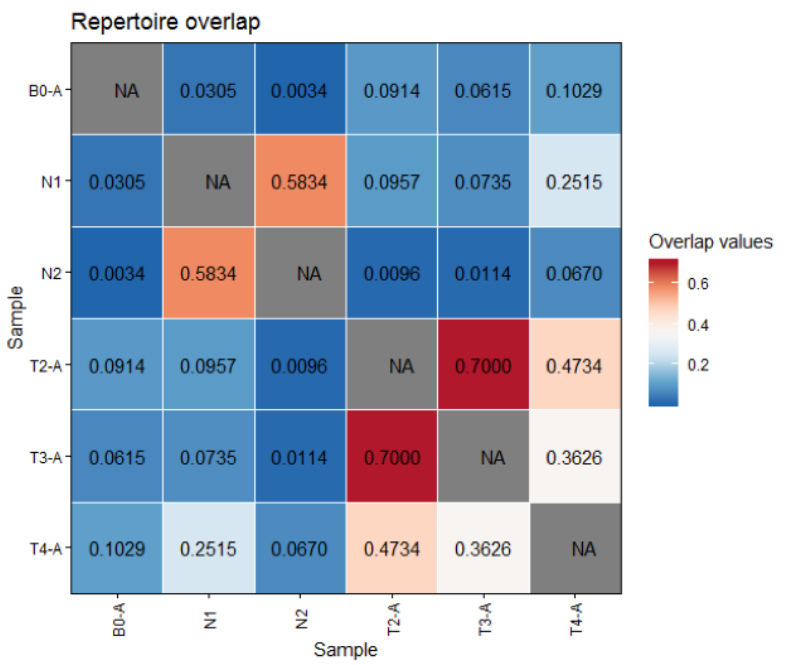

Figure 3.14: Repertoire overlap.

In the Motif extraction analysis, a K-mers based analysis was used to extract motifs from clonotype sequences (see Figure 3.15). It is possible to do this directly using immunarch by using the getKmers()[27] method.

In terms of feasibility and benchmarking (see Figure 3.16), Motif Extraction demanded more resources than Gene Usage and Clone Frequency. Like Clone Frequency, a non-proportional behavior between samples and time to process was noticed.

All the previous examples were based on previously extracted clonotypes using MiXCR, not raw sequences.

There are differences in data organization between epidemiology and immunogenetics datasets. For example, in the OBiBa stack, each dataset is usually one line per subject, while in AIRR Data Commons datasets, there are millions of sequences per subject (see Figure 3.17).

These differences did not favor Opal's solution for data storage, since OBiBa's solution initially was more focused on epidemiological studies.

The tests performed raised the consideration that processing times can become fairly high even with modest numbers of samples (on average, 10 samples take about 5 minutes).

Once performance can be further improved, it can be useful to perform tests retrieving data directly from a iReceptor+ repository, i.e. not using Opal for data storage.

This proposal was suggested before the development of a feature known as "Resources"[28], that we will explore in the proposed solution, and limitations of the use of Opal to data store were related.

[27] https://immunarch.com/reference/split_to_kmers.html?q=getkmers#arguments.
[28] https://github.com/obiba/resourcer.

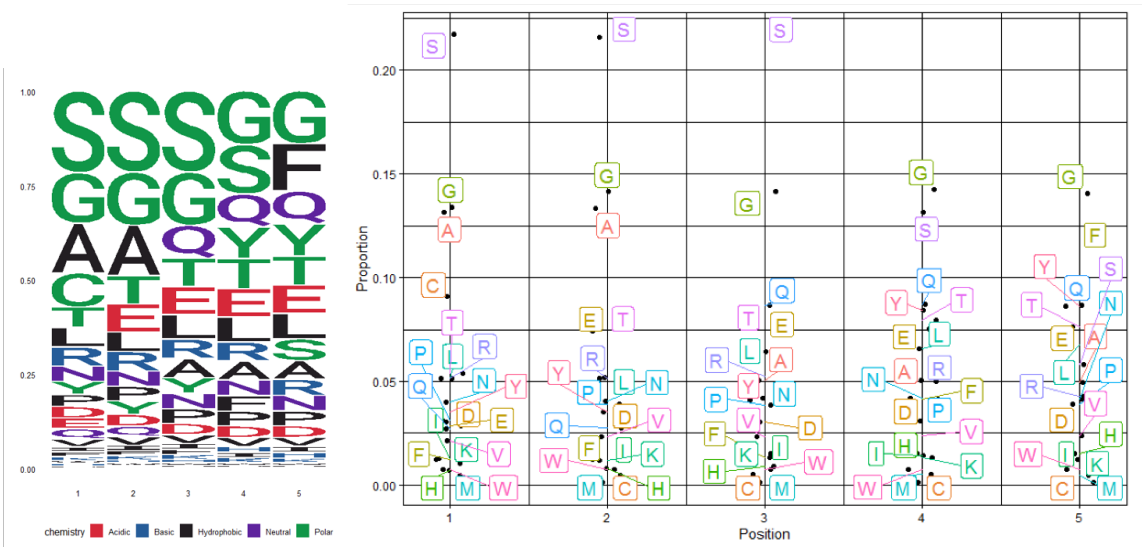

Figure 3.15: Example of K-mers analysis based on a k size = 5, including frequency matrix.

Number of Samples	Number of Clonotypes	Process duration (avg)
1	69,402	0.24 mins
6	321,602	1.61 mins
6 (*)	1,140,494	4.29 mins
12	763,607	3.37 mins
18	1,904,101	8.35 mins

(*) – Different data sequencing project

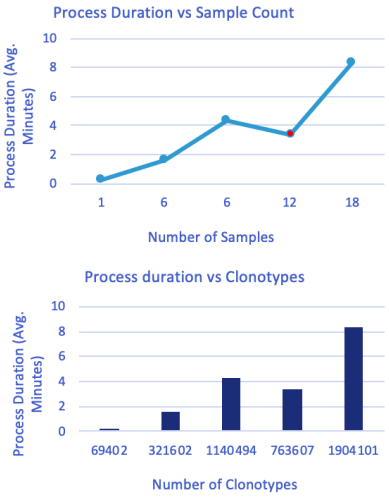

Figure 3.16: Benchmarking and Feasibility of Motif Extraction.

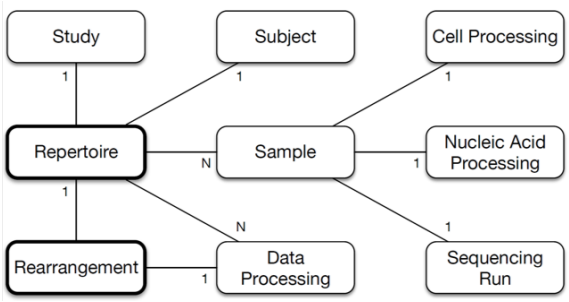

Figure 3.17: AIRR Data Commons datasets: millions of sequences per subject.

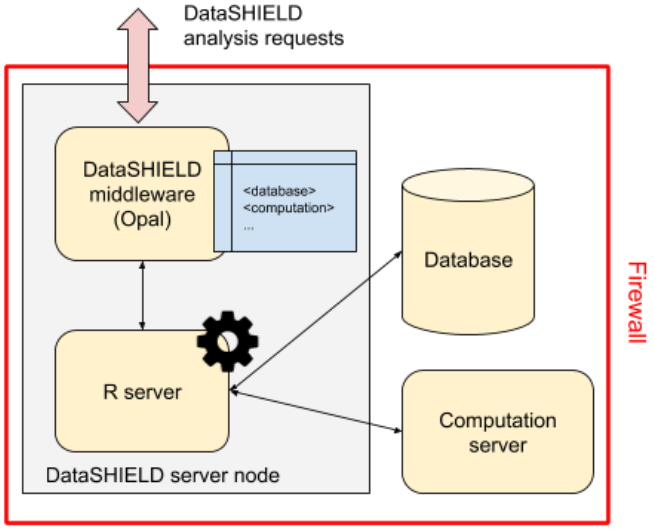

Figure 3.18: Resource R package architecture. From OBiBa.org.

3.1.2.3 Resource R

In the beginning of 2020, OBiBa announced a new feature for the Opal/DataSHIELD solution: the Resource. A Resource, in this context, can represent datasets or computation units which location is described by a URL and access is protected by credentials. Instead of storing the data in Opal's database, the datasets, usually in a non-relational representation, are kept in their original format and location. These Resources can take on different formats such as database resources, file resources, computation resources, can be defined in Opal. Opal is also responsible to takes care of the DataSHIELD permissions and the resources assigned to a R/DataSHIELD session. Once a DataSHIELD session is defined, it is possible for the researchers to combine different datasets in their analyses.

The Resource R package (see Figure 3.18) provides access to resources and circumvents some of Opal's performance limitations. Resource R will enable connections to several resources.

The Resource R package allows the researchers to deal a wide range of data sources (using tidyverse, DBI, dplyr, sparklyr, MongoDB, AWS S3, SSH, etc.) and is extensible to new ones.

The Resource R package is extensible and enables to work with specific R data classes, such as Omics data structure classes. New data types can be created as well. You can see an example of this extensibility in packages such as dsOmics[29] and dsOmicsClient[30]. Generally speaking, any data format and storage that can be read by R can be expressed as a resource. Opal works like a register of these resources, not storing the data itself, but all metadata necessary to access the resource.

[29] https://github.com/isglobal-brge/dsOmics.
[30] https://github.com/isglobal-brge/dsOmicsClient.

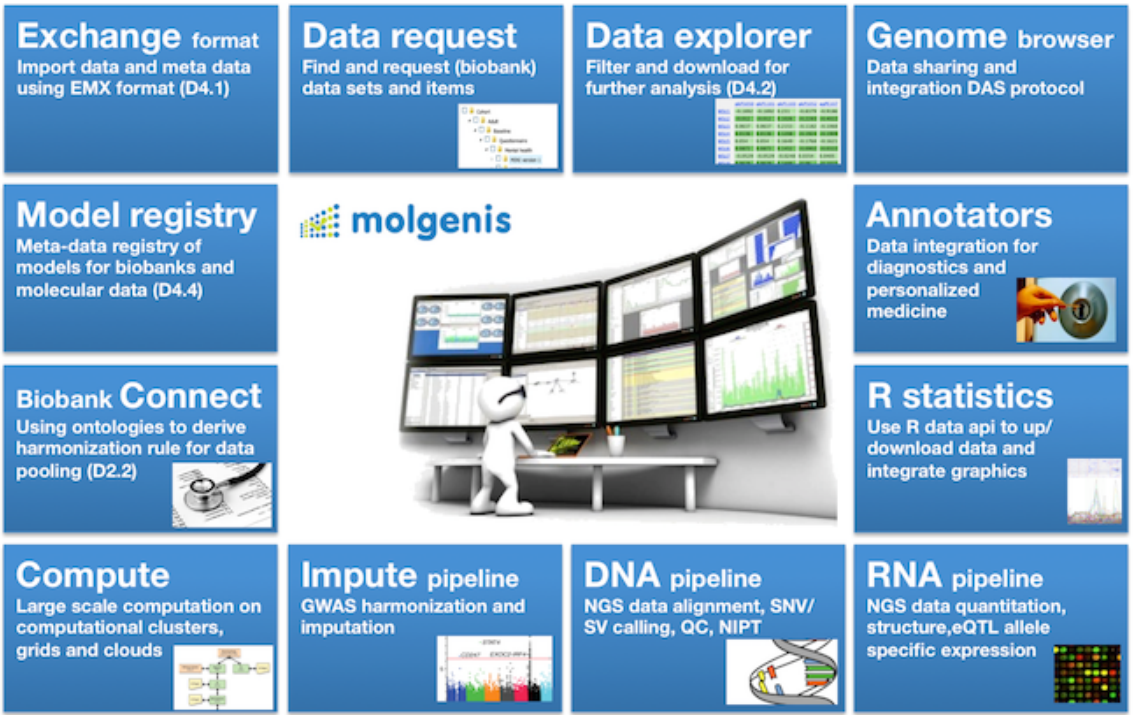

Figure 3.19: MOLGENIS overview and its modular architecture. From molgenis.org.

3.1.3 The MOLGENIS solution

MOLGENIS[31] is a modular web application focused initially on molecular genetics research, but has grown to other uses, such as patient registries, rare disease research, and biobanking. It is developed and maintained by the Genomics Coordination Center (GCC), from the University Medical Center Groningen, Netherlands. MOLGENIS is capable of capturing, exchanging and exploiting large datasets, and runs on a scalable software infrastructure.

One of the key features is that it has a customizable data system, allowing researchers to model the data according to their needs. Figure 3.19 depicts MOLGENIS modularity that allows researchers to use or create extensions modules, such as R and Python scripts, to store and interact with the data. This enables the researchers to add their own statistical modules to run statistical analysis, or create plots based on their data within an online environment. MOLGENIS deals with storing data, and offers filters and fast search capabilities.

MOLGENIS helps the researcher to implement the FAIR principles by providing the following features:

- **Structured Data Management**: Model, capture, and manage data. Data can be imported by forms or data files, such as CSV. Data and metadata modeling can be refined dynamically.
- **FAIR Data Sharing**: MOLGENIS enables the researcher to create views for their datasets and variables to the outside world while preventing exposure of (sensitive) data values using the fine-grained permission system.

[31] https://molgenis.org/.

- **Secure Access**: MOLGENIS provides controls by group, role and individual access. Authentication can be done using Google two-factor authentication or using SURFconext (Netherlands) [32] and BBMRI/ELIXIR AAI (Europe) [33].

- **Scripting and Visualisation**: Bioinformaticians and researchers can add scripts (e.g. R, javascript, python) and connect to the data using API's to add analysis tools and views.

- **Harmonization and Integration**: MOLGENIS aims to promote interoperability. MOLGENIS offers 'FAIRification' (the process to make data FAIR) tools to find related data, codify data contents and transform different tables into one standardized table. It helps to promote combined analysis, more powerful than running smaller analyses on each dataset separately.

- **Task Automation**: It is possible to automate data upload, transformation and statistics scripts. Frequently data from multiple sources must be combined.

