

UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum



Nurlan Kerimov

**Designing a robust and portable workflow for
detecting genetic variants associated with
molecular phenotypes across multiple studies**

Master's Thesis (30 EAP)

Supervisor(s): Kaur Alasoo, PhD

Tartu 2019

Title: Designing a robust and portable workflow for detecting genetic variants associated with molecular phenotypes across multiple studies

Abstract:

Quantitative trait locus (QTL) analysis links variations in molecular phenotype expression levels to genotype variation. This analysis has become a standard practice to better understand molecular mechanisms underlying complex traits and diseases. Typical QTL analysis consists of multiple steps. Although a diverse set of tools is available to perform these individual analysis, the tools have so far not been integrated into a reproducible and scalable workflow that is easy to use across a wide range computational environments. Our analysis workflow consists of three modules. The analysis starts with quantification of the phenotype of interest, proceeds with normalisation and quality control and finishes with the QTL analysis. For phenotype quantification and QTL mapping modules we developed pipelines following best practices of the nf-core framework. The pipelines are containerized, open-source, extensible and eligible to be parallelly executed in a variety computational environments. For quality control module we developed a script which automatically computes the measures of quality and provides user with information. As a proof of concept, we uniformly processed more than 40 context specific groups from more than 15 studies and discovered at least one significant eQTL for more than 9000 genes. We believe that adopting our pipelines will increase reproducibility, portability and robustness of QTL analysis in comparison to existing approaches.

Keywords: pipeline; workflow; bioinformatics tools; workflow framework; QTL mapping; containerized pipeline

CERCS: B110 Bioinformatics, medical informatics, biomathematics, biometrics

Pealkiri eesti keeles Töökindla ja teisedatava töövoog väljatöötamine molekulaarsete tunnustega seotud geneetiliste variantide tuvastamiseks mitmetest andmestikest

Lühikokkuvõte:

Kvantitatiivse tunnuse lookusteks (*quantitative trait locus*, QTL) nimetatakse geneetilisi variante, millel on statistiline seos mõne molekulaarse tunnusega. QTL analüüs võimaldab paremini aru saada komplekshaiguseid ja tunnuseid mõjutavatest molekulaarsetest mehhanismidest. Tüüpiline QTL analüüs koosneb suurest hulgast sammudest, mille kõigi jaoks on olemas palju erinevaid tööriistu, kuid mida ei ole siiani kokku pandud ühte lihtsasti kasutatavasse, teisedatavasse ning korratavasse töövoogu. Käesolevas töös loodud töövoog koosneb kolmest moodulist: huvipakkuva tunnuse kvantifitseerimine (i), andmete normaliseerimine ja kvaliteedikontroll (ii) ning QTL analüüs (iii). Kvantifitseerimise ja QTL analüüsi moodulite jaoks kasutasime Nextflow töövoog juhtimise süsteemi ning järgisime kõiki nf-core raamistiku parimaid praktikaid. Mõlemad töövoog moodulid on avatud lähekoodiga ning kasutavad tarkvarakonteinereid, mis võimaldab kasutajatel neid lihtsalt laiendada ning jookstada erinevates arvutuskeskkondades. Kvaliteedikontrolli teostamiseks ning andmete normaliseerimiseks arendasime välja skripti, mis automaatselt arvutab välja erinevad kvaliteedimõõdikud ning esitab need kasutajale. Juhtprojekti raames viisime läbi geeniekspressiooni QTL analüüsi 15 andmestikus ja 40 erinevas bioloogilises kontekstis ning tuvastasime vähemalt ühe statistiliselt olulise QTLi enam kui 9000 geenile. Loodud töövoogude laialdasem kasutuselevõtt võimaldab muuta QTL analüüsi korratavamaks, teisedatavamaks ning lihtsamini kasutatavaks.

Võtmesõnad: töövoog, QTL analüüs, töövoog raamistik, konteinerdamine

CERCS: B110 Bioinformaatika, meditsiiniinformaatika, biomatemaatika, biomeetrika

Table of Contents

1. Introduction.....	5
1.1. Terms and Notions	5
1.2. Biological Background	6
1.2.1. Central Dogma of Molecular Biology	6
1.2.2. Quantifying RNA-seq Transcription.....	8
1.2.3. Genetic variation.....	9
1.2.4. Genome-Wide Association Studies (GWASs)	10
1.2.5. Quantitative Trait Locus (QTL) Mapping	10
1.2.6. Colocalisation	12
1.3. Project Overview	13
2. Bioinformatics Pipelines.....	15
2.1. Properties of Pipelines	15
2.1.1. Reproducibility (Replicability).....	15
2.1.2. Portability.....	17
2.1.3. Scalability	17
2.1.4. Dependency Isolation.....	18
2.1.5. Parallelisation.....	20
2.1.6. Reusability	21
2.2. Comparison of Pipeline Frameworks and Design Decisions.....	22
2.2.1. Classification of Pipelines.....	22
2.3. Pipeline Requirements of the Project.....	25
2.3.1. Comparison of Most Popular Frameworks	26
2.3.2. Pipeline Design Decisions	28
3. Quantification Pipeline	28
3.1. Quantification Methods	29
3.1.1. Gene Expression Quantification	30
3.1.2. Transcript Usage Quantification	31
3.1.3. Exon Expression Quantification	31
3.1.4. Alternative Splicing Usage Quantification	32

3.2. Pipeline overview.....	32
4. Quality Control and Normalisation.....	34
4.1. Principal Component Analysis (PCA).....	35
4.2. Multidimensional Scaling (MDS).....	37
4.3. Sex-specific Gene Expression Analysis.....	39
4.4. Sequence-genotype Matching (MBV analysis)	40
5. QTL Mapping Pipeline	43
5.1. Description of QTL Mapping Process	44
5.2. Pipeline Overview.....	44
5.3. Pipeline Implementation Details.....	47
5.3.1. Containerisation and Conda support.....	47
5.4. Pipeline Input and Output Description	48
5.4.1. Input Preparation.....	48
5.4.2. Description and Interpretation of Pipeline Output.....	50
6. Conclusions.....	50
7. References.....	51
8. Appendices.....	61
9. License	63

1. Introduction

1.1. Terms and Notions

Deoxyribonucleic acid (DNA): A molecule carrying genetic information and composed of four nucleotides (cytosine [C], guanine [G], adenine [A] or thymine [T])

Ribonucleic acid (RNA): A molecule primarily created on basis of DNA code and also composed of four nucleotides (cytosine [C], guanine [G], adenine [A] or uracil [U]). RNA has various biological roles in a cell.

Nucleotide: organic molecules that serve as building units of DNA and RNA

Genomic variant: a difference in a specific position between genotypes of two organisms belonging to the same species

Single Nucleotide Polymorphism (SNP): genomic variant which occurs as a substitution of single nucleotide

Gene: a sequence of nucleotides in DNA or RNA that codes for a molecule that has a function.

Transcript: one of the possible versions of the gene sequence.

Genomic feature: a genomic region with some annotated function. (e.g. gene, transcript, exon)

Expression: Abundance of the specific genomic feature in a specified biological environment

Sample: Genetic material obtained from a specific source (e.g. human, tissue)

Phenotype: The set of observable characteristics of a sample (e.g. gene and exon expression levels)

Genotype: The genetic constitution of an individual organism.

Metadata: A set of data that describes and gives information about other data.

Computation node: A set of configured hardware in order to serve computational power

Computation cluster: A set of computation nodes

Job (Task): A computational activity to be executed in computation node and requiring pre-defined amount of computational power.

Executor (Job scheduling system): A software to orchestrate execution of tasks in the computation cluster.

Execution environment: An environment equipped with necessary software in order to execute tasks.

1.2. Biological Background

Bioinformatics is a multidisciplinary field which needs the expertise of biologists, software engineers and computer scientists (Brass, 2000). To make efficient tools for biologists, software engineers and computer scientists should understand the basics of the domain they are working in. Thus, knowledge of the central dogma of molecular biology, Genome Wide Association Studies, Quantitative Trait Loci mapping and colocalisation is essential.

1.2.1. Central Dogma of Molecular Biology

In molecular biology, molecular functions are mostly performed by proteins. These large and complex molecules are required for the structure, function and regulation of organism's tissues. The abundance or structure of the specific protein in human cells can provide information about a specific trait (e.g. disease) (Dermitzakis, 2008; Emilsson et al., 2008; Liu, Gershon, & Kelsoe, 2017). The idea of the central dogma describes the process of protein production from genetic code. This unidirectional process consists of two main steps: *transcription* and *translation*, and can be described as DNA \Rightarrow RNA \Rightarrow Protein (Figure 1.1).

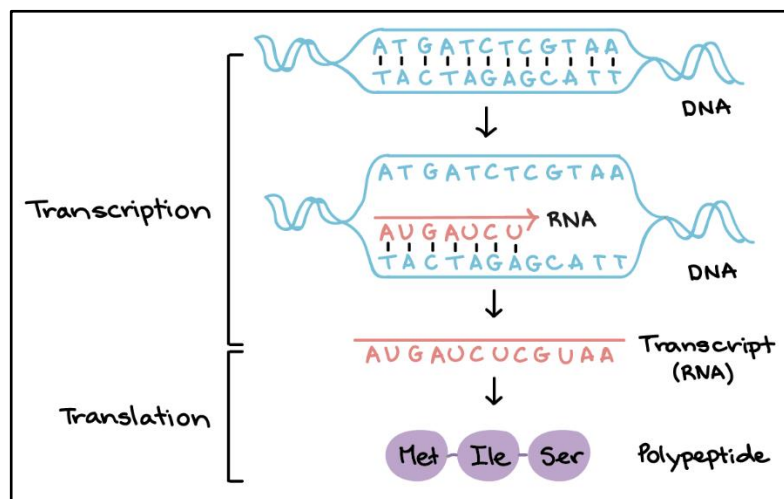


Figure 1.1. High level representation of central dogma of molecular biology.
Figure obtained from molecular biology curriculum of KhanAcademy¹.

In the human body, all cells have the same DNA, however the amount of transcribed RNA is different. The amount of transcribed RNA from a specific gene determines the gene expression level, which directly affects protein abundance produced from the same gene. Although each cell can express (activate, turn on) majority of the genes, some cell types can additionally express specific genes and repress (unexpress, turn off) others (Ramsköld, Wang, Burge, & Sandberg, 2009). The behaviour of expressing and repressing the genes is called gene regulation. Gene

¹<https://www.khanacademy.org/science/biology/gene-expression-central-dogma/transcription-of-dna-into-rna/a/overview-of-transcription>

regulation plays an important role in cell development, functionality and adaptation to the environment. Genes are regulated in different patterns, for example, muscle cells have to look and function differently from a brain cell or liver cell. Although we know that gene regulation is vital for life, this complicated process is not fully understood yet². Gene regulation most commonly occurs at the transcription level, however it can be effective in any step of gene expression.

Transcription is the process of producing messenger RNA (mRNA) from DNA. One strand of the DNA double helix acts as a template for the construction of a matching complementary RNA strand. A molecular machine called the *RNA polymerase* binds to a *promoter* region of a gene with the help of *transcription factors*, and starts to copy a sequence of the gene to an RNA molecule. In eukaryotic cells (e.g. human cells), the transcribed RNA molecule in this stage is considered “immature” RNA (a.k.a. pre-mRNA or primary transcript), and needs to go through modifications (RNA processing) to become a mature mRNA. These modifications are *splicing*, *capping* and *polyadenylation* (poly-A tail addition), and are the main events which designate the content of mature mRNA.

A pre-mRNA contains exons (regions in genes which will become a piece of mature mRNA) and introns (regions which do not survive the RNA processing modifications). In the splicing process, introns are removed and exons are joined in order to form mRNA from pre-mRNA. Splicing occurs in the nucleus of the cell, either during the transcription process or immediately after transcription is completed. Sometimes *alternative splicing* occurs, where splicing process creates different sequences of mRNA by varying the exon composition of pre-mRNA (J. Chen & Weiss, 2015; Y. Wang et al., 2015). When alternative splicing happens, different combinations of exon and intron usage result in production of a variety of proteins and other gene products (Figure 1.2). Alternative splicing is not a random process. It is regulated by regulatory protein molecules and often depends on genetic variants within or nearby the transcribed gene.

Other two steps of pre-mRNA processing are five-prime capping (5' capping) and three-prime (3') end tail polyadenylation. The 5' end of the transcribed gene is the side where the transcription starts and capping of this side happens shortly after the transcription is initiated. A special molecule is added to the 5' end to make mRNA more stable and mature to be able to undergo the translation process. Polyadenylation (poly(A)) on the other hand, is the addition of multiple adenine bases to specific (poly(A)) site in the 3' end of the newly transcribed pre-mRNA. This process is also vital for stability and translation of the mRNA, because the tail of the mRNA is shortened overtime and absence of poly(A) tail can result in degradation of the mRNA before reaching the translation. Usually, protein coding genes have multiple polyadenylation sites, so poly(A) tail can be added in any of them and change the content of mRNA. This phenomenon, called *alternative polyadenylation*, makes it possible to produce different mRNAs from one gene that are different in their 3' ends, eventually ending up with production of different proteins. Alternative promoter

² <https://ghr.nlm.nih.gov/primer/howgeneswork/geneonoff>

usage (choice), on the other hand, is another regulatory event, which defines the first exon of a gene to be transcribed (Figure 1.2) (Ayoubi & Van De Ven, 1996; Kimura et al., 2006).

Translation is the process where a mature mRNA is decoded in order to build a protein or a subunit of a protein. In eukaryotes (organisms whose cells have a nucleus), it happens outside of the nucleus by ribosomes with the help of transport RNA (tRNA). Although gene expression can also be regulated at the translation level (Wilkie, Dickson, & Gray, 2003), we will not discuss this further, because the RNA sequencing data used in this thesis measures mRNA abundance.

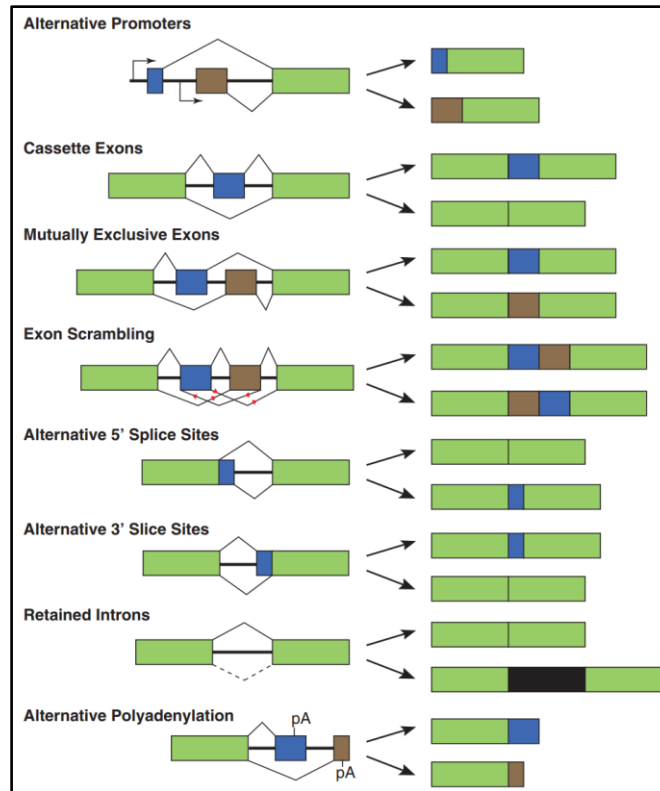


Figure 1.2. Symbolic representation of alternative splicing, alternative promoter and alternative polyadenylation events. Figure is obtained from (J. Chen & Weiss, 2015) study.