- **Questionnaires**: The questionnaire tool provides chapters, sub questions, advanced validations, conditional or 'skip' questions and intermediate save.

- **High-Performance Computing**: Schedule large scale analysis jobs on a computer cluster. MOLGENIS does also provide a high-performance computing (HPC) framework that works with OpenPBS[34] and SLURM[35].

The MOLGENIS team has developed a DataSHIELD implementation, known as Armadillo, within the MOLGENIS suite [36] [37] [38]. Like in the OBiBa solution, the "FAIRification" of MOLGENIS datasets is achieved by using the DataSHIELD solution.

3.1.4 The GA4GH solutions

The Global Alliance for Genomics and Health (GA4GH) was established in 2013, focused on discussions about responsible and effective sharing of genomic and clinical data. To support these premises, the conclusion was that data underlying genomic medicine must be federated. This hypothesis is anchored on the framework document developed by the GA4GH Regulatory and Ethics Working Group (REWG), that provides principles and core aspects for responsible data sharing[5].

The GA4GH Data Working Group (DWG) has developed a standardized API, which offers a defined protocol to allow disparate technology services of institutions to communicate with each other to exchange genotypic and phenotypic information.

[32] https://www.surf.nl/en/surfconext-global-access-with-1-set-of-credentials.
[33] https://wiki.egi.eu/wiki/Competence_centre_BBMRI.
[34] https://www.openpbs.org/.
[35] https://slurm.schedmd.com/.
[36] https://github.com/molgenis/molgenis-service-armadillo.
[37] https://github.com/molgenis/molgenis-r-armadillo.
[38] https://github.com/molgenis/molgenis-r-datashield.

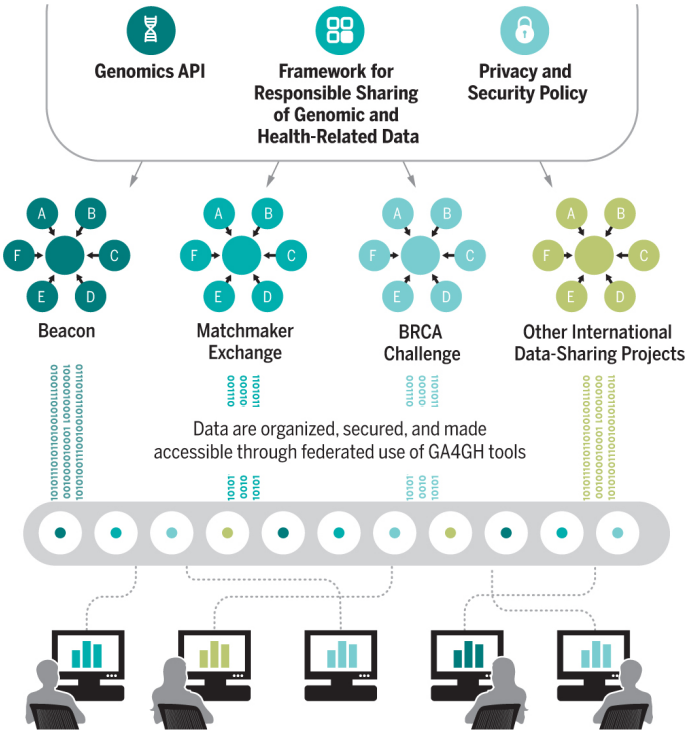

Figure 3.20: The GA4GH federated ecosystem. From ga4gh.org.

GA4GH members developed several projects using this API and the framework document, as we can see on Figure 3.20. One of promising projects based on these artifacts is the Beacon Project[39]. It is a driver project of the GA4GH and supported through ELIXIR[40]. ELIXIR is an european intergovernmental organisation whose goal is to coordinate life science resources, that include databases, tools, training materials, cloud storage, so that they form a single infrastructure.

The Beacon Project is a web application that allows researchers to query and determine whether they contain a genetic variant of interest. The solution is capable of answering questions, such as of the examples below, with a yes/no:

- “Do you have any genomes with an ‘L’ at position ’U’ on chromosome ’M’?”
- "Does this dataset contain an allele ’R’ at ’W’ genomic position?"

The project is being expanded to provide other advanced query options and answers, and some information retrieved can also be useful alongside additional metadata, including allele frequencies, pathogenicity scores, and phenotypic information associated with the queried allele. These additional features in the future may include other advanced analyses, such as: identify different types of genomic variants, e.g. copy number variations (CNVs) or fusion events; flexible variant queries (e.g. genomic “range queries”); identify clinical and other biological parameters (e.g. phenotypes, diagnoses, time related data, geodata).

[39] https://beacon-project.io/.
[40] https://elixir-europe.org/about-us/commissioned-services/beacons.

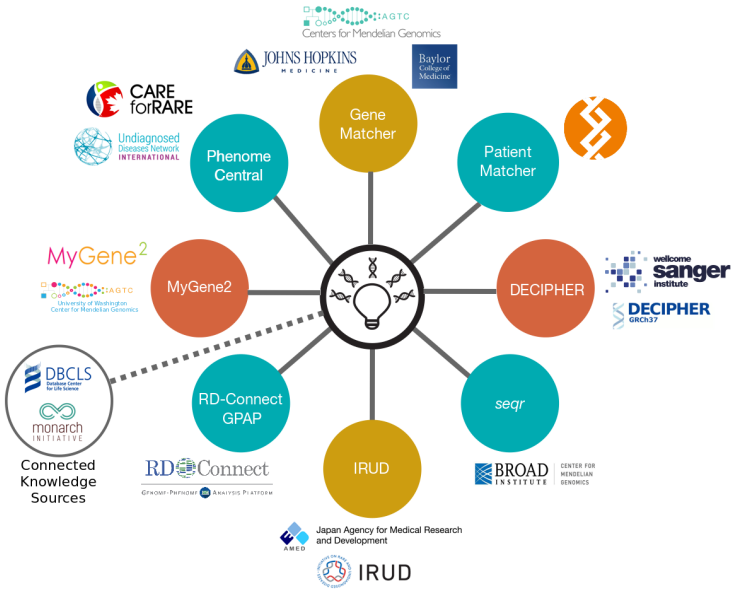

Figure 3.21: The MME data repositories. From matchmakerexchange.org.

The Beacon protocol was not designed to deliver data beyond the aggregated responses to Beacon queries. However, extensions of the protocol may provide mechanisms to deliver to other protocols and services which could provide such additional functionality, preserving constraints included in the framework document developed by GA4GH.

The Matchmaker Exchange (MME)[13] is a solution more focused on facilitating the discovery and analyses in rare diseases databases. For example, the MME solution helps the researchers to identify cases with phenotypes and disrupted genes in common.

After consortia established an API, several matchmaker services as we can see on Figure 3.21, have implemented it. GeneMatcher[41], Phenome Central[1], and DECIPHER[4] are some examples. The Human Phenotype Ontology[42] is used to ensure an accurate comparison of patients.

One important aspect is that successful matching increases as the volume of cases through MME data repositories increases. MME has already led to the diagnosis of several previously undiscovered rare diseases.

MME helps to find significant correlations, matching in which the genotype aspects of matching can occur by direct query of variants within a VCF that meet certain criteria.

MME is compliant with the GA4GH Regulatory and Ethics Working Group (REWG), so a mechanism and policy were developed to define the type of consent needed for using MME and when no consent is needed. For example, if the data are associated with a unique or sensitive

[41] https://genematcher.org/.
[42] https://hpo.jax.org/app/.

phenotype or with sequence-level data, consent from the patient is required to share it for research purposes. However, if only standard phenotype terms and candidate gene names are used, consent to clinical care allows for matchmaking. Even so, challenges remain in balancing discovery with privacy and data protection.

The BRCA Challenge aims to advance the understanding of the genetic basis of breast, ovarian, and other cancers that are driven by germline variants in BRCA1 and BRCA2 (Breast cancer type 1 or 2 susceptibility protein is a protein that in humans is encoded by the BRCA1/BRCA2 gene). The project's product is the BRCA Exchange[43], a public web portal to access curated, expert interpretations of BRCA1/2 genetic variants, as well as supporting evidence. Like others GA4GH projects, BRCA Exchange has a public API[44].

Liability concerns faced by federated databases of this kind, such as misclassifications or failure to regularly validate and update classifications, are permanent discussion topics by BRCA Exchange team members. It is expected that a variety of issues arise when data must cross multiple domains, such as the aspects that involve patient privacy, individual academic success in gene discovery, distinct international laws, etc.

The GA4GH highlights that the scalability of all solutions that involve omics data is a big challenge. It occurs because for every problem there will be domain-specific challenges that may require uniquely applicable tools.

For instance, GA4GH says that the field of dementia research may demand new solutions that integrate data from brain Magnetic Resonance Imaging (MRI) technology. Applying existing GA4GH approaches in new contexts will require solutions that are portable, customizable, and interoperable. GA4GH must also focus on solutions that can benefit many different spectrums, such as specific patient groups, jurisdictions, health systems, and environmental and socioeconomic realities.

3.2 Reproducible Bioinformatics Pipeline Tools

Bioinformatics pipeline tools often involve a number of heterogeneous steps, from applying various command-line tools, such as provided by the Immcantation framework[45], combined with script languages, such as Python, AWK, or R, for the pre-processing, population structure determination, and repertoire analysis. There is a high diversity of pipeline tools available. A curated list can be found on *"Awesome Pipeline"*[46].

It is broadly desirable that Bioinformatics pipelines should be modeling in a reproducible way. Reproducibility enables technical validation and regeneration of results over time. Other requirements, such as Portability (i.e., ability to run in different platforms), Scalability (i.e., ability to deploy big distributed workloads), Usability (i.e., minimizing the complexity of deployment

[43] https://brcaexchange.org/.
[44] https://brcaexchange.org/about/api.
[45] https://immcantation.readthedocs.io/en/stable/.
[46] https://github.com/pditommaso/awesome-pipeline.

Table 3.1: Diversity of Scientific Pipeline Tools.

Group	Tools	Characteristics
1	Galaxy, Watchdog, KNIME	Offer graphical user interfaces for composition and execution of workflows, smoothing the learning curve and making it accessible for people with no programming skills.
2	SCOOP, Ruffus, Pwrake, Hyperloom, COMPSs, Jug, Balsam, Anduril, SciPipe	Workflows are specified using specialized packages available in script languages such as Scala, Python, and others. Such systems can be used in server environments, and that workflows can be managed and easily shared in version control systems.
3	Snakemake, BioQueue, Nextflow, Cylic, Bpipe, BigDataScript, Clusterflow	Workflows are specified using Domain Specific Languages (DSL). It contains all advantages of group 2, adding the additional benefit of improved readability. DSL provides annotations that explicitly model central elements of workflow management, thereby preventing excessive operators or boilerplate code.
4	Popper	Workflows are specified declaratively, using configuration file formats like YAML, particularly readable for non-developers. These declarative tools share some benefits with the third group, but the declarative format can be more restrictive in the processes that can be expressed.
5	System-independent workflow specification languages like CWL and WDL	These define a standard syntax for describing workflows, which can be interpreted and performed by arbitrary runners. Similar to the fourth group, a downside is that imperative or functional programming is not or less integrated into the specification language, thereby limiting the expressive power.

workloads instead of adding new ones), and Consistency (i.e., track changes and revisions consistently for code, config files, and binary dependencies), are also necessary to guarantee a robust Bioinformatics pipeline.

MOLDER [10] classifies these tools in five groups, listed on Table 3.1. Several of the mentioned tools support the full reproducibility of pipelines (e.g., Nextflow[47], Snakemake[48], Galaxy[49]) by enabling the description of each processing step, including all expected inputs and outputs. Additionally, these tools allow a scalable execution, including deploying the software stack in container technologies, such as Docker[50] and Singularity[51].

It is not the purpose of this Section to present an exhaustive comparative list of all pipeline tools. To summarize, the tools from group 3, including those that have an integration with version

[47] https://www.nextflow.io/.
[48] https://snakemake.readthedocs.io/en/stable/.
[49] https://usegalaxy.org/.
[50] https://www.docker.com/.
[51] https://singularity.hpcng.org/.

control systems, are the most suitable for dealing with bioinformatics pipelines like those used by the iReceptor+ project (see Section 3.1.1), as such tools allow the needed flexibility in writing pipelines while providing the requirements of reproducibility, portability, scalability, usability, and consistency. Of this list, and according to Google Trends[52], Nextflow and Snakemake are more widespread until the present date.

3.3 Summary

This chapter presented known initiatives about privacy-preserving analyses and some previous related work. Tools to deal with reproducible pipeline tools were also discussed.

[52] https://trends.google.pt.

Chapter 4

Proposed solution

The proposed solution will suggest a suitable architecture to deal with privacy-preserving analysis, leveraging the existing solution's capabilities to deal with Omics data. In Section 4.1 we will show the main aspects that must be considered, including a proposal to deal with the lack of information about how the Omics data were generated.

4.1 Requirements

A typical architecture of a solution that adheres to the FAIR principles and is compliant with the GDPR, must address several questions. The first assumption is that health data repositories are disorganized, population biased, hard to integrate and search. Furthermore, data needs to be analyzed together to accomplish statistical power with high diversity in individuals, populations, and environments, relating each profile to health and disease.

Proceeding, regulations about data-privacy are not ready to deal with secure sharing of data, especially across countries. When data protection and governance rules discourage centralized storage, the data cannot be shared. Despite much progress, these data sets are still collected in clusters: by institution, by disease, by consortium, by country.

Another relevant assumption is that the methods used by institutions to collect cohort data are heterogeneous. In this way, to use these data in a combined analysis requires the addition of data descriptors and/or, if necessary, harmonization of these data using research-ready core variables that use measurements similar enough to be analyzed in unison.

Of the four FAIR Principles, Interoperability is the most difficult to accomplish[16]. There are many standards emerging, addressing several aspects of Interoperability. Also related to this point, data in the health sciences are vastly diverse, ranging from those designed for particular-purpose, to those that are for general-purpose.

Data management requires different privacy levels, including security considerations. For example, clinical analyses about genetic mutations in humans are especially sensitive, since involves

personal data, while analyses of other species are not. The diversity of models across repositories and the different level of detail, including (or not) metadata, makes the integration and analysis of these data a hard effort with low scalability.

Current implementations of privacy-preserving analysis frameworks seem to be limited when data becomes very large (millions of rows, hundreds of variables). Biological sample data, collected by high-throughput technologies, such as Next Generation Sequencing (NGS), which allows the sequencing of entire genomes, are examples of this kind of data[9].

The Omics technologies intend to produce a systematic identification of all mRNA (transcriptomics), proteins (proteomics), and metabolites (metabolomics), present in a given biological sample. In the particular case of Omics data, these data are produced by computational workflows known as bioinformatics pipelines. The reproducibility of these pipelines is hard and it is often underestimated. Nevertheless, it is important to generate trust in scientific results, and therefore, it is fundamental to know how these Omics data were generated or obtained.

AIRR-seq data are mostly private data from patients and, therefore, only privacy-preserving analyses can be done. Data usage typically goes through ethic boards, requiring data stewards at given institutions to be confident that data are treated securely from several institutions[2]. A federated data model enables a data steward to curate, maintain, and share data as appropriate to the study's ethics and common agreements, while at the same time having visible control over who has access to those data[2].

For example, the iReceptor+ project (see Section 3.1.1) aims to follow a distributed, federated architecture. Their approach is not fully distributed, but intermediately distributed, where repositories are distributed and separately maintained, but are connected by a central resource and/or share technical service components such as a registry. A distributed data model, although difficult to support, is critical to the success of research in this area. One of the objectives of iReceptor+ is to hide the bureaucratic complexities, while empowering AIRR-seq researchers to perform sophisticated, and in many cases, computationally expensive, analyses on federated data from multiple, distributed repositories[2].

To address some of these challenges, a suitable implementation of a privacy-preserving health data analysis framework must be designed accurately. The proposed solution aims to identify these characteristics and generalize existing privacy-preserving analytic capabilities for large/complex data, including Omics data.

The following list displays questions that must be addressed in designing such a solution:

Privacy-preserving questions

1. How to preserve sensitive data?
2. How to enable non-disclosive analysis in the context of Omics data?
3. How to be adherent to GDPR regulation in a Omics scenario?

Interoperability questions

4. How to deal with any type (format) of data?
5. How to avoid a specific solution, or in other words, how to achieve a generic and open solution?

FAIR and TRUST principles questions

6. How to be adherent to FAIR principles?
7. How to be adherent to TRUST principles?

Omics data questions

8. How to deal with large unstructured data (millions of rows, hundreds of variables), such as Omics data?
9. How to improve the reproducibility of Bioinformatics Pipelines to generate trust in scientific results?

Federated architecture questions

10. How to avoid, whenever possible, duplicate data?
11. Do centralized and federated approaches allow institutions the same level of autonomy?
12. How to make institutions autonomous in storing data in a way that non-disclosive collaboration with other partners becomes achievable?
13. How to correlate information between different data sources?

In an initial analysis, we observed that the incorporation of FAIR principles for optimal reuse of existing data, calls for an approach where sensitive data kept locally and only metadata and aggregated results being shared. Secondly, since governance rules discourage central pooling, it is necessary to adopt the federated approach.

INESC TEC is a partner of research consortiums, such as EUCAN-Connect and iReceptor+. The participation in multiple projects enables the INESC TEC team to suggest solutions that are not directly involved in the context of a specific consortium. For example, EUCAN-Connect have partners that are maintainers from the OBiBa solution (see Section 3.1.2) and MOLGENIS solution (see Section 3.1.3), while iReceptor+ promotes the iReceptor+ platform (see Section 3.1.1).

INESC TEC is not directly involved with the GA4GH project (see Section 3.1.4), so a deep understanding of its architecture, and even its usage, was not explored in the scope of this dissertation. Nevertheless, genomic data sharing initiatives such as GA4GH, can benefit directly from the DataSHIELD architecture[18].

Non-disclosive analyses (see Section 2.7) is a core aspect of OBiBa and MOLGENIS solutions, who benefit from DataSHIELD (see Section 3.1.2.1) integration. The robustness of the DataSHIELD solution to deal with non-disclosive analyses cannot be ignored.

Moving forward, the implementation of the Resources feature (see Section 3.1.2.3) by OBiBa opens new possibilities, such as making non-disclosive analyses using large datasets. The Resources feature was not used in previous tests executed by in Section 3.1.2.2.

INESC TEC also developed the Coral distribution[1], that is comprised of a series of Docker containers, built using images originally created by OBiBa, adapted and customized for ease of configuration and deployment, which integrates additional features, such as proxy support and monitoring.

These developments favor the OBiBa solution (using the Coral distribution) in conjunction with DataSHIELD to leverage the promising results of the distributed privacy-preserving analyses of Omics data.

Although Omics data can be referred to as resources in the solution we intend to propose, it is still necessary to understand how such resources were originated and which pipelines and parameters were used.

The lack of information about how a pipeline was executed and how the outputs were generated are critical ingredients that can affect the robustness and reliability of an analysis. Another scenario where reproducibility is crucial is when data sets require auditing to prevent result manipulation. This is especially critical when this kind of auditing is done by an automated tool.

4.2 Methodology

While understanding the OBiBa solution and DataSHIELD was quite simple, since we developed the Coral distribution, understanding the Bioinformatics pipelines (see Section 2.2) required some extra effort and it was necessary to involve other stakeholders.

We purposely requested a complex pipeline example from an iReceptor+ member (related to the project PRJNA368623[2]). This pipeline was developed using a set of tools, such as AWK, IgBlast, Python, and several specialized packages from the Immcantation framework.

In order to run such a pipeline, a virtual machine was made available to install and execute these tools and execute the required pipeline. The execution takes some time (after several adjustments, the entire process takes more than one week), even with a virtual machine with the following configuration: 8 vCPUs, 20GB memory RAM, and 500GB storage.

After all necessary dependencies were installed successfully, we developed a mind map to understand the workflow of the processing steps, also taking the opportunity to assimilate some concepts. This mind map is shown in Figure 4.1.

The development of this mind map gave us some insights, listed below:

- Each step of a typical pipeline receives inputs and produces outputs.
- A step usually manipulates files. Intermediary steps produce intermediary files, and these files can be useful or not.

[1] https://coral.inesctec.pt/.
[2] https://www.ncbi.nlm.nih.gov/sra/?term=PRJNA368623.

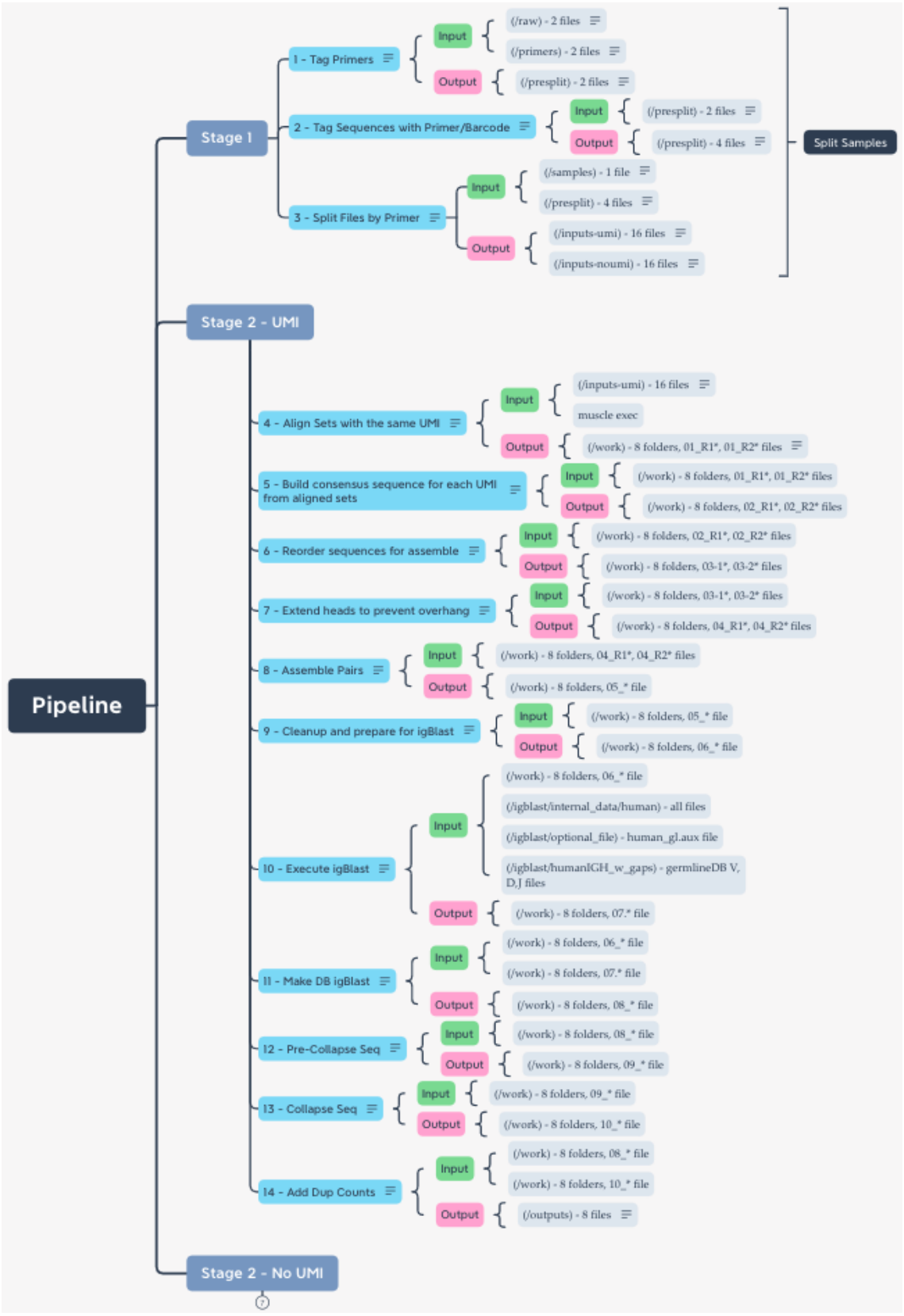

Figure 4.1: A mind map that represents a complex Bioinformatics pipeline.

- The “raw sequences”, or “raw data” are the input for the first step in a pipeline.
- A stage is a set of steps, but there are pipelines without stages. It is an abstract grouping.
- A stage can be ordered, but can exists parallel stages.
- A step can read multiple input files.
- A step can produce multiple output files.
- Some input files can be from external sources, e.g., files related to “germlines” or “primers”.
- Steps are executed following an order, but a step can invoke tools that create multiple processes.
- “Loop” operators can appear in a step to create multiple processes.
- A step is composed of commands.
- One command receives parameters or options. These options can affect the command results and the outputs of a step.
- A command can be a custom script or a command derived from a tool.
- A tool has a version. Different tool versions can produce different outputs.
- The order of command parameters is important.
- The last step usually produces the output files in which researchers are interested for making analyses.

In summary, the multiple components of a Bioinformatics Pipeline frequently have dependencies on different software run-times, parameters, and, in some instances, different versions of the same software. Additionally, there are external dependencies that evolve along time, such as germline reference databases. The pipelines can result in a complex software ecosystem with a lot of commands, options, and versions.

Having the various processing steps and execution parameters documented in a machine-readable and relational model format, enables this kind of process to be more consistent, accessible, robust and reliable.

4.3 Architecture

The proposed solution will use the OBiBa stack (using the Coral distribution[3]) in conjunction with DataSHIELD to leverage the promising results for distributed, privacy-preserving analyses of Omics data.

[3] https://coral.inesctec.pt/.

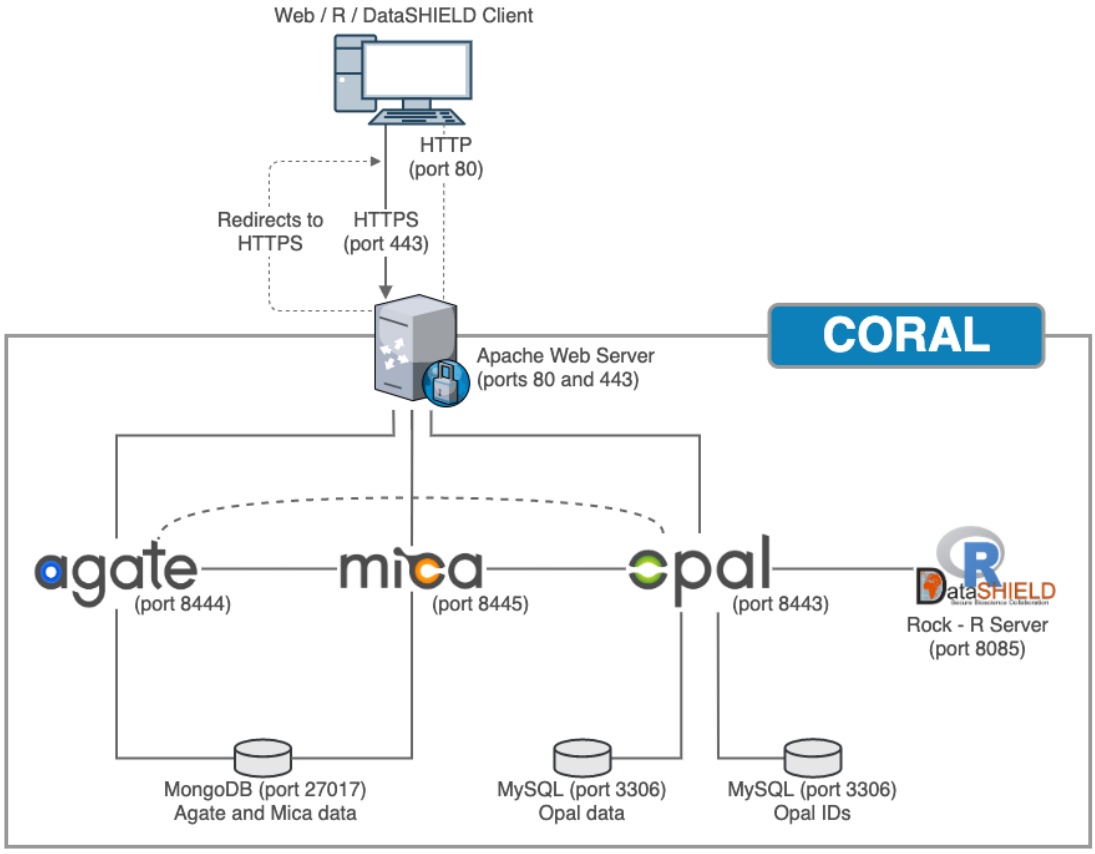

Figure 4.2: Coral Architecture.