1.2.2. Quantifying RNA-seq Transcription

RNA sequencing (RNA-seq) data are essentially a collection of text strings representing nucleotide sequences. To produce this data, mature mRNA is extracted from the cell, fragmented, complementary DNA (cDNA) fragments made and then sequenced with a high-throughput sequencing machine (Marioni, Mason, Mane, Stephens, & Gilad, 2008) (Figure 1.3). High-throughput sequencing machines produce short sequences of basepairs, called sequencing reads. To extract interpretable information from the experiment, the reads need to be aligned to the reference sequence to quantify the relative counts of specific phenotypes. For instance, to quantify gene expression, the reads should be aligned to the reference genome sequence of the corresponding species (Figure 1.3).

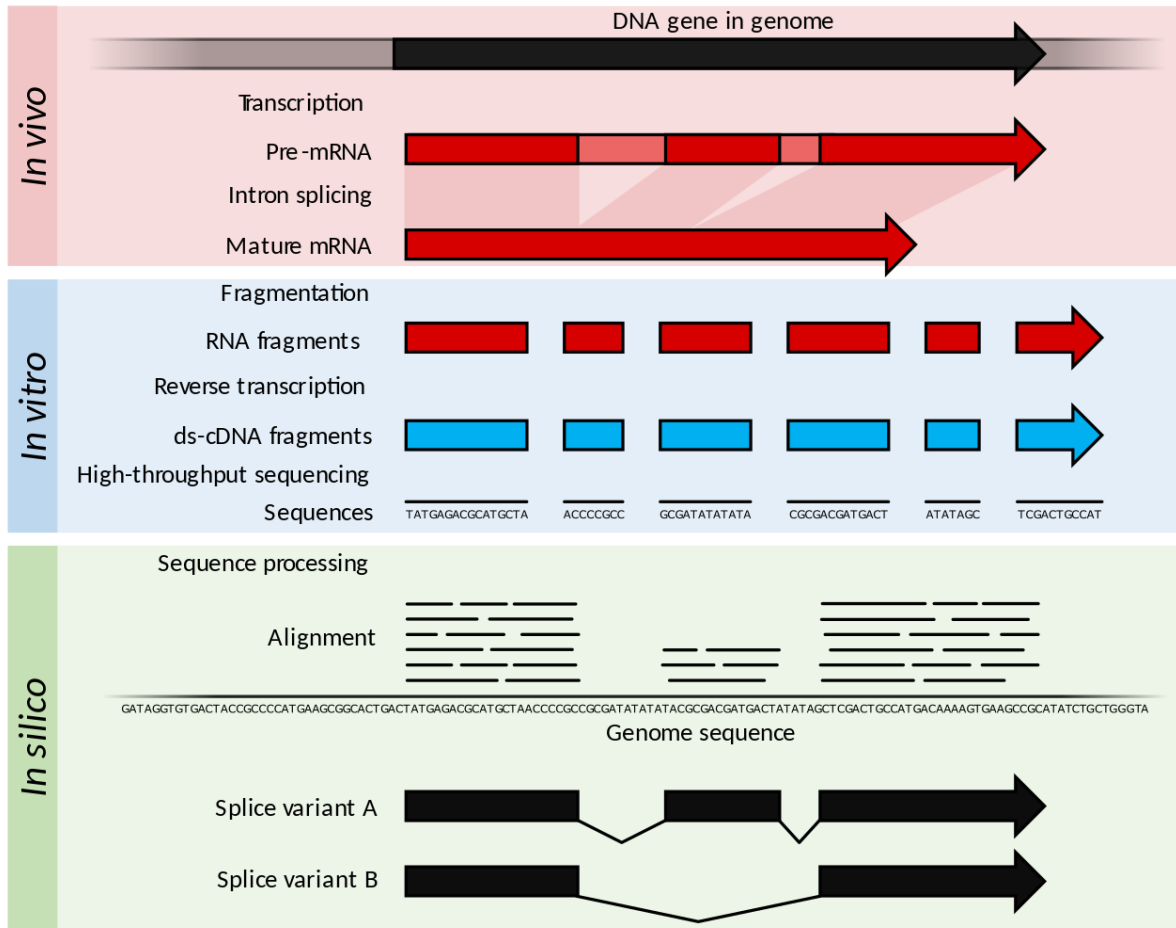


Figure 1.3. Visual summary of RNA-seq data sequencing and alignment. Figure obtained from RNA-seq wikipedia article³ and is available under CC BY 4.0 license⁴

Reads produced by the sequencing machines are usually stored in the FASTQ file format (Figure 1.4).

```
@HS40_15367:2:1101:10004:44007#21/1
CTGCTCCTTGGTGCGGATGTCAGGCCGAGCCACTGGCGGGACAGCTCCCGCAGCTGCCGAAAGCCTCCCGGGG
+
BB/BB<F/FFFFFFF<<<<FFFFFFFBF<FFFFFF<FFFFF/B<<FFFBFFF<FFFFFFB BBBFF<FFFFFFBF<
```

Figure 1.4. Example of a read in FASTQ file. First line: unique identifier, second line: sequence of bases, third line: optional unique ID repetition with description, fourth line: quality values for sequence of bases in line 2.

1.2.3. Genetic variation

The human genome is 3.1 billion nucleotides long and there are two copies of each chromosome. Any two individuals differ from each other at 0.5% of the loci, which means there are differences in more than 100 million locations (a.k.a. genetic variants). Each variant can have either reference

³ <https://en.wikipedia.org/wiki/RNA-Seq>

⁴ Thomas Shafee [CC BY 4.0 (<https://creativecommons.org/licenses/by/4.0>)]

or alternative allele inherited from one of the parents. Since humans have two sets of each chromosome (diploid, one inherited from the mother and another from the father), there are three possible value combinations of the inherited variant (1000 Genomes Project Consortium et al., 2015). For instance, in Table 1.1 variant with ID *chr1_1301656_T_C* has a reference allele of nucleotide “T” and alternative allele of nucleotide “C” in the reference genome. Hence, three possible inheritance combinations are “TT”, “TC” and “CC” (“CT” is the equivalent of “TC”). The sample (e.g. donor, person) indicated with ID “geno_275” has two reference alleles (e.g. “TT”, homozygous) inherited from the parents, and does not inherit alternative allele for this variant. The same person has one copy of both reference and alternative alleles for the variant *chr1_1302799_C_A* (e.g. “CA”, heterozygous). Genotypes can be measured using genotyping microarrays or whole genome sequencing. Genotype data is typically stored in variant call format (VCF, (Danecek et al., 2011)) files (Table 1.1).

Table 1.1. Visual representation of two variants in VCF file format.

CHR	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	geno_275
1	1301656	chr1_1301656_T_C	T	C	100	PASS	---	GT	0 0
1	1302799	chr1_1302799_C_A	C	A	100	PASS	---	GT	1 0

1.2.4. Genome-Wide Association Studies (GWASs)

The main goal of human genetics is to find genetic risk factors for the specific traits (e.g. diseases). There are a number of different tools, study designs and technologies to identify these risk factors. GWAS is an observational study of genetic variants in a specific cohort of individuals, which aims to find if any genetic variant is associated with a particular trait (e.g. disease). If a new genetic association is identified, researchers can contribute this finding to the pool of associations for future use by the community. This tremendous database helps researchers to come up with better strategies to detect, treat or prevent various diseases (Buniello et al., 2018). GWASs have been continuously grown over the past fifteen years into a great resource. While interpreting the complexity of human diseases is an essential objective, it is not the only target of human genetics. Pharmacology is a major beneficiary of GWAS. Pharmacogenetics studies associations of DNA sequence variations with drug metabolism and efficacy along with negative effects (Bush & Moore, 2012). This type of genetic studies have led to establishment of a new field called personalised medicine that aims to fit healthcare to individual patients based on their genetic information and other biological parameters.

1.2.5. Quantitative Trait Locus (QTL) Mapping

Another method for identifying associations between phenotypic data (traits) and genotypic data is quantitative trait locus (QTL) analysis. This statistical method attempts to clarify genetic bases of variation in complex traits. The phenotypic data in QTL analysis are quantitative traits which

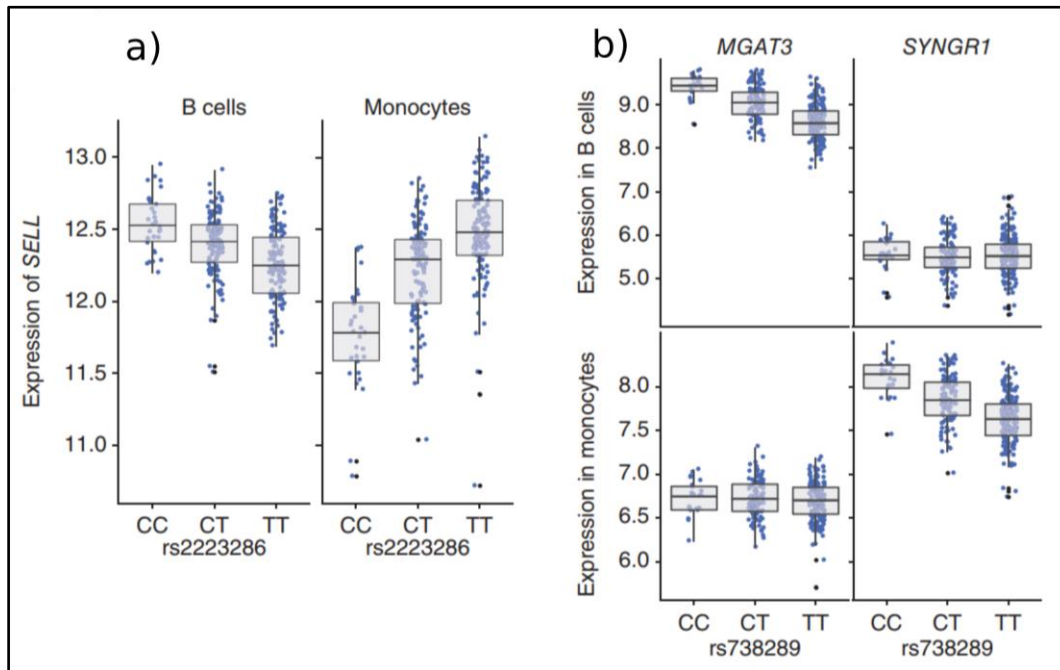


Figure 1.5. Cell type specific effect of the allele to the gene expression. a) T allele of rs2223286 variant is associated with decreased expression of *SELL* gene in B cells, but increased expression in Monocytes. b) T allele of rs738289 variant is associated with decreased expression of *MGAT3* gene in B cells but does not affect regulation in monocytes. Same allele does not affect regulation of *SYNGR1* gene in B cells but downregulates the expression of it in monocytes. Figure is obtained from (Fairfax et al., 2012).

can include aspects of morphology (e.g. weight, height); behavior (e.g. aggression, stress); physiology (e.g. blood pressure, oxygen saturation levels); as well as molecular phenotypes (e.g. gene expression levels, splicing events). If the abundance of a quantitative trait can be associated with a specific genetic variant in the genome, then this association can help to understand certain diseases associated with the quantitative trait. This process of associating quantitative traits with the genotype is called QTL mapping. Whereas phenotypes are represented by quantitative traits, genotype information is usually represented as molecular markers, such as single nucleotide polymorphisms (SNPs), polymorphic insertions or deletions (indels) or larger structural variants (Mackay, 2009). Another powerful feature of QTL studies is the ability to analyse associations in various contexts. Quantitative traits can be affected by several properties such as environment, sex, diet, cell type, experimental time point or any external stimulus. QTL mapping enables researchers to observe the effect of a specific SNP in a specific context which can be compared to the effect of the same SNP in another context. For example, the same allele can increase the expression of one gene in monocytes but might not affect the regulation of the same gene in B cells (Figure 1.5). The output of the QTL mapping process is called summary statistics, because it summarises the association information by containing p-values (probability of observing an association under the null hypothesis of there is no association), effect sizes and standard errors of

it⁵. Ultimately, summary statistics are compared to variants associated to disease in order to find if they are colocalised.

1.2.6. Colocalisation

QTL analysis is becoming increasingly popular in genetic research and they are an excellent complement to GWASs. GWASs have become a very powerful method to identify genetic variants associated with a complex disease. Nonetheless, most of the significant loci identified by GWASs are in the non-coding regions of the genome. This makes it challenging to understand the molecular mechanisms underlying these associations (Visscher, Brown, McCarthy, & Yang, 2012). On the other hand, QTL studies identify candidate SNPs associated with molecular traits, such as gene expression or splicing, but these associations cannot be easily related to the higher level organismal phenotypes (e.g. diseases). Several algorithms have been developed to colocalise information of eQTL SNPs and GWAS candidate SNPs to deduce the information between disease and gene (J. Wang, Zheng, Wang, Li, & Deng, 2019). The underlying idea is that if an allele is more common in disease carrier samples and at the same time this allele is found to be associated with the expression of particular genes, then it is likely that these genes influence the disease risk through changed expression (Figure 1.6).

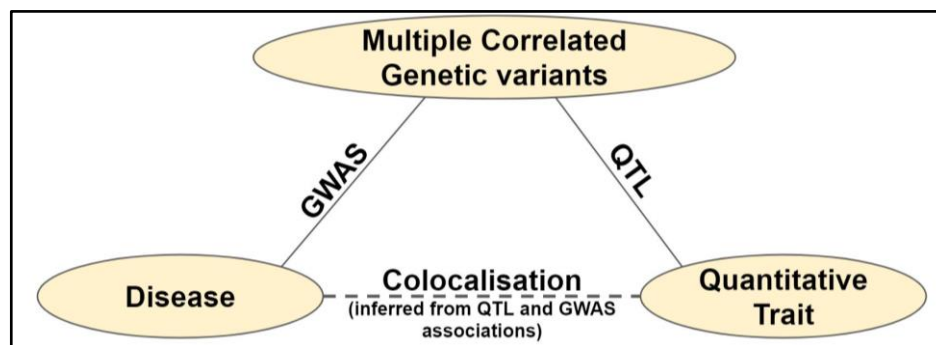


Figure 1.6. Association map between genetic variants, quantitative trait and disease. Colocalisation uses QTL and GWAS associations to associate a quantitative trait to a disease by finding probable causal variants.

Usually there is no single genetic variant in GWAS and QTL associations, but multiple correlated variants, simply because they are inherited together. This fact makes difficult to assess if the disease and quantitative trait are regulated by the same causal variant. Thus, Giambartolomei et al. (Giambartolomei et al., 2014) developed a statistical methodology to assess if the two associations are consistent (GWAS and QTL associations) with a shared causal variant. As a result, this method enables to infer associations between quantitative traits and disease by using GWAS and QTL summary statistics (Figure 1.6).

⁵ https://qtltools.github.io/qtltools/pages/mode_cis_nominal.html

1.3. Project Overview

The work in this thesis contributes to the eQTL Catalogue⁶, a collaborative project between the University of Tartu and the European Bioinformatics Institute. The aim of the project is to compile the largest catalogue of genetic variants associated with different transcriptional quantitative traits (gene expression, alternative splicing, transcript usage and exon expression) across tissues, cell types and cellular contexts. In the project, we aim to process quantitative trait expressions and genotype data from more than 23,000 biological samples across more than 18 distinct studies and 40 distinct biological contexts (“QTL groups”), all of which need to be processed separately for QTL mapping (Table 1.2 and Appendix 1). These groups are mostly formed of combinations of cell types and conditions (e.g. applied stimuli) of samples. For example, a study by Alasoo et al. (Alasoo et al., 2018) exposed human macrophages to Salmonella and IFN γ (interferon gamma) stimulations and detected eQTLs whose effect sizes changed after stimulation (response eQTLs). With the comparison of different QTL groups they were able to uncover novel molecular mechanisms concerning the response of immune cells to environmental stimuli. Ultimately, the catalogue allows researchers to query any disease-associated variant to identify associated target genes across a range of tissues, cell types and conditions, leading to better hypotheses about possible disease mechanisms.