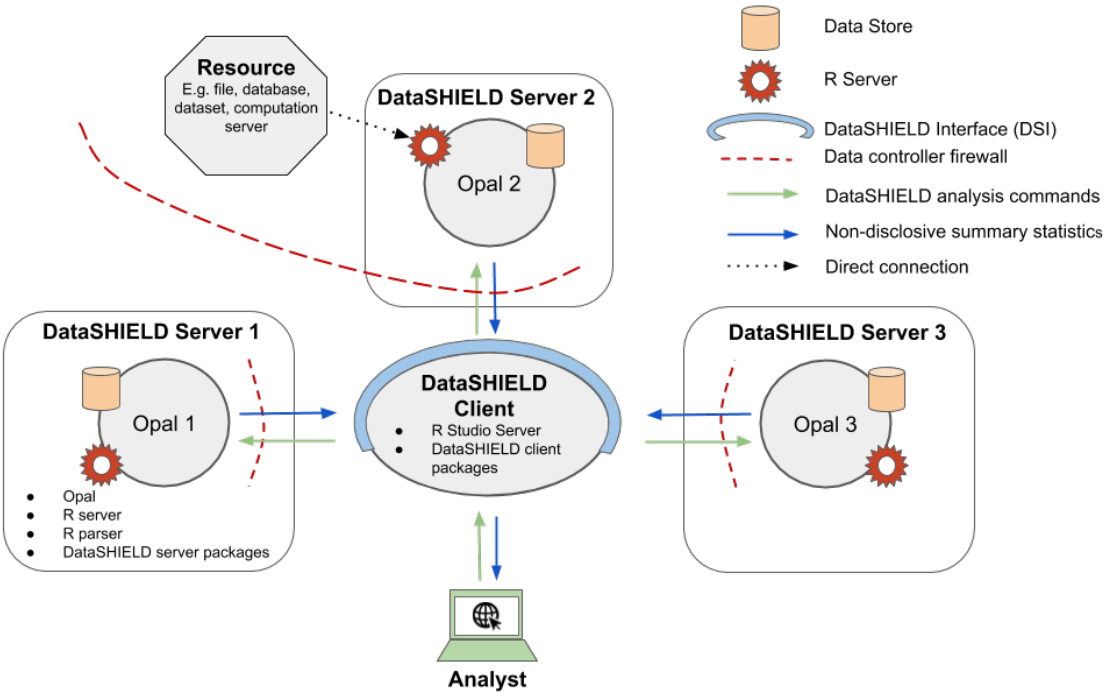

Figure 4.3: Deployment Architecture. By DataSHIELD team.

As depicted in Figure 4.2, the Apache container acts as the only entry point for the whole system. All incoming requests must go through the Apache server acting as a reverse proxy for Agate, Opal, and Mica. In the context of privacy-preserving analysis of Omics data, Agate and Mica are not so relevant the Opal and R Server. The Resources are configured in Opal.

The Resource feature (see Section 3.1.2.3) will be a key piece of this architecture. The Resource integration can be visualized on Figure 4.3.

In this scenario, each Opal box represents a server where a Coral stack is installed, including Opal and R Server services, following a federated architecture. Each Opal instance is responsible for maintaining a list of its own resources.

DataSHIELD is composed by client and server packages. A DataSHIELD server package (dsBase package[4]) is installed on the R Server that is integrated with Opal. The dsBaseClient package[5], needs to be installed on the Analyst/Researcher client machine. Both package need to be used in conjunction to do the analyses.

A recent and relevant development is that since Opal v4.x, the R Server (now called "Rock Server") can be horizontally scalable[6], which means that, as the computation load grows, horizontal scalability will allow the expansion of the number of Rock servers, creating a Rock Cluster. It is a valuable feature since the execution of all privacy-preserving analysis occurs in the Rock Cluster.

[4] https://github.com/datashield/dsBase.
[5] https://github.com/datashield/dsBaseClient.
[6] https://rockdoc.obiba.org/en/latest/introduction.html#scalability.

Using Resources (see Section 3.1.2.3), it is possible to register the output files of interest, generated by the Bioinformatics pipelines, i.e., the alignments, clonotypes, receptor files, etc. This way, it will enable these resources, or datasets, to be used by DataSHIELD in privacy-preserving analyses.

4.4 Reproducibility: Relational model

After the distress of running the pipeline completely, it was decided to create a model that would allow documenting the entire pipeline execution process, as well as helping with traceability and reproducibility. We developed a relational model to represent the execution process. The insights obtained from the mind map of Figure 4.1 were useful to detail the proposed relational database schema (see Figure 4.4).

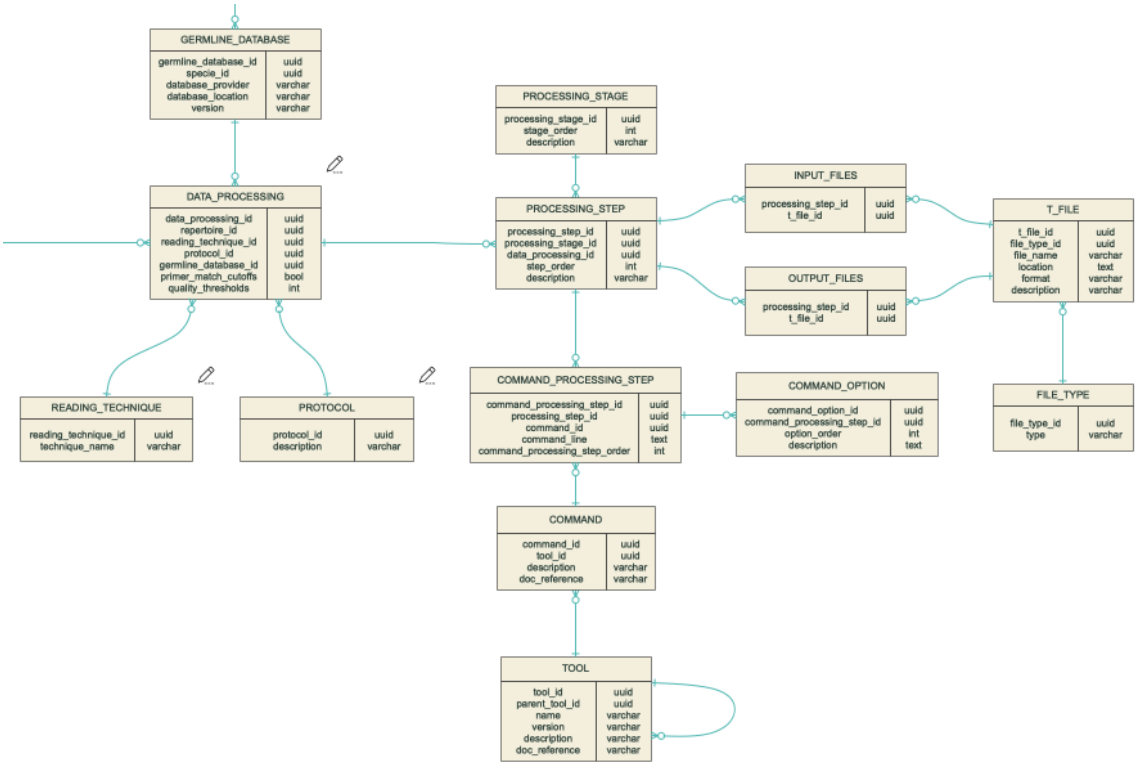

Figure 4.4: Database schema to store the Pipeline's execution metadata.

Figure 4.4 does not represent the entire model. The focus here is to show the entities related with the pipeline execution process. This model can be complemented with a more thorough representation that includes other pre-processing entities, or entities related to the Bioinformatics itself, such as Organism, Species, Subject, Repertoire. We developed a simplified version containing these other entities (see the Appendix A). Table 4.1 summarizes a description of each table of the database schema.

Table 4.1: Reproducibility of a Bioinformatics Pipeline

Entity	**Usage**
DATA_PROCESSING	Represents the execution of a pipeline. One Repertoire can contains multiple data processing.
READING_TECHNIQUE	Represents the technique used to read a fragment. Usually, it is Single-end or Paired-end.
PROTOCOL	Refers to the library preparation protocols used by the data processing.
GERMLINE_DATABASE	Represents the germline provider associated with the species referred on the subject. E.g.: IMGT, Homo Sapiens, version x.x.
PROCESSING_STEP	Represents a data processing step. E.g.: "Assemble Data".
PROCESSING_STAGE	Represents a set of processing steps. Filling it is optional.
INPUT_FILES	All the necessary input files to execute a processing step, such as raw sequences, files with primers, etc.
OUTPUT_FILES	All the output files generated/manipulated by a processing step, such as intermediary files generated by processing steps, rearrangement files, etc.
COMMAND_PROCESSING_STEP	Represents the commands associated with one processing step.
COMMAND	Represents a command used from a tool. E.g.: Tool pRESTO, command AlignSets.
COMMAND_OPTION	Represents the list of parameters used by a command in a processing step.
TOOL	Represents the Tools used by the Pipelines. E.g., Python, pRESTO (parent tool is Python), IgBlast, MIxCR, AWK, etc.
T_FILE	Represents the list of files used/created/manipulated by the pipeline in each processing step.
FILE_TYPE	Defines the types of files used during the pipeline. E.g.: RAW Data, Primer Data, Rearrangement Data, other intermediary processing Step Data, etc.

A Data Processing registry can be associated to a Repertoire. Accordingly with the iReceptor+ definition, a Repertoire is an abstract organizational unit of analysis that is defined by the researcher and consists of study metadata, subject metadata, sample metadata, so on, including a set of raw sequence files, data processing metadata, and a set of rearrangements[7].

The best way to populate or search registers in this database is through a REST API. Once this is done, the registers can be inserted by the pipelines during the execution (an initial ad hoc way is to use the *curl* command[8], for example). The development of a REST API will be listed in the Fu-

[7] https://docs.airr-community.org/en/stable/datarep/metadata.html#.
[8] https://curl.se/.

ture Work (see Chapter 6). For now, it was developed a Docker stack with the database configured to run on PostgreSQL[9]. More details about this Docker stack can be found on Appendix C.

4.5 Summary

This chapter presented the proposed solution to deal with privacy-preserving analyses using a federated architecture, leveraging existing solutions to deal with Omics data. A database model focused in the minimize the reproducibility problem was also proposed.

[9] https://www.postgresql.org/.

Chapter 5

Implementation

Throughout this chapter, we will demonstrate the execution of a bioinformatics pipeline, which uses MiXCR and is more straightforward than the one provided by one of the iReceptor+ members in Section 4.2.

5.1 Introduction

The bioinformatics pipelines will produce some files, such as alignments and clonotypes. These files are usually used in scientific analyses.

To make such files (or datasets) available for review, we will need to register them as resources in Opal. The location where the files (or resources) are originally saved does not need to be the same location where Opal was installed. In Opal, the location of these resources can be referenced using a Uniform Resource Identifier (URI). These resources can be accessed, for example, through an HTTPS connection, through the SSH protocol, locally, through the Amazon S3 service, etc. Once resources have been registered, they will be available for further review.

As discussed in Section 3.1.2.1, the DataSHIELD architecture is composed of two R packages: dsBase (server) and dsBaseClient (client). The dsBase package needs to be installed on the R server associated with Opal, while the dsBaseClient is installed on the R environment (usually RStudio) on the Analyst machine.

In our setup, Opal will simply work as a resource catalog. All the processing interactions will be carried out by the DataSHIELD packages, i.e., between the R Server and the Analyst machine.

5.2 Setup

It was provided a virtual machine (PIPELINE-SERVER) for the execution of bioinformatics pipelines. On this server, all pipeline dependencies were installed, such as MiXCR, Python, Java, and other libraries.

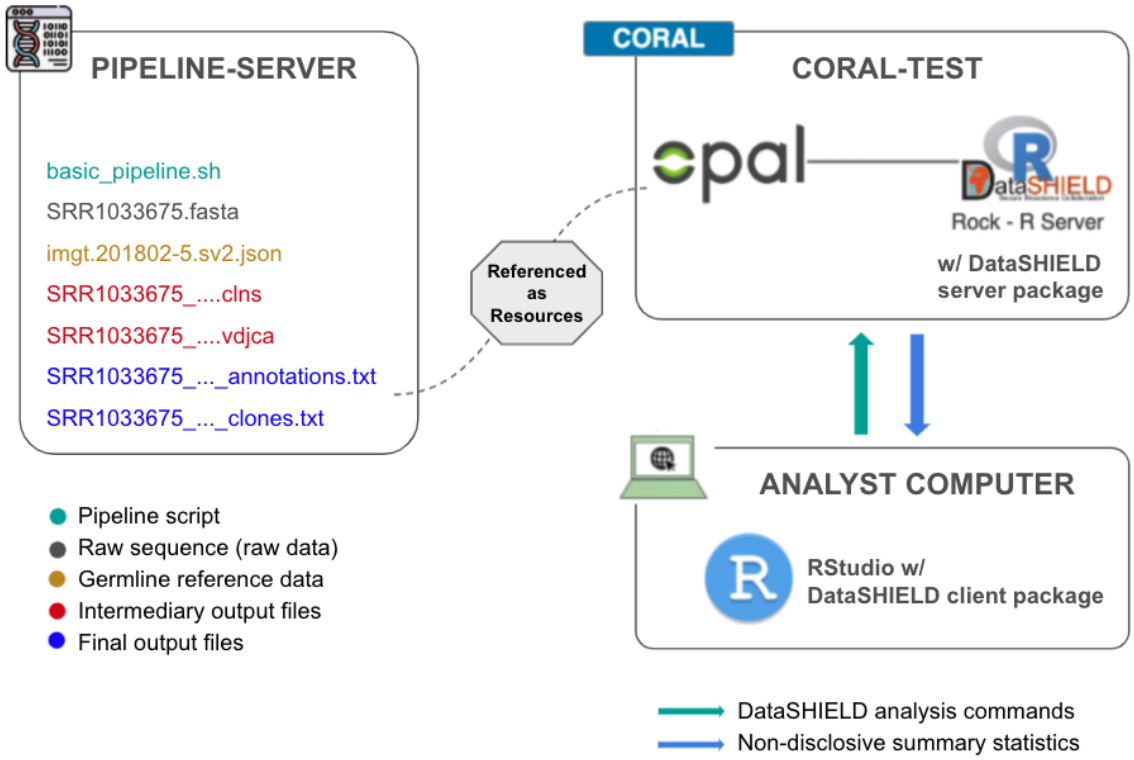

Figure 5.1: Setup configuration using pipeline outputs as resources.