Table 1.2: Subset of studies we processed with the pipeline. The QTL group is the combination of cell type and condition for each study. For example, BLUEPRINT study has 3 QTL groups and Alasoo_2018 study has 4. Studies: Alasoo_2018, BLUEPRINT (L. Chen et al., 2016), GEUVADIS (Lappalainen et al., 2013), Nedelec_2016 (Nédélec et al., 2016), Quach_2016 (Quach et al., 2016), TwinsUK (Buil et al., 2015).

Study	Donors	Samples	Cell types	Conditions	QTL group count
Alasoo_2018	84	336	macrophage	naive, IFN γ , Salmonella, IFN γ +Salmonella	4
BLUEPRINT	197	608	monocyte, T-cell, neutrophil	naive	3
GEUVADIS	462	462	LCL	naive	1
Nedelec_2016	171	503	macrophage	naive, Salmonella, Listeria	3
Quach_2016	200	970	monocyte	naive, LPS, R848, IAV, Pam3CSK4	5
TwinsUK	433	1364	LCL, skin, fat, blood	naive	4

The eQTL Catalogue is, in essence, a set of QTL summary statistics. To produce QTL summary statistics, raw data should be processed through two distinct procedures: quantification and QTL mapping. Since multiple datasets and QTL groups from various studies are involved in the project, they should be processed uniformly to reduce technical biases and variability between datasets. The fact that datasets are big in volume (more than 250 Terabytes) and they should be processed by QTL groups necessitates to have a scalable and robust processing method. Additionally, this

⁶ <https://ensembl.github.io/eQTL-Catalogue-website/>

method should support addition of new datasets to the project. The method should also enable to process the data in parallel and should support usage of computational clusters. Taking into account all the mentioned requirements, we decided to develop a bioinformatics pipeline to perform both quantification and QTL mapping procedures.

A key contribution of the thesis is the development of a robust and portable data analysis pipeline that can be run across diverse computational environments. The pipeline allows us to uniformly process large datasets in a parallel, scalable and reproducible manner. Additionally, developed pipelines are extremely portable which enables to process data hosted on public and private clouds without the need to download the data locally. This is especially important for the CINECA project⁷, in which we will jointly analyse gene expression data from multiple large national cohorts such as BIOS (Zhernakova et al., 2016) and the Estonian Genome Center (Lepik et al., 2017) where sensitive individual-level genetic data cannot leave country boundaries.

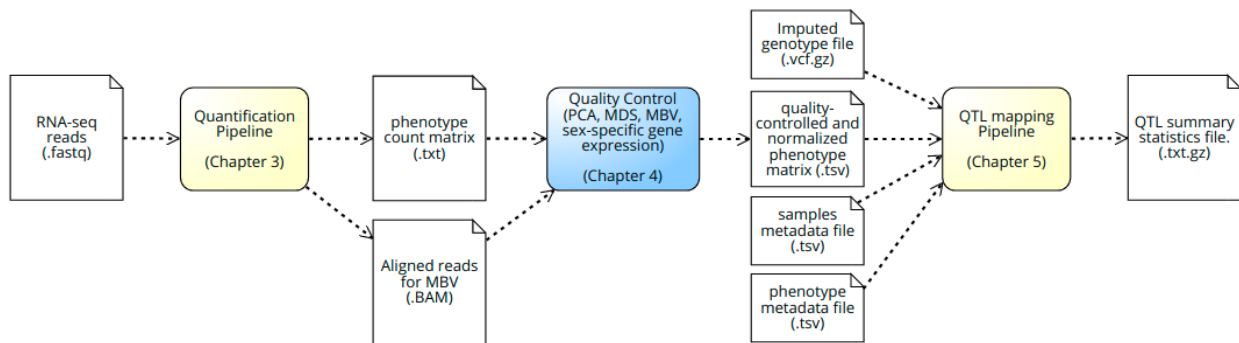


Figure 1.7. High level representation of QTL analysis steps.

The thesis consists of six chapters. In chapter 2, I will explain the properties of modern bioinformatics pipelines, classification of the pipeline frameworks and pipeline design decisions. Then, I will describe each step of QTL mapping analysis in detail. In Figure 1.7 the high-level representation of QTL summary statistics generation (QTL mapping analysis) from raw RNA-seq reads and genotype data is shown. Chapter 3 will contain descriptions of supported quantification methods and technical overview of the developed quantification pipeline. After quantification, a number of quality control steps should be performed to ensure the high quality of the processed data. Post-quantification quality control measures will be extensively described in chapter 4. Finally, I will explain all the details of QTL mapping pipeline in chapter 5, including an overview of the pipeline, description of the QTL mapping process, description of inputs and outputs, and decisions made about the technical implementation. The thesis will end with explanations of conclusions of the related work and list of the used studies' references.

⁷ <https://www.ebi.ac.uk/about/news/press-releases/CINECA-facilitates-transcontinental-human-data-exchange>

2. Bioinformatics Pipelines

Due to fast developments in technology, the cost of DNA sequencing has decreased significantly since the Human Genome Project (Lander et al., 2001) and the 1000 Genomes project (1000 Genomes Project Consortium et al., 2015; Siva, 2008) were completed. Now the whole human genome can be sequenced for less than \$1000. Based on this factor and other technological advances, biological data is being generated in large volumes, and it has been accepted as a big data field for several years (Y. Li & Chen, 2014). The future of genomics data alone is predicted to exceed other big data related fields such as astronomy, YouTube and Twitter by the year 2025 (Stephens et al., 2015).

To cope with increasing volumes of genomics data, better and more efficient processing techniques are required. To satisfy these requirements, new tools are constantly being developed by academic institutions, private companies and government-funded organisations. In data analysis, reproducibility is required to enable the validation and consistency of the study results. Reproducing the results is easy, if the analysis is performed in a single computer, using a software containers such as Docker (Merkel, 2014) and Singularity (Kurtzer, Sochat, & Bauer, 2017), and tools like Jupyter Notebooks (Perez & Granger, 2007) and Rmarkdown (Allaire et al., 2016). However, genomic data analysis usually has to be run in parallel on compute clusters. This analysis has multiple steps that need to be abstracted and coordinated according to the computation environment (execution environment, cluster queue systems, software dependencies etc.). Eventually, results computed in parallel need to be collated together as output of analysis. In the bioinformatics field, this process is called a *pipeline* or a *workflow*. In addition to reproducibility of the results and task parallelisation, the most important requirements for modern pipelines are reusability, portability and dependency isolation.

2.1. Properties of Pipelines

To process genomic data, researchers usually shepherd files through a series of specific steps. The set of these steps is called a bioinformatic pipeline. Modern pipelines need to have the following properties to meet the requirements of modern research.

2.1.1. Reproducibility (Replicability)

Processing a dataset at different times, with same set of parameters should produce the same (or consistent) results, independent of location and computational environment. This property is known as reproducibility of the results and modern pipelines are expected to support it. Nowadays the major scientific journals require to publish data and code to assure reproducibility of the study results (a.k.a. “reviewable research”, (Stodden, Borwein, & Bailey, 2013)). This requirement is reasonable since otherwise, it would not be possible to verify the results presented in the publication. The concept of reproducibility is usually referred to in terms of “provenance” (Gil et al., 2007; Kanwal, Khan, Lonie, & Sinnott, 2017) which in pipeline circles refers to the origins of

input data, tools, results and intermediates (Leipzig, 2018). Replicating (reproducing the results) an experiment in a location and operator agnostic manner is a key element in modern science (Kulkarni et al., 2018). To reproduce the results of a bioinformatics study, the data needs to be accessible and the tools, together with the step-by-step guidelines to reprocess the data, need to be available (*Usability and archival stability of computational tools*).