Another server (CORAL-TEST) was used to install the Coral stack, which contains Opal and R Server running in separated containers (see Section 4.3). The latest versions of dsBase (DataSHIELD package server) and Resource R package are installed on the R Server during the deployment process.

The third machine of our setup is the ANALYST-COMPUTER, where RStudio is installed. In this same environment, it will also be necessary to install the dsBaseClient package, responsible for establishing the connection with dsBase, installed on the R Server. Figure 5.1 summarizes this setup.

The Figure 5.1 already shows the files generated by the execution of the pipeline, in the PIPELINE-SERVER. We will take a look at more details in the next Section.

5.3 Implementation

In our test, we will use a simple pipeline built using MiXCR. The file `SRR1033675.fasta` contains the raw sequencing data used in this pipeline. These data are public and can be accessed in PRJNA229070[1]. The germline reference can be accessed in `imgt.2018-5.sv2.json.gz`[2].

In the following script, we can see the `basic_pipeline.sh`. For instance, in the first step (Process Alignments), we can see the usage of the MiXCR tool, with the command *"align"*,

[1] https://www.omicsdi.org/dataset/omics_ena_project/PRJNA229070.
[2] https://github.com/repseqio.

receiving a param *"–library imgt.201802-5"*, related to the Germline reference database version, and a param *"-s hsa"*, related to the *Homo Sapiens* species.

```
#!/bin/bash

# Process Alignments
mixcr align -Xmx16g --verbose -f --library imgt.201802-5 -s hsa -OvParameters.
    parameters.absoluteMinScore=25 -OsaveOriginalReads=true ./raw/SRR1033675.fasta
    SRR1033675_mixcr_redo.vdjca

# Assemble Data
mixcr assemble -Xmx16g -f SRR1033675_mixcr_redo.vdjca SRR1033675_mixcr_redo.clns

# Export Assembled Alignments
mixcr exportAlignments -Xmx16g -f -c TRB -vHit -dHit -jHit -vGene -dGene -jGene -
    vFamily -dFamily -jFamily -vHitScore -dHitScore -jHitScore -nFeature CDR3 -
    aaFeature CDR3 -lengthOf CDR3 -readIds -targetSequences -descrsR1
    SRR1033675_mixcr_redo.vdjca SRR1033675_mixcr_redo_annotations.txt

# Export Clonotypes
mixcr exportClones -Xmx16g -f -c TRB -targets -vHit -dHit -jHit -cHit -vGene -dGene
     -jGene -vGenes -dGenes -jGenes -vFamily -dFamily -jFamily -vHitScore -
    dHitScore -jHitScore -cHitScore -vAlignment -dAlignment -jAlignment -cAlignment
     -nFeature CDR3 -aaFeature CDR3 -lengthOf CDR3 -cloneId -count -fraction -
    targetSequences -defaultAnchorPoints SRR1033675_mixcr_redo.clns
    SRR1033675_mixcr_redo_clones.txt
```

Listing 5.1: MiXCR pipeline basic example.

This pipeline will produce four output files, being two intermediary files (with extensions `.clns` and `.vdcja`) and two final output files, `SRR1033675_mixcr_redo_annotations.txt` (refers to the alignment sequences) and `SRR1033675_mixcr_redo_clones.txt` (refers to the clonotypes).

The relational model that we created can accommodate the traceability of this example very well. For instance, we can see the result of a simulated tracking execution in Appendix B. In an ideal scenario, the filling process of this tracking execution should be automated, i.e., without human intervention. It should be possible by using reproducible tools, such as the ones listed in Section 3.2. Therefore, the proposed relational model helps to organize and structure the script according to the domain entities described on the database schema but does not replace the usage necessity of reproducible tools. The proposed relational model is a complementary approach to enhance the reproducibility and traceability of bioinformatics pipelines.

After this pipeline is executed, we can register the output target files as resources in Opal (see Figure 5.3).

We define a resource to be a data file. A registered resource will have the following properties

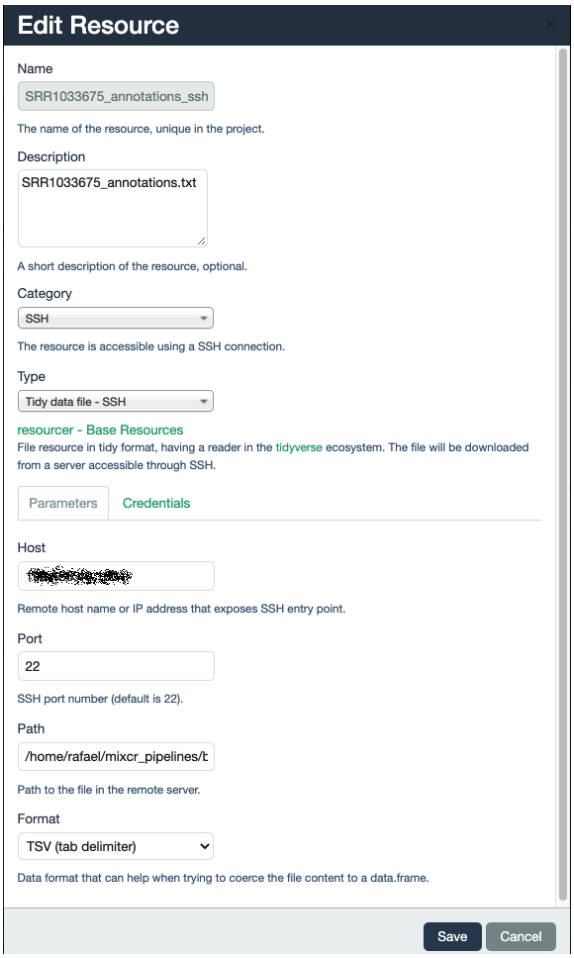

Figure 5.2: Resource properties on Opal.

(see Figure 5.2): the location of the data file; the data format; the access credentials (if applicable). The resource location description will use the Uniform Resource Identifier (URI); more specifically, the Uniform Resource Locator (URL). The URL syntax is composed of several parts:

- A *scheme* that describes how to access the Resource, e.g., HTTPS, SSH, or "s3" (for accessing Amazon Web Service S3 file store services),
- An *authority* (optional), e.g., a server name address,
- A *path* to identify the location of the Resource hierarchically.

The Resource's credentials property can be used for authenticating with a username/password, or an access token or any other credentials encoded string. The advantage of separating the credentials property from the resource location property is that a user with limited permissions could access the Resource's location information while the credentials are kept secret.

Once a resource has been formally defined, it should be possible to programmatically build a connection object that will use the data described.

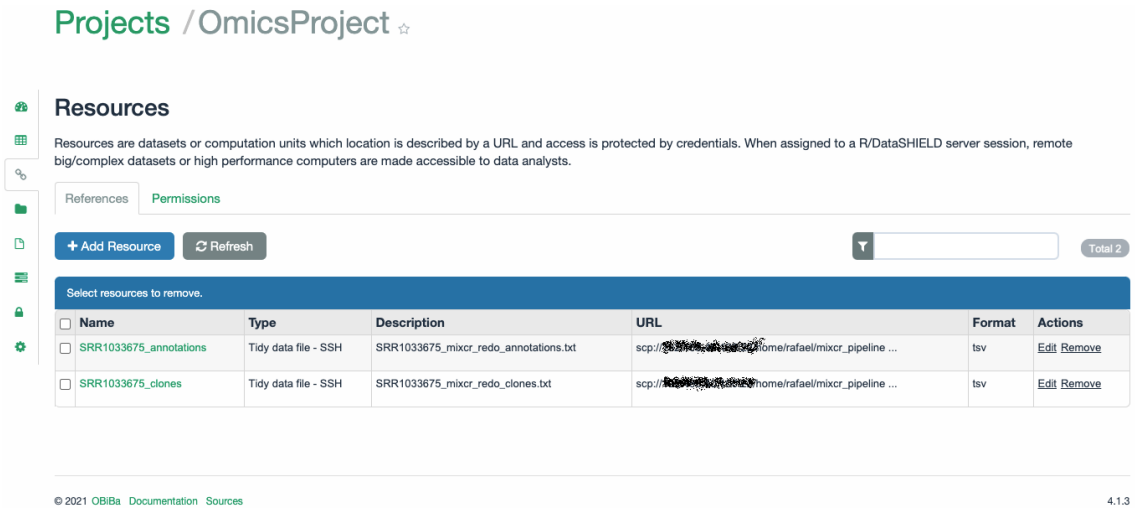

Figure 5.3: Alignments and Clones datasets registered as resources in Opal software.

Opal can register access to different types of resources, including different formats, such as CSV, TSV, R data, SQL, tiddy files. The Resources and DataSHIELD help to promote a federated architecture, avoiding duplicated data in different research centers.

With the resources registered, it is time to configure an R script in RStudio to access these resources and make them available for analyses using DataSHIELD or even more specialized packages, such as dsOmicsClient[3].

```
# Install packages
install.packages("DSOpal", dependencies = TRUE)
install.packages("dsBaseClient", repos = c("https://cloud.r-project.org", "https://
    cran.obiba.org"), dependencies = TRUE)

# Load libraries
library(DSOpal)
library(dsBaseClient)

# Opal Connection
builder <- newDSLoginBuilder()

builder$append(server = "study1", url = "https://coral-test/repo",
               user = "administrator", password = "XXXXXXXXXXX",
               resource = "OmicsProject.SRR1033675_annotations",
               driver = "OpalDriver")

logindata <- builder$build()
conns <- datashield.login(logins = logindata, assign = TRUE,
                          symbol = "res")

```

[3] https://htmlpreview.github.io.

```
#Annotations
datashield.assign.expr(conns, symbol = "annotations",
                       expr = quote(as.resource.data.frame(res)))

#Clones
datashield.assign.resource(conns, symbol = "res.clones",
                           resource = list(
                                study1 = "OmicsProject.SRR1033675_clones"
                           ))
datashield.assign.expr(conns, symbol = "clones",
                       expr = quote(as.resource.data.frame(res.clones)))

ds.ls()
ds.colnames("annotations")
ds.colnames("clones")
```

Listing 5.2: Basic R script to access the resources.

The output of *ds.ls()*, *ds.colnames("annotations")*, and *ds.colnames("clones")* can be viewed as follows:

```
> ds.ls()
  Aggregated (lsDS(search.filter = NULL, 1L))
      [=========================================] 100% / 1s
$study1
$study1$environment.searched
[1] "R_GlobalEnv"

$study1$objects.found
[1] "annotations" "clones"      "res"         "res.clones"

> ds.colnames("annotations")
  Aggregated (exists("annotations"))
      [=================================================] 100% / 1s
  Aggregated (classDS("annotations"))
      [=================================================] 100% / 0s
  Aggregated (colnamesDS("annotations"))
      [=================================================] 100% / 1s
$study1
 [1] "bestVHit"        "bestDHit"        "bestJHit"        "bestVGene"       "
     bestDGene"       "bestJGene"       "bestVFamily"     "bestDFamily"     "
     bestJFamily"     "bestVHitScore"
[11] "bestDHitScore"   "bestJHitScore"   "nSeqCDR3"        "aaSeqCDR3"       "
    lengthOfCDR3"    "readId"          "targetSequences" "descrsR1"

> ds.colnames("clones")
  Aggregated (exists("clones"))
      [=================================================] 100% / 1s
```

```
20   Aggregated (classDS("clones"))
         [==========================================================] 100% / 0s
21   Aggregated (colnamesDS("clones"))
         [==========================================================] 100% / 0s
22 $study1
23  [1] "numberOfTargets" "bestVHit"        "bestDHit"        "bestJHit"        "
      bestCHit"        "bestVGene"       "bestDGene"       "bestJGene"       "
      allVGenes"       "allDGenes"
24 [11] "allJGenes"       "bestVFamily"     "bestDFamily"     "bestJFamily"     "
     bestVHitScore"   "bestDHitScore"   "bestJHitScore"   "bestCHitScore"   "
     bestVAlignment"  "bestDAlignment"
25 [21] "bestJAlignment"  "bestCAlignment"  "nSeqCDR3"        "aaSeqCDR3"       "
     lengthOfCDR3"    "cloneId"         "cloneCount"      "cloneFraction"   "
     targetSequences" "refPoints"
```

Listing 5.3: Output of the R script.

The function `datashield.assign.expr` is responsible for accessing the resource identified on the `builder` object. Once executed, this command will be interpreted by the dsBase package, installed on R Server from CORAL-TEST. In sequence, dsBase will parse the resource name and request from Opal all the resource information, including location, format, and credentials.

After that, R Server will establish a connection with the PIPELINE-SERVER and copy the file related to the resource to a temporary and protected folder on R Server, i.e., the R Server will copy the file to the same R Server machine. It is not a security problem since the credentials to access the resources are defined in Opal, allowing access to that data. On the other hand, it can be a legal problem if the Opal and R Server run in a location subordinate to different privacy regulations, when compared with the associated resource. Therefore, it is a good practice to have a Coral environment (or an Opal/R Server) installed on a location where the data owner is also responsible, e.g., if the data owner is a research center, this same research center should have a Coral environment, with their resources registered there. This will not be a problem because Opal and DataSHIELD are fully compatible with a federated setting.

Depending on the size of the file that represents the resource, the copying process can take some time, but it will occur once time per active R Session (an R Session can be saved to be used later).

The technical advantage of having the file associated to the resource available at the same machine (or container) on the R Server is the performance. The content of these files can be quickly loaded on memory and transformed into R-specific data structures, such as data frames. So, the Opal performance problems, mentioned in Section 3.1.2.2, are effectively minimized in this architecture.

Establishing connections to others Opal instances is also possible, i.e., the researchers can access resources from different locations, keeping the privacy-preserving analyses constraints.

Endpoint	Type	HTTP method	Description
/v1	Service status	GET	Returns success if API service is running.
/v1/info	Service information	GET	Upon success, returns service information such as name, version, etc.
/v1/repertoire/:repertoire_id	Retrieve a repertoire given its repertoire_id	GET	Upon success, returns the Repertoire information in JSON according to the AIRR Data Model.
/v1/repertoire	Query repertoires	POST	Upon success, returns a list of Repertoires in JSON according to the AIRR Data Model.
/v1/rearrangement/:sequence_id	Retrieve a rearrangement given its sequence_id	GET	Upon success, returns the Rearrangement information in JSON or TSV format according to the AIRR Data Model.
/v1/rearrangement	Query rearrangements	POST	Upon success, returns a list of Rearrangements in JSON or TSV format according to the AIRR Data Model.

Figure 5.4: ADC API endpoints.

There are many DataSHIELD commands available[4]. Nevertheless, the usage of specific packages to deal with Omics analysis should be necessary. An example of the extensibility of DataSHIELD is the dsOmics package[5]. This package provides facilities, such as transcriptomic, epigenomic, and genomic data analyses. A detailed tutorial is available[6].

The usage of other packages for specific repertoires analyses, such as Immunarch[7] or Alakazam[8] depends, at first, on adapting them so that they work according to the DataSHIELD architecture and disclosure control rules. For instance, the "DataSHIELD-like" wrapper for Immunarch[9] [10] used in Section 3.1.2.2 does not have disclosure control rules and it was used only to support a proof of concept.

5.3.1 Evaluating the Resources to access AIRR-Rearrangement data

The AIRR Standards[11], maintained by AIRR Community, provides specifications about data representations, including Repertoire, Rearrangement, Alignment (experimental), Clonal and Lineage Tree (experimental) and Cell data (experimental). These specifications define how the ADC API[12] works. The ADC API provides a list of endpoints[12], as shown on Figure 5.4.

Some AIRR repositories are currently adhering to AIRR Data Commons (ADC), such as the iReceptor Turnkey. We have a test environment with iReceptor Turnkey and ADC API available. In the context of the iReceptor+ project, this API specifies how data generated by pipelines are organized and how they can be retrieved to perform analyses. As we can see on Figure 5.4, if we want to make queries in rearrangements data, we need to use a POST HTTP method in the endpoint `/v1/rearrangement`. Upon success, it will return a list of Rearrangements in JSON or TSV format according to the AIRR Data Model. Filters can also be applied. An example of this request can be viewed on Figure 5.5.

[4] https://cran.datashield.org/web/.
[5] https://github.com/isglobal-brge/dsOmics.
[6] https://htmlpreview.github.io/?https://github.com/isglobal-brge/dsOmicsClient/blob/master/vignettes/dsOmics.html.
[7] https://immunarch.com/.
[8] https://alakazam.readthedocs.io/en/stable/.
[9] https://gitlab.inesctec.pt/ireceptorplus/r-projects/dsimmunarch
[10] https://gitlab.inesctec.pt/ireceptorplus/r-projects/dsimmunarchclient.
[11] https://docs.airr-community.org/en/stable/index.html

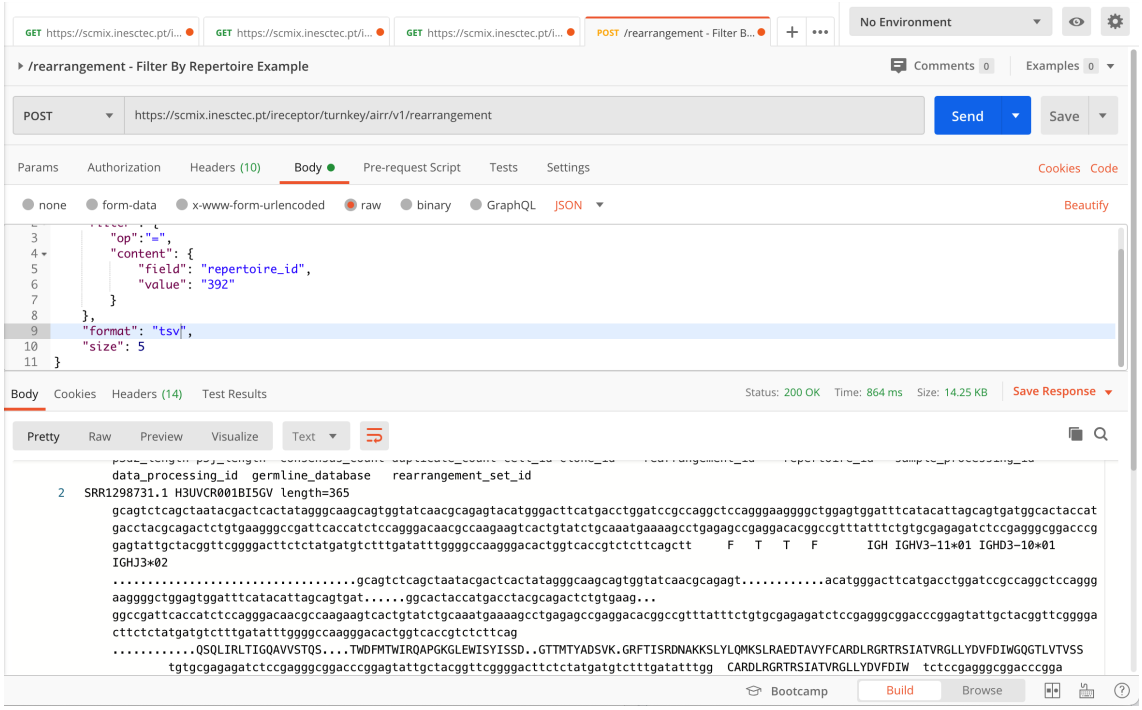

Figure 5.5: An example of HTTP request to the `v1/rearrangement` endpoint.

Resources also provide a mechanism to retrieve data files using HTTPS and TSV format (see an example on Figure 5.6) but, instead of using the POST method, Resources uses the GET method. Additionally, Resources are not able to receive filter information on the `Body Request`.

This is a typical situation where we need to design a custom resource connector. First, this connector should be initiated with the URL to the desired endpoint and credentials. Then the implementation of the connection will send a POST request with the appropriate body and headers and will handle the response content (the TSV data) to make it available for further analysis. Such a custom resource connector is not yet available.

5.3.2 Accessing the pipeline execution metadata

The pipeline execution metadata, in our simulated scenario, should contain the registers according to Appendix B.

If an Analyst wants to check this information, he/she can simply access the data registered on the pipeline execution database. By convention, the name of the resources can be the same as the data file. Then, this name can be used in a SELECT statement to retrieve all the information related to the pipeline that generate that file. The script below is a simple example of how to configure a PostgreSQL connection on RStudio:

```
1 # Install the latest RPostgres release from CRAN:
2 install.packages("RPostgres")
3
```

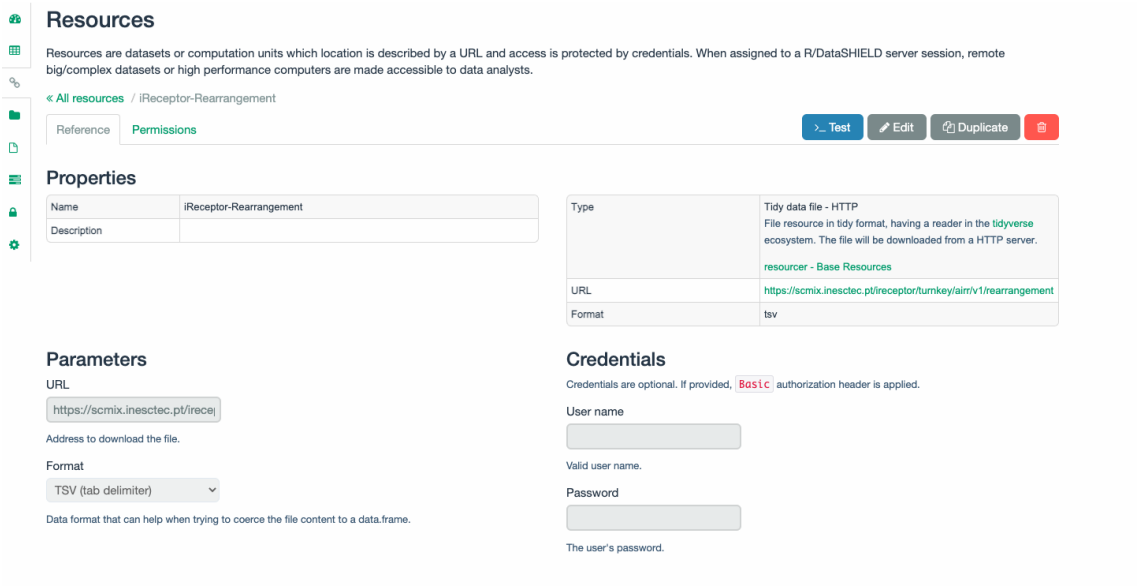

Figure 5.6: A Resource of the HTTP category.

```
# Connect to the default postgres database
con <- dbConnect(RPostgres::Postgres(),dbname = 'SEQUENCING_PIPELINE',
                 host = 'XXX.XXX.XXX.XXX',
                 port = 5432,
                 user = 'XXX',
                 password = 'XXX')

dbListTables(con)

dbListFields(con, "t_file")
dbReadTable(con, "t_file")

# You can fetch all results:
res <- dbSendQuery(con, "SELECT * FROM t_file")
dbFetch(res)
dbClearResult(res)
```

Listing 5.4: Example of how to access a PostgreSQL database on R.

Moreover, a database table (or a materialized view with custom queries) can also be registered as a resource, so all these data can be accessed through RStudio as a resource and transformed into R data frames.

The `step_order` field in PROCESSING STEP table, together with the `command_line` field in the COMMAND PROCESSING STEP table and other auxiliary tables, allow assembling a pipeline script easily, making the pipelines traceable.

Having the various processing steps and execution parameters documented in a machine-readable and relational model format, enables this kind of process to be more auditable, consistent, accessible, robust and reliable. These requirements, in the context of ADC API, are important,

since the endpoint `/v1/repertoire` (see Figure 5.4) shows part of this information for each repertoire. Nevertheless, as we mentioned, this model alone does not guarantee a full reproducible ecosystem.

5.4 Summary

This chapter presented an example of the implementation of the proposed solution using a basic pipeline example to generate target datasets to be used in future analyses. We also presented a way to access the stored data on the proposed database that deals with reproducibility.

Chapter 6

Conclusions and Future Work

Privacy-preserving analyses across federated data repositories is a paramount requirement in health research. Indeed, there are many recent articles whose central aspect involves this topic.

On the other hand, Omics data, even if they are related to the health area, have their own characteristics: they are large, poorly structured, and do not have evident labels that identify them. Making an analogy, while in computation, we only need two characters (0 and 1) to represent the entire computational ecosystem, we also need a few primary characters in Omics data to represent biological samples. Still, such data is also highly sensitive, and privacy concerns should also be applied. Not just, the analysis of this kind of data can show researchers a path to many scientific conclusions.

We started this work by defining our motivation and objectives. The requirements for our proposed solution were further described in Section 4.1. The requirements were met with the adoption of Coral Distribution, which includes the Opal DataSHIELD and Resources feature. The proposed relation model enhances the reproducibility but does not replace existing reproducible Bioinformatics pipeline tools. Indeed, further work is needed to evaluate the incorporation of this model into existing reproducible pipeline tools.

6.1 Results

We evaluated that the proposed solution (see Section 4.3) worked as expected in the scenario of privacy-preserving analyses for Omics data. Adjustments were necessary on the Apache Server provided by the Coral distribution to deal with the upload of large files.

The files referenced as resources are copied to a temporary folder on the R Server machine during the analyses, which can cause a legal problem if the Opal and R Server run in a location subordinate to different privacy regulations, when compared with the associated resource. Therefore, a workaround is to have a Coral environment (or an Opal/R Server) installed on a location where the data owner is also responsible and registers the resources in this installation.

The Resources feature is not fully compatible with the ADC API, but it is possible to develop a custom resource connector to directly enable access to AIRR-Rearrangement data.

The proposed reproducible relation model can afford the traceability of bioinformatics pipelines very well, but this model alone does not guarantee a full reproducible ecosystem, since it does not solve the platform isolation problem. It can only be guaranteed when combining reproducible tools that offer built-in support for containers, such as Nextflow or Snakemake, and a set of values and good practices. Some of these best practices were cited by Di Tommaso[3]:

- Publish your pipeline project from day one on a version control system, such as GitHub, to manage pipeline revisions;
- Create a small dataset to test your scripts quickly and include it as default data in your project;
- Use a Continuous Integration server (e.g., GitLab, Travis) to test any change timely;
- Isolate the pipeline tools using container technology, such as Docker. The support for container runtimes is a practical way to guarantee the execution of the stack in the same environment, also enabling the portability and facilitating scalability;
- Join a community to collect, improve and discuss pipelines[7]. A standout community can be seen in nf-core[1];
- Preferably, choose reproducible tools that are command-line oriented, in order to accommodate the migration of existing bioinformatics pipelines.

6.2 Future Work

Although we have analyzed several scenarios involving privacy-preserving analyses and pipelines reproducibility, we did not have time to setup and test all the desired functionalities.