Data availability - Genomic data and its privacy are concepts which need very careful attention, since misuse of the data can have serious consequences. Genomics is a relatively new field, and with the development of biotechnology and increases in computational power, findings are happening more frequently in comparison to the previous century. It is not known what can be done with the genetic data of the donor in the future and that is why it should be kept according to donor's consent. Taking this potential power of genomic data into consideration, usually, data is separated according to donor's consent to access levels: open access data which is publicly available for anyone to use and managed (controlled) access data which needs special permission for usage. Hence, if the experiment involves controlled access data, corresponding permissions must be obtained to replicate the experiment.

Usability and archival stability of computational tools - After obtaining the data to be reprocessed, the main challenge of replicating an experiment is technical. Usually the problem is one of the two following aspects:

1. The software used in the experiment is not available or can not be installed. A recent comprehensive analysis of 24,490 bioinformatics software resources published from 2000 to 2017 showed that 26% of the tools needed for reproduction were not accessible at all, 24% of accessible software failed to install and 49% were deemed "hard to install" (Mangul et al., 2018). This empirical study also found that the publications introducing new software have significantly more citations if they provide an accessible and easy installation process.
2. The exact guidelines about usage of the software involved in the experiment are unavailable. Usually in bioinformatics data analysis, command-line tools are used, where each tool executes a specific job according to provided parameters. Providing different parameters or not providing the needed parameters can lead to generation of different results. Hence, the information about how exactly the software was used in the experiment should be provided in order to achieve consistent results. If the used software has multiple released versions, it is important to specify which version was used in the experiment, in order to ensure the consistency of results.

Therefore, reproducibility is an essential issue especially because pipelines use several tools. Each tool in the pipeline should be accessible and usage of it should be human-readable and understandable (explanation comments can be added if the code is not self explanatory, e.g. explanation of each parameter used). A pipeline can become useless if even one of the tools is not available or produces inconsistent results.

2.1.2. Portability

In data science fields the primary resource is the data itself. It is understandable that some organisations do not want to share this valuable resource publicly. In genomics, in addition to the value of the resource, there are also legal duties to protect the sensitive data of research participants. For these reasons, some authors and institutions simply are not allowed to share their genetic data. Genetic data analysis is an incredibly dynamic field, which continuously produces new methods, and there is the potential to generate new discoveries by applying new or different analysis methods to already existing data. Genetic data generated (sequenced) with one scientific question in mind can often be used for another study even if it was not initially generated for it (Denk, 2017). However, sometimes, it is challenging to overcome the legal issues to get raw genetic data, and that is where the portability of the pipeline plays a vital role. For instance, sometimes genomic data cannot be moved outside of a country due to local data protection laws. In this case, renting private clouds within the borders of the country and performing the computation there with a portable pipeline is an eligible option. Because, even if the sharing of raw data is not allowed, the summary-level data analysis results generated from the raw data can often be shared with third party organisations or even published publicly, since it is impossible to deduce the donors' private information from the generated results.

Portability is also appreciated when in an organisation an existing infrastructure changes or some other organisation with different technical infrastructure wants to use the pipeline. Being able to publish and re-use existing pipelines can significantly improve the efficacy of the data analysis process, because researchers do not have to reimplement the same pipelines from scratch and can spend more time on interpreting the results. For instance, nf-core (Ewels et al., 2019) is a community curated initiative which provides portable and ready-to-use pipelines for public use.

2.1.3. Scalability

Each step of the pipeline has specific hardware resource requirements such as memory, time and number of needed compute processors (CPU cores) in order to successfully process the specified data. When the specified set of resource requirements and data to be processed becomes available for executing the step (process) of the pipeline, a task scheduler (e.g. job scheduling systems of high performance computing cluster) reserves resources and performs the step. Usually, the resource needs of software used in the pipeline are in a linear relationship with the input file size. When the needed resources become very large to handle because of the size of the input file, software sometimes provides additional options to divide the input file into chunks and process them in a parallel manner with less resources but as multiple individual tasks. Therefore, if the functionality of the software can scale according to the volume of the input data then the software is considered to be scalable. Since the pipeline in essence is a set of tools used in a specific configuration, all the tools used in the pipeline should be scalable in order to consider a pipeline as scalable (Fjukstad & Bongo, 2017).

2.1.4. Dependency Isolation

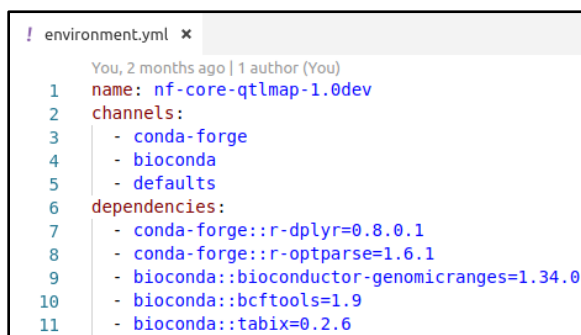
It is very common to use existing tools and libraries in software development. That is how a developed software becomes dependent on other software. Moreover, if any of the dependencies have special needs like a specific operating system or environment, dependent software also inherits these needs. When it comes to bioinformatics pipelines, the situation is similar, especially when pipelines are script wrappers which use a number of external tools to process and analyse data in a structured and efficient manner. Accessibility and stability of software tools used in the pipeline increase according to their level of dependency isolation. Consequently, pipelines with highly isolated dependencies provide better portability and replicability features.

We have classified the dependency isolation levels according to their degree from not isolated to highly isolated.

1. No isolation - This is the case when the dependencies and environment needed for the execution of the pipeline are only documented in a relevant chapter of the pipeline documentation file and it is assumed that the needed tools are waiting in a “ready to use” state in the specific environment the pipeline is designed for. Consequently, any change to the software environment can cause the pipeline to fail (Baggerly & Coombes, 2009; Ioannidis et al., 2009) or generate inconsistent results due to different versions of the software tools used (Piccolo & Frampton, 2016). Typically, single script file pipelines are good examples of this, where all tasks and their order, environment-dependent configurations, and parameters of tools used in the tasks are defined in a single file. Thus, if the working environment changes, the change of the pipeline file becomes inevitable.
2. Isolate workflow logic from the execution environment - When the pipeline is hardcoded to communicate with a specific cluster environment, running it in a different environment takes a lot of effort. However, if the workflow logic and execution environment configuration are isolated, it requires no change in the workflow logic. Usually, workflow frameworks offer easy-to-set configuration options to run the pipeline in a different execution environment. This isolation level still requires manual installation of software required by the pipeline, but isolation between workflow logic and the execution environment is guaranteed. It means that if a user has a workflow and the needed dependencies are installed, a pipeline can be executed with any executor by only changing the configuration file and keeping the workflow logic unchanged. That is a good feature to have, especially since there is a number of different computing platforms that can be used such as SGE, SLURM, LSF, PBS/Torque, NQSII, HTCondor, AWS Batch, Ignite, GA4GH TES and Kubernetes.
3. Conda integration - Conda is a package, dependency and environment manager, which can be used on all major operating systems⁸. Conda downloads software packages from defined channels and installs them into an isolated environment. That is a good way of isolating dependencies for several reasons. Firstly, it is enough to provide a recipe for needed

⁸ <https://conda.io/>

dependencies and Conda will take care of creating an environment with the provided recipe (Figure 2.1). Therefore, researchers with less software experience will not have difficulties running the pipeline, and there is no need to provide the dependency itself but only the recipe. It is also possible to manually add scripts to a created Conda environment, which enables local testing of the developed script in an isolated environment, but in return it decreases the portability of the pipeline. Additionally, Conda is an open source initiative, so that anyone can contribute their own software package to it. Finally, it has a very well tested package repository, which guarantees the stable performance of a specific version of a specific tool. However, Conda environments do not contain an isolated operating system, and some packages are not available for all operating systems, which limits the portability of Conda environments and makes it inferior to containerised isolation.



```
! environment.yml x
You, 2 months ago | 1 author (You)
1 name: nf-core-qtmap-1.0dev
2 channels:
3   - conda-forge
4   - bioconda
5   - defaults
6 dependencies:
7   - conda-forge::r-dplyr=0.8.0.1
8   - conda-forge::r-optparse=1.6.1
9   - bioconda::bioconductor-genomicranges=1.34.0
10  - bioconda::bcftools=1.9
11  - bioconda::tabix=0.2.6
```

Figure 2.1. Example of Conda environment recipe file.