Mainly, we are interested in adapting the basic pipeline presented in Section 5.3 to run across some reproducible tools. From the options listed in Section 3.2, Nextflow seemingly allows fast prototyping and steps declarations. It promotes a Domain Specific Language (DSL) with a declarative reactive programming approach based on functional composition. The parallelization steps are implicitly defined by inputs and outputs declarations.

The suggested reproducible database (see Appendix A) should be accessible through an API. We can use a framework, such as LoopBack[2], to build this API. We can extend Nextflow to fill the relational database while the pipeline is executed.

While we were evaluating the Resources to access AIRR-Rearrangement data (see Section 5.3.1), we noticed that it is necessary to design a custom resource connector. Accordingly to the main

[1] https://nf-co.re/
[2] https://loopback.io/

developer of Resources, it should be possible by implementing the POST request in plain R, using the `httr` package[3]. Then this code can be wrapped in a ResourceClient[4] subclass (from the resources package). Finally, an R package should be created with this code and deployed in the R server.

We didn't have time to perform analyses on the datasets from the presented example. The usage of other packages for specific repertoires analyses, such as Immunarch or Alakazam, depends, at first, on adapting them to work according to the DataSHIELD architecture and disclosure control rules. At first, not all types of analyses can be non-disclosed. It is necessary evaluate if existing analyses have the potential to expose privacy data. A good example of a specialized package adherent with DataSHIELD architecture is dsOmics[5].

Lastly, we also would like to evaluate the performance of a Rock Cluster[6], a recent and relevant development feature, and how it works with Resources. Unfortunately, its benefits remain theoretical, since we didn't have time to evaluate it.

There is a long way to go towards in direction to the incorporation of privacy-preserving analyses for Omics data. Though regulations and principles, such as GDPR, FAIR, and TRUST guide the development of solutions to deal with sensitive data at the same time promote mechanisms to improve scientific research, many challenges are remaining. Interoperability, Scalability, Reproducibility are examples of challenging requirements that rarely are addressed. In the context of bioinformatics pipelines, the raw data and the list of tools used in the workflow could not be enough to guarantee the reproducibility of the results. Indeed, different releases of the same tools or the system libraries might lead to sneaky reproducibility issues.

Additionally, the sharing of raw data should be avoided in the context of privacy-preserving. Existing reproducible tools allow users to create reproducible pipelines, but the flexibility of the metalanguage offered by these tools can make their utilization difficult for users without advanced programming skills. Novel approaches can appear in the following years, and we expect that the present work collaborates to seed new discussions.

[3] https://cran.r-project.org/web/packages/httr/vignettes/quickstart.html
[4] https://rdrr.io/github/obiba/resourcer/man/ResourceClient.html
[5] https://htmlpreview.github.io.
[6] https://rockdoc.obiba.org/en/latest/introduction.html#scalability

References

[1] O. J. Buske, M. Girdea, S. Dumitriu, B. Gallinger, T. Hartley, H. Trang, A. Misyura, T. Friedman, C. Beaulieu, W. P. Bone, A. E. Links, N. L. Washington, M. A. Haendel, P. N. Robinson, C. F. Boerkoel, D. Adams, W. A. Gahl, K. M. Boycott, and M. & Brudno. Phenomecentral: A portal for phenotypic and genotypic matchmaking of patients with rare genetic diseases. *Human Mutation*, 1(1), July 2015.

[2] B. D. Corrie, N. Marthandan, B. Zimonja, J. Jaglale, Y. Zhou, E. Barr, N. Knoetze, F. M. W. Breden, S. Christley, J. K. Scott, L. G. Cowell, and F. & Breden. ireceptor: A platform for querying and analyzing antibody/b-cell and t-cell receptor repertoire data across federated repositories. *Blackwell Publishing Ltd*, 284(1), June 2018.

[3] et al. Di Tommaso, Paolo. Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35(4), April 2017.

[4] H. V. Firth, S. M. Richards, A. P. Bevan, S. Clayton, M. Corpas, D. Rajan, S. Van Vooren, Y. Moreau, and N. P. Pettett, R. M. & Carter. Decipher: Database of chromosomal imbalance and phenotype in humans using ensembl resources. *American Journal of Human Genetics*, 1(1), April 2009.

[5] The Global Alliance for Genomics and Health. A federated ecosystem for sharing genomic, clinical data. *American Association for the Advancement of Science*, 352(1), June 2016.

[6] K. Imkeller, P. F. Arndt, H. Wardemann, and C. E. & Busse. scireptor: Analysis of single-cell level immunoglobulin repertoires. *BMC Bioinformatics*, 1(1), June 2016.

[7] N. Kulkarni, L. Alessandrì, and R. et al. Panero. Reproducible bioinformatics project: a community for reproducible bioinformatics analysis pipelines. *BMC Bioinformatics*, 19, October 2018.

[8] D. Lin, J. Crabtree, I. Dillo, R. R. Downs, R. Edmunds, D. Giaretta, M. De Giusti, H. L'hours, W. Hugo, R. Jenkyns, V. Khodiyar, and M. E. et al Martone. The tRUSt principles for digital repositories. *Scientific Data*, 1(1), May 2020.

[9] B. B. Misra, C. Langefeld, M. Olivier, and L. A. & Cox. Integrated omics: tools, advances and future approaches. *Journal of Molecular Endocrinology*, 62(1), January 2019.

[10] F. Mölder, K.P. Jablonski, B. Letcher, and et al. Sustainable data analysis with Snakemake. *F1000Research*, 1(2), April 2021.

[11] V. I. Nazarov, M. V. Pogorelyy, E. A. Komech, I. V. Zvyagin, D. A. Bolotin, M. Shugay, D. M. Chudakov, Y. B. Lebedev, and I. Z. Mamedov. tcR: an R package for T cell receptor repertoire advanced data analysis. *BMC Bioinformatics*, 16(1), May 2015.

[12] Tuan Pham, Jason J Jung, Ernesto Satoshi Nakayasu, Scott Christley, Ademar Aguiar, George Blanck, Felix Breden, Syed Ahmad, Chan Bukhari, Christian E Busse, Jerome Jaglale, Srilakshmy L Harikrishnan, Uri Laserson, Bjoern Peters, Artur Rocha, Chaim A Schramm, Sarah Taylor, Jason Anthony, Vander Heiden, Bojan Zimonja, Corey T Watson, Brian Corrie, and Lindsay G Cowell. Article 22 LG (2020) The ADC API: A Web API for the Programmatic Query of the AIRR Data Commons. *Front. Big Data*, 3:22, June 2020.

[13] A. A. Philippakis, D. R. Azzariti, S. Beltran, A. J. Brookes, C. A. Brownstein, M. Brudno, H. G. Brunner, O. J. Buske, K. Carey, C. Doll, S. Dumitriu, S. O. M. Dyke, J. T. den Dunnen, H. V. Firth, R. A. Gibbs, M. Girdea, M. A. Gonzalez, M.and Haendel, A. Hamosh, and H. L. Rehm. The matchmaker exchange: A platform for rare disease gene discovery. *Human Mutation*, 352(1), August 2015.

[14] Somak Roy, Christopher Coldren, Arivarasan Karunamurthy, Nefize S Kip, Eric W Klee, Stephen E Lincoln, Annette Leon, Mrudula Pullambhatla, and et al. Robyn Temple-Smolkin. Standards and Guidelines for Validating Next-Generation Sequencing Bioinformatics Pipelines. *Monica The Journal of Molecular Diagnostics*, 20(1), January 2018.

[15] J. M. M. Rumbold and B. Pierscionek. The effect of the general data protection regulation on medical research. *Journal of Medical Internet Research*, 19(2), Feb 2017.

[16] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.W. Boiten, L.B. da Silva Santos, P.E. Bourne, and J. Bouwman. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1):160018, March 2016.

[17] R.C. Wilson, O.W. Butters, D. Avraam, J. Baker, J.A. Tedds, A. Turner, M. Murtagh, and P.R. Burton. DataSHIELD – New Directions and Dimensions. *Data Science Journal*, 16, April 2017.

[18] Marcon Y., Bishop T., Avraam D., Escriba-Montagut X., Ryser-Welch P., and et al. Wheater S. Orchestrating privacy-protected big data analyses of data from different resources with R and DataSHIELD. *PLoS Comput Biol*, 17, March 2021.

[19] Maryam B Yassai, Yuri N Naumov, Elena N Naumova, and Jack Gorski. A clonotype nomenclature for T cell receptors. *Immunogenetics*, 1(1), July 2009.

Appendix A

Reproducibility: Relational model

This is a simplified version of a relational database model that includes entities related to the Bioinformatics domain and the Pipeline execution process domain.

Some entities were inspired in the AIRR Data Model[1], from AIRR Community. Still, it does not represent a definitive model, since we need to improve our domain-specific knowledge of Bioinformatics.

[1] https://docs.airr-community.org/en/latest/datarep/overview.html#relationship-between-schema-objects.

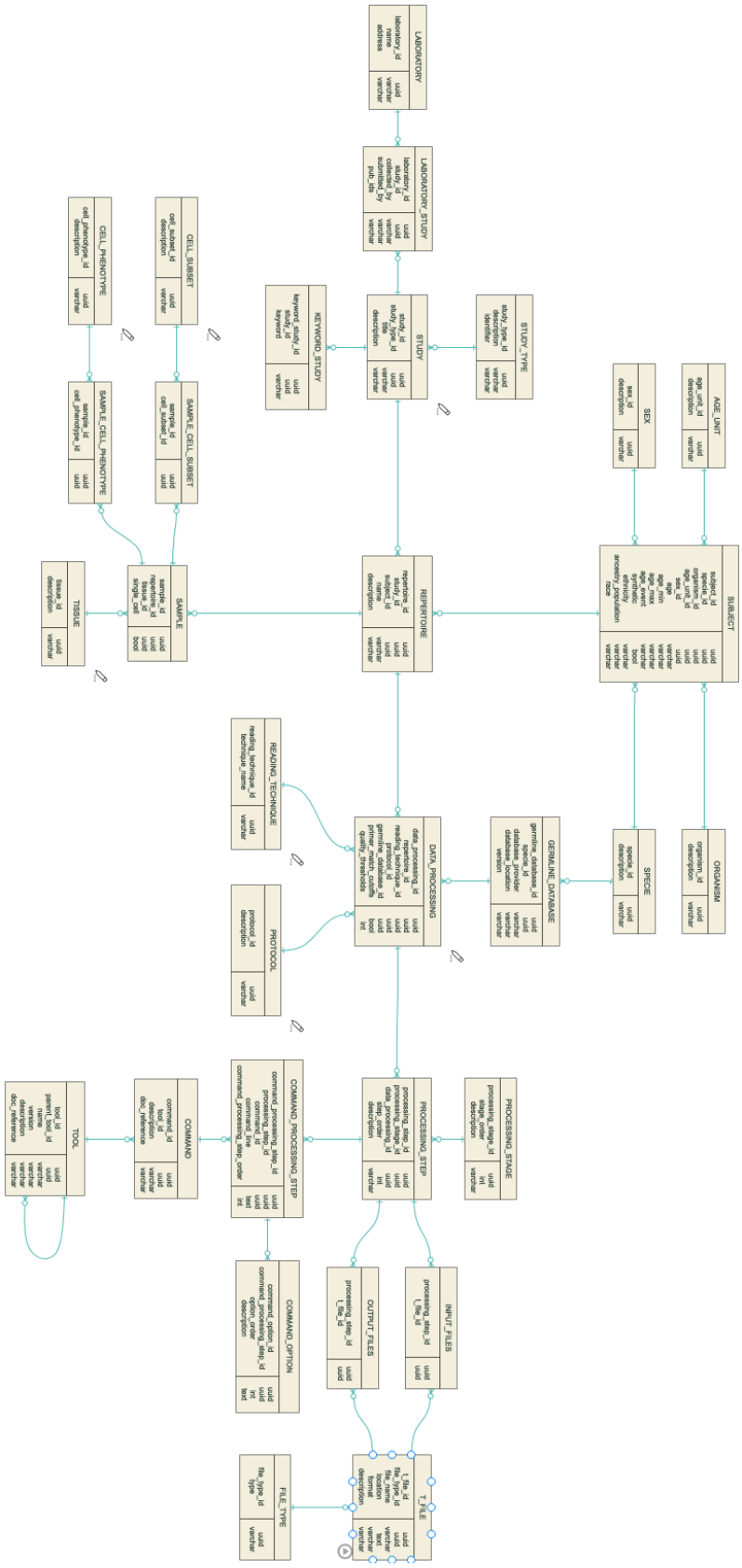

Figure A.1: Reproducibility - Relational model.

Appendix B

Registers generated in a simulated pipeline execution

This is the result of a simulated tracking execution of the pipeline presented on Section 5.3. In an ideal scenario, the filling process of this tracking execution should be automated, i.e., without human intervention.

The proposed relation model is a complementary approach to enhance the reproducibility and traceability of pipelines and does not replace all functions of reproducible bioinformatics pipeline tools.

Table B.1: Data Processing Registers.

DATA_PROCESSING		
data_processing_id	repertoire_id	...
1	...	...

Table B.2: Processing Step Registers.

PROCESSING_STEP			
processing_step_id	data_processing_id	description	step_order
1	1	Process Alignments	1
2	1	Assemble Data	2
3	1	Export Assembled Data	3
4	1	Export Clonotypes	4

Table B.3: File Type Registers.

FILE_TYPE	
file_type_id	type
1	Raw Sequences
2	Alignment Data
3	Assembled Data
4	Rearrangement Data
5	Clonotype Data
6	Germline reference Data

Table B.4: File Registers.

T_FILE		
t_file_id	file_type_id	file_name
1	1	SRR1033675.fasta
2	2	SRR1033675_mixcr_redo.vdjca
3	6	imgt.2018025.sv2.json
4	3	SRR1033675_mixcr_redo.clns
5	4	SRR1033675_mixcr_redo_annotations.txt
6	5	SRR1033675_mixcr_redo_clones.txt

Table B.5: Input Files Registers.

INPUT_FILES	
processing_step_id	t_file_id
1	1
1	3
2	2
3	2
4	4

Table B.6: Output Files Registers.

OUTPUT_FILES	
processing_step_id	t_file_id
1	2
2	3
3	4
4	5

Table B.7: Tool Registers.

TOOL		
tool_id	name	version
1	MiXCR	3.0.13

Table B.8: Command Registers.

COMMAND		
command_id	tool_id	description
1	1	align
2	1	assemble
3	1	exportAlignments
4	1	exportClones

Table B.9: Command Processing Step Registers.

COMMAND_PROCESSING_STEP				
command_ processing_ step_id	processing_ step_id	command_id	command_line	step_order
1	1	1	<entire command line>	1
2	2	2	...	2
3	3	3	...	3
4	4	4	...	4

Table B.10: Command Option Registers.

COMMAND_OPTION			
command_ option_id	command_ processing_ step_id	option_order	description
1	1	1	-Xmx16g
2	1	2	–verbose
3	1	3	-f
4	1	4	–library img.201802-5
5	1	5	-s hsa
...	...	...	...

Appendix C

Docker stack for the database pipeline

To run this Docker stack, it is necessary to configure Docker and Docker-Compose[1]. It is composed by two services: a PostgreSQL database and an administrator client (adminer). After the stack is installed, the administrator client can be accessed on port 8080.

docker-compose.yml

```
version: '3.7'

services:
  db:
    image: postgres
    restart: always
    environment:
      - POSTGRES_USER=admin
      - POSTGRES_PASSWORD=admin_pwd
      - POSTGRES_DB=SEQUENCING_PIPELINE
    ports:
      - 5432:5432
    volumes:
      - db_data:/var/lib/postgresql/data
      - ./ddl.sql:/docker-entrypoint-initdb.d/ddl.sql
    networks:
      - net

  adminer:
    image: adminer
    restart: always
    ports:
      - 8080:8080
    networks:
      - net

volumes:
```

[1] https://docs.docker.com/compose/.

```
  db_data:
    labels:
      system: "SEQUENCING_PIPELINE"

networks:
  net:
    name: db_net
    driver: overlay
    attachable: true
```

Listing C.1: Docker-Compose to deploy the suggested database model.

In the same folder, it is necessary to create an SQL file like below:

ddl.sql

```


CREATE TABLE FILE_TYPE (
  file_type_id serial PRIMARY KEY,
  type VARCHAR ( 255 ) UNIQUE NOT NULL
);

CREATE TABLE T_FILE (
  t_file_id serial PRIMARY KEY,
    file_type_id INT NOT NULL,
    file_name VARCHAR ( 255 ) NOT NULL,
    location TEXT NULL,
    format VARCHAR ( 255 ) NOT NULL,
    description VARCHAR ( 255 ) NOT NULL,
  FOREIGN KEY (file_type_id) REFERENCES FILE_TYPE (file_type_id)
);

CREATE TABLE PROCESSING_STAGE (
  processing_stage_id serial PRIMARY KEY,
  stage_order INT NULL,
    description VARCHAR ( 255 ) NULL
);


CREATE TABLE TOOL (
  tool_id serial PRIMARY KEY,
    parent_tool_id INT NULL,
    name VARCHAR ( 255 ) UNIQUE NOT NULL,
    version VARCHAR ( 255 ),
    description VARCHAR ( 255 ),
  doc_reference VARCHAR ( 255 ),
    FOREIGN KEY (parent_tool_id) REFERENCES TOOL (tool_id)
```

```
);

CREATE TABLE COMMAND (
  command_id serial PRIMARY KEY,
    tool_id INT NOT NULL,
    description VARCHAR ( 255 ),
  doc_reference VARCHAR ( 255 ),
    FOREIGN KEY (tool_id) REFERENCES TOOL (tool_id)
);


CREATE TABLE READING_TECHNIQUE (
  reading_technique_id serial PRIMARY KEY,
    technique_name VARCHAR ( 255 ) NOT NULL
);

CREATE TABLE PROTOCOL (
  protocol_id serial PRIMARY KEY,
    description VARCHAR ( 255 ) NOT NULL
);

CREATE TABLE LABORATORY (
  laboratory_id serial PRIMARY KEY,
    name VARCHAR ( 255 ) NOT NULL,
    address VARCHAR ( 255 ) NOT NULL
);

CREATE TABLE STUDY_TYPE (
  study_type_id serial PRIMARY KEY,
    description VARCHAR ( 255 ) NOT NULL,
    identifier VARCHAR ( 255 ) NULL
);

CREATE TABLE STUDY (
    study_id serial PRIMARY KEY,
  study_type_id INT NOT NULL,
    title VARCHAR ( 255 ) NOT NULL,
    description VARCHAR ( 255 ) NOT NULL,
    FOREIGN KEY (study_type_id) REFERENCES STUDY_TYPE (study_type_id)
);

CREATE TABLE LABORATORY_STUDY (
    laboratory_id INT NOT NULL,
    study_id INT NOT NULL,
    collected_by VARCHAR ( 255 ) NULL,
    submitted_by VARCHAR ( 255 ) NULL,
    pub_ids VARCHAR ( 255 ) NULL,
    PRIMARY KEY (laboratory_id, study_id),
    FOREIGN KEY (laboratory_id) REFERENCES LABORATORY (laboratory_id),
```

```
    FOREIGN KEY (study_id) REFERENCES STUDY (study_id)
);

CREATE TABLE KEYWORD_STUDY (
    keyword_study_id serial PRIMARY KEY,
    study_id INT NOT NULL,
    keyword VARCHAR ( 255 ) NOT NULL,
    FOREIGN KEY (study_id) REFERENCES STUDY (study_id)
);

CREATE TABLE ORGANISM (
  organism_id serial PRIMARY KEY,
    description VARCHAR ( 255 ) NOT NULL
);

CREATE TABLE AGE_UNIT (
  age_unit_id serial PRIMARY KEY,
    description VARCHAR ( 255 ) NOT NULL
);

CREATE TABLE SEX (
  sex_id serial PRIMARY KEY,
    description VARCHAR ( 255 ) NOT NULL
);

CREATE TABLE SPECIE (
  specie_id serial PRIMARY KEY,
    description VARCHAR ( 255 ) NOT NULL
);

CREATE TABLE SUBJECT (
  subject_id serial PRIMARY KEY,
    specie_id INT NOT NULL,
    organism_id INT NOT NULL,
    age_unit_id INT NOT NULL,
    sex_id INT NOT NULL,
    age VARCHAR ( 255 ) NULL,
    age_min VARCHAR ( 255 ) NULL,
    age_max VARCHAR ( 255 ) NULL,
    age_event VARCHAR ( 255 ) NULL,
    synthetic BOOLEAN NULL,
    ethnicity VARCHAR ( 255 ) NULL,
    ancestry_population VARCHAR ( 255 ) NULL,
    race VARCHAR ( 255 ) NULL,
    FOREIGN KEY (specie_id) REFERENCES SPECIE (specie_id),
    FOREIGN KEY (organism_id) REFERENCES ORGANISM (organism_id),
    FOREIGN KEY (age_unit_id) REFERENCES AGE_UNIT (age_unit_id),
    FOREIGN KEY (sex_id) REFERENCES SEX (sex_id)
);
```

```
131
132 CREATE TABLE GERMLINE_DATABASE (
133   germline_database_id serial PRIMARY KEY,
134     specie_id INT NOT NULL,
135     database_provider VARCHAR ( 255 ) NOT NULL,
136   database_location VARCHAR ( 255 ) NOT NULL,
137     version VARCHAR ( 255 ) NULL,
138     FOREIGN KEY (specie_id) REFERENCES SPECIE (specie_id)
139 );
140
141 CREATE TABLE REPERTOIRE (
142   repertoire_id serial PRIMARY KEY,
143     study_id INT NOT NULL,
144     subject_id INT NOT NULL,
145     name VARCHAR ( 255 ) NOT NULL,
146     description VARCHAR ( 255 ) NULL,
147     FOREIGN KEY (study_id) REFERENCES STUDY (study_id),
148     FOREIGN KEY (subject_id) REFERENCES SUBJECT (subject_id)
149 );
150
151 CREATE TABLE TISSUE (
152   tissue_id serial PRIMARY KEY,
153     description VARCHAR ( 255 ) NOT NULL
154 );
155
156 CREATE TABLE SAMPLE (
157     sample_id serial PRIMARY KEY,
158   repertoire_id INT NOT NULL,
159     tissue_id INT NOT NULL,
160     single_cell BOOLEAN NULL,
161     FOREIGN KEY (repertoire_id) REFERENCES REPERTOIRE (repertoire_id),
162     FOREIGN KEY (tissue_id) REFERENCES TISSUE (tissue_id)
163 );
164
165 CREATE TABLE CELL_SUBSET (
166     cell_subset_id serial PRIMARY KEY,
167   description VARCHAR ( 255 ) NOT NULL
168 );
169
170 CREATE TABLE CELL_PHENOTYPE (
171     cell_phenotype_id serial PRIMARY KEY,
172   description VARCHAR ( 255 ) NOT NULL
173 );
174
175 CREATE TABLE SAMPLE_CELL_SUBSET (
176     sample_id INT NOT NULL,
177     cell_subset_id INT NOT NULL,
178   PRIMARY KEY (sample_id, cell_subset_id),
179     FOREIGN KEY (sample_id) REFERENCES SAMPLE (sample_id),
```

```
180     FOREIGN KEY (cell_subset_id) REFERENCES CELL_SUBSET (cell_subset_id)
181 );
182
183 CREATE TABLE SAMPLE_CELL_PHENOTYPE (
184     sample_id INT NOT NULL,
185     cell_phenotype_id INT NOT NULL,
186   PRIMARY KEY (sample_id, cell_phenotype_id),
187     FOREIGN KEY (sample_id) REFERENCES SAMPLE (sample_id),
188     FOREIGN KEY (cell_phenotype_id) REFERENCES CELL_PHENOTYPE (cell_phenotype_id)
189 );
190
191 CREATE TABLE DATA_PROCESSING (
192   data_processing_id serial PRIMARY KEY,
193     repertoire_id INT NOT NULL,
194     reading_technique_id INT NULL,
195     protocol_id INT NULL,
196     germline_database_id INT NULL,
197     primer_match_cutoffs BOOLEAN NULL,
198     quality_thresholds INT NULL,
199     FOREIGN KEY (repertoire_id) REFERENCES REPERTOIRE (repertoire_id),
200     FOREIGN KEY (reading_technique_id) REFERENCES READING_TECHNIQUE (
            reading_technique_id),
201     FOREIGN KEY (protocol_id) REFERENCES PROTOCOL (protocol_id),
202     FOREIGN KEY (germline_database_id) REFERENCES GERMLINE_DATABASE (
            germline_database_id)
203 );
204
205
206 CREATE TABLE PROCESSING_STEP (
207   processing_step_id serial PRIMARY KEY,
208     processing_stage_id INT NULL,
209     data_processing_id INT NOT NULL,
210   step_order INT NULL,
211     description VARCHAR ( 255 ) NULL,
212     FOREIGN KEY (processing_stage_id) REFERENCES PROCESSING_STAGE (
            processing_stage_id),
213     FOREIGN KEY (data_processing_id) REFERENCES DATA_PROCESSING (data_processing_id
            )
214 );
215
216 CREATE TABLE COMMAND_PROCESSING_STEP (
217   command_processing_step_id serial PRIMARY KEY,
218     command_id INT NOT NULL,
219     processing_step_id INT NOT NULL,
220     command_line TEXT NULL,
221     command_processing_step_order INT NULL,
222     FOREIGN KEY (command_id) REFERENCES COMMAND (command_id),
223     FOREIGN KEY (processing_step_id) REFERENCES PROCESSING_STEP (processing_step_id
            )
```

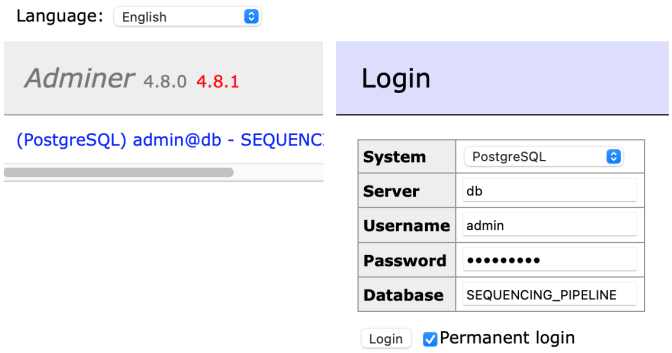

Figure C.1: Adminer - Login page.

```
224 );
225
226 CREATE TABLE COMMAND_OPTION (
227   command_option_id serial PRIMARY KEY,
228     command_processing_step_id INT NOT NULL,
229     option_order INT NOT NULL,
230     description TEXT NULL,
231     FOREIGN KEY (command_processing_step_id) REFERENCES COMMAND_PROCESSING_STEP (
          command_processing_step_id)
232 );
233
234 CREATE TABLE INPUT_FILES (
235     processing_step_id INT NOT NULL,
236     t_file_id INT NOT NULL,
237   PRIMARY KEY (processing_step_id, t_file_id),
238     FOREIGN KEY (processing_step_id) REFERENCES PROCESSING_STEP (processing_step_id
          ),
239     FOREIGN KEY (t_file_id) REFERENCES T_FILE (t_file_id)
240 );
241
242 CREATE TABLE OUTPUT_FILES (
243     processing_step_id INT NOT NULL,
244     t_file_id INT NOT NULL,
245   PRIMARY KEY (processing_step_id, t_file_id),
246     FOREIGN KEY (processing_step_id) REFERENCES PROCESSING_STEP (processing_step_id
          ),
247     FOREIGN KEY (t_file_id) REFERENCES T_FILE (t_file_id)
248 );
```

Listing C.2: DDL script to create the database.

Once the stack is running, it can be accessed using any database client. We can see on Figure C.1 how to do this on Adminer.

In our test, the server name is "db", the username is "admin", and the password is "admin_pwd", the same values defined on the db service in the *docker-compose.yml*.