4. Containerisation - Nowadays the best-known dependency isolation is done by containers. Containers contain all the needed software to run the pipeline. It is different from Conda because containers keep the software itself and not the recipe of it. There are some advantages of containers over other dependency isolation methods. First, containers are extremely portable and can make dependencies of a pipeline highly accessible. Consequently, given that the execution environment also supports containerisation technologies, containerised pipeline dependencies will enable portability and reproducibility of the pipeline, basically by easily accessing the dependencies provided within container. As in Conda-like isolation, manually adding custom scripts to a container is also possible, and is not limiting the portability of either the container or the pipeline. Although it is possible to manually build “black-box” containers, this approach is not advisable. Because, even though the pipeline is reproducible (produces the same results when rerun), it is not possible to verify what the custom software actually does in the container. Thus, in order to provide the exact content (tools with corresponding versions) of the container, the recipe used to build it should be also provided. Currently, the most popular software container tools are Docker (Merkel, 2014) and Singularity (Kurtzer et al., 2017). Typically, the software containers are built using a recipe file (e.g. Dockerfile) which contains the base image and step-by-step instructions about how to build the container. The base image is a previously built container, usually containing basic needs of a container such as an operating system. However, the nf-core initiative provided a base

image with an operating system and Conda software installed, together with a Dockerfile and Singularity recipes to use the provided base image. Using these provided resources makes it straightforward and painless to build the container with a Conda-like recipe. For instance, to build a Docker container, Docker software creates a container with base image which contains an operating system and Conda, and creates a Conda environment with provided Conda-like recipe inside the container. Hence, if the pipeline provides a Conda-like recipe for creation of Conda environment, it is possible to easily build a software container with the same provided recipe. After building the container, storing it in a specific web repository like Docker Hub⁹ or Singularity Hub¹⁰ is an easy and common way of storing and sharing containers. Since the container hubs are open for public, it is possible to pull and use already existing containers in the hub. Docker is essentially designed for use of enterprise software production systems and it gives superuser privileges to the user. In multi-user systems such as HPCs, there is no good way of restricting users with such privileges from accessing other users' data. Singularity, another containerization platform, behaves like Docker but does not require administrative privileges to be used (Silver, 2017). Finally, with containerisation it became relatively easier to automate the testing of dependencies. Continuous integration and delivery are vital for most software development organisations and containerisation provides an opportunity to apply well-known practices of dependency isolation and continuous integration to bioinformatics pipeline development.

2.1.5. Parallelisation

Genomic data files usually are large collections of short reads (~100 base pairs, depending on the sequencing platform). Reads are the sequences of characters read out from the DNA. There are several steps in a pipeline to process these files and extract meaningful information. Some of these steps should be performed in parallel to achieve high efficiency in terms of time and cost. Nowadays, even personal laptops usually have more than one multi-core CPUs which can run jobs in parallel. However, to host these parallel operations there are High Performance Computing (HPC) clusters specifically designed to increase the throughput of scientific analysis. HPCs' operating systems are usually Linux based and have job scheduling systems (also called executors) which orchestrate jobs into different nodes to be processed. Each job has its own resource needs, and scheduling systems should use the resources in an efficient way to meet these needs. For instance, given that there are ready-to-use sufficient hardware resources, from the user's point of view, the total running time of the pipeline for processing either 2 or 200 samples should not differ greatly.

⁹ <https://hub.docker.com/>

¹⁰ <https://singularity-hub.org/>

Computational power can also be outsourced if an institution does not want to maintain this computational infrastructure by themselves. Cloud batch computing services are gaining more and more popularity for the following reasons:

- Users do not have to think about maintenance of the cluster since service providers take care of it.
- In bioinformatic analysis, lots of huge reference files are used. Storing these files also has costs. Some cloud batch computing services keep these reference files in common storage which means that anyone can use them without paying extra (Yung et al., 2017).
- Since transferring the files can also be costly, it is usually a good idea to keep the raw data close to the computing power. That is why these services also offer private storage options to keep the raw data in the cloud. Furthermore, some large-scale genomics project such as The Cancer Genomics Atlas have made their data available on the cloud, removing the need for analysts to download hundreds of terabytes of raw data¹¹.
- Cloud services usually offer different pricing options, and usually, a user pays only when they use it. For instance, AWS Batch calls it the “Pay-as-you-go” approach¹². This also makes cloud systems extremely scalable.

On the other hand, cloud systems also have some drawbacks. The major reasons why some institutions decline to use cloud services are the cost of the service and security of their data. First, if there is a high demand to continuous service of computational power in an institution, it is usually cheaper in a long term to acquire inhouse HPC. Secondly, although cloud service providers ensure to keep users’ data safe, some research institutions do not prefer to use these services due to the potential issues described in (Kandukuri, V., & Rakshit, 2009).

2.1.6. Reusability

There are two contexts to evaluate the reusability of a pipeline: *reusability of a pipeline by end-users* and *reusability of the pipeline by other developers*.

Reusability of a pipeline by end-users - Although there are already multiple options for biologists to gain enough informatics knowledge to use bioinformatics software and interpret its results, bioinformatic pipelines should be designed such that minimum configuration change would be enough to run it successfully. In this context, reusability means how easy it is for users to install and execute the pipeline on their own data. The bioinformatics pipeline is considered to be reusable if the the user without prior knowledge of the field can use the pipeline with minimal effort. Additionally, it is acceptable to allow the users to change parameters of the pipeline in order to make it better fit to their data and execution environment. Therefore, giving some options to change the parameters is considered as flexibility of the pipeline, which in return increases the reusability of it. Currently, there is no consensus test to measure the reusability of a bioinformatics

¹¹ <http://www.cancergenomicscloud.org/>

¹² <https://aws.amazon.com/pricing/>

pipeline, that is why feedback provided by users and popularity in the community are considered to be the main measures.

Reusability of the pipeline by other developers (a.k.a. extensibility) - Researchers could want to use an existing pipeline as a basis for the development of new pipelines. In this context, reusability means how easy it is for developers to extend a pipeline or use the source code of a pipeline for development of a new one. The adoption of a “pipeline step modularisation” approach by most institutions made it common to reuse these steps in other studies (Leipzig, 2018).

Open Source Software (OSS) movement is one of the major powers of software engineering nowadays (Carillo & Okoli, 2008). The main beneficial properties of OSS are transparency, easy collaboration and an immense knowledge pool (Dabbish, Stuart, Tsay, & Herbsleb, 2012). The bioinformatics field also used the benefits of the shared development platform. Various institutions have proclaimed developing reusable, robust and open source bioinformatics pipelines as one of their objectives (e.g. (Ewels et al., 2019)). Another benefit of OSS is the community support which is usually provided by code repository issues, chat channels like Gitter¹³, free agile development tools like Slack and Trello and mail groups. With the help of these communication channels, users and developers get quick help on how to use and develop the pipelines, and the main contributors get rapid feedback which results in bug fixes and development of new features.

2.2. Comparison of Pipeline Frameworks and Design Decisions

Some well-known pipelines have been collected in the public repository (Di Tommaso, awesome-pipeline). These pipelines have differences regarding design philosophies, technical issues, difficulty of usage, environment dependency and some other factors. To get a broader view of bioinformatics pipelines a review study by Jeremy Leipzig (2017) classifies existing pipelines according to three criteria: syntax, paradigm and interaction.

2.2.1. Classification of Pipelines

Syntax - *Explicit frameworks* following the idea of tightly linking the tasks together in a certain order and providing inputs to the very first task in order to be processed in a fixed-order chain of tasks. Not relying on input names, output names and transformation rules between them makes explicit frameworks simple and robust but limits the flexibility of the pipeline to process the newly added files in addition to already processed ones. Galaxy (Goecks, Nekrutenko, Taylor, & Galaxy Team, 2010), Taverna (Wolstencroft et al., 2013) and Ruffus (Goodstadt, 2010) are good examples of explicit paradigm using pipelines. On the other hand, in the *implicit syntax frameworks*, it is sufficient to specify inputs and expected outputs (target files), and intermediate steps are calculated automatically by the framework. Hence, the order of tasks is managed implicitly by the framework and not hard-coded by the user (Figure 2.2).

¹³ <https://gitter.im/>

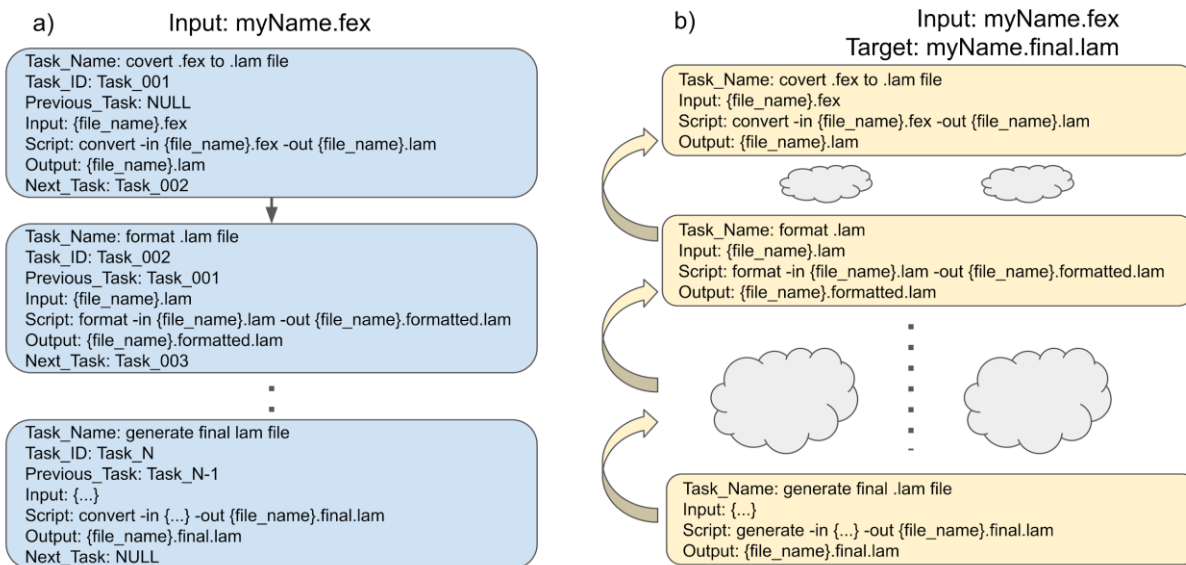


Figure 2.2. a) Symbolic representation of **explicit pipeline** frameworks. The tasks are tightly coupled in a specific order. Input file is processed step by step in a chain of ordered tasks to produce an output file. b) Symbolic representation of **implicit pipeline** frameworks. There is no fixed order of tasks. Instead, framework calculates intermediate steps according to provided target file and input file.

Implicit frameworks are mainly descendants of Make which was designed in the '70s as one of the early developed domain specific languages (DSL). Make offers a set of rules and a symbol-based syntax which define how input files will be transformed into outputs, and output files will feed the next step with inputs and so on. The implicit DSLs' crucial feature is reentrancy, i.e. ability to distinguish already processed files and not reprocess them. It checks the modification date of the input file and compares it to the modification date of an existing output file. If the target file exists and the input file is older than the target file, Make considers it as an already processed file. Reentrancy is vital for a pipeline development process when potential errors are expected and recovering from them takes less time thanks to this ability. Some implicit frameworks such as Nextflow (Di Tommaso et al., 2017) and Snakemake (Köster & Rahmann, 2012) follow the Make concept with the support of full-featured programming languages, respectively Groovy and Python. Snakemake, as a faithful follower of Make, is file-centric, whereas Nextflow introduces channels to pass intermediate forms of steps to each other which eliminates the need of tagging intermediates with complex file suffix names (Leipzig, 2018). In Nextflow, input and outputs are typed values which offer more flexibility concerning keeping intermediates in memory while ensuring reentrancy with caching.

Design Paradigm - DSLs are known to be designed based on a *conventional* design paradigm, when a directed acyclic graph (DAG) is dynamically generated according to the syntax rules of the language, input files and target files. On the other hand, explicit frameworks create DAGs with the help of APIs or visual user interfaces. Explicit frameworks' DAG structure is static, where an ordered chain of tasks is described in a fixed XML or JSON format, and changing the provided

input does not change the structure of generated DAG. Hence, explicit frameworks are considered to be members of *configuration-based* paradigm. Another design paradigm called *class-based* paradigm covers design principles of frameworks that are bounded to an existing code library instead of independent executables. More detailed analysis of design paradigms is provided in (Leipzig, 2017).

Table 2.1. The classification of modern pipeline frameworks. Table is obtained from (Leipzig, 2017) and used without modifications

Syntax	Paradigm	Interaction	Example	Ease of Development	Ease of Use	Performance
Implicit	Convention	CLI	Snakemake, Nextflow, BigDataScript	★★★★☆	★★★★★	★★★★
Explicit	Convention	CLI	Ruffus, bpipe	★★★★★	★★★★★	★★★★
Explicit	Configuration	CLI	Pegasus	★★★★☆	★★★★	★★★★★
Explicit	Class	CLI	Queue, Toil	★★★★☆	★★★★★	★★★★★
Implicit	Class	CLI	Luigi	★★★★★	★★★★★	★★★★★
Explicit	Configuration	Open Source Server Workbench	Galaxy, Taverna	★★★★★	★★★★★	★★★★★
Explicit	Configuration	Commercial Cloud Workbench	DNAexus, SevenBridges	★★★★☆	★★★★★	★★★★★
Explicit	Configuration	Open Source Cloud API	Arvados, Agave	★★★★★	★★★★★	★★★★★

Interaction - In contrast to command-line based pipeline frameworks, workbenches provide users with a graphical user interface. In this interface, the user can add nodes (pipeline tasks - preconfigured modular tools) and connect them (representing the data flow - the outputs of the previous step become the input of the next) to develop a pipeline. These kinds of frameworks are very suitable for scientists who have an understanding of the expected input and output files but have a little or no coding experience. One drawback is that existing modular steps have to be sufficient for the analysis. Open source workbenches can be installed both locally and in the cloud. These workbenches convert a graphically designed pipeline to a configuration-based pipeline in the background. With their large number of options, Galaxy (Goecks et al., 2010) and Taverna (Wolstencroft et al., 2013) are the most popular ones. Galaxy offers a web-based interface for command-line tools. Galaxy's interface is easy to use, but it needs some coding skills to add new modules. Taverna, on the other hand, allows pipelines to reach tools distributed across the Internet and needs more development skills to develop new plug-ins (modules). There are some commercial cloud-based Software as a Service (SaaS) workbenches such as Illumina's

BaseSpace¹⁴, SevenBridges¹⁵ and DNANexus¹⁶. Cloud-based services like these make the pipeline reusability, sharing and collaborations easier. To sum up the findings, the tabular representation of the modern pipeline frameworks' classification made by Jeremy Leipzig (Leipzig, 2017) is shown in Table 2.1.

2.3. Pipeline Requirements of the Project

Having a variety of pipeline development frameworks providing number of different features makes it difficult to choose the best among them. One pipeline framework can be highly scalable but lack flexibility to modify the pipeline by the users. Therefore, a pipeline framework should be chosen according to the specific needs of the project. Our project requires that the pipeline to be developed should have the following features:

1. Pipeline should be portable between a wide range of different compute environments. It can be developed and prototyped in the HPC at University of Tartu, but it should be able to run in other environments such as EBI Embassy Cloud¹⁷, Google Cloud¹⁸ and HPC environments of our partners with minimal effort. Next release of GTEx (GTEx Consortium, 2013) will be distributed using Google Cloud and is expected to contain 160 terabytes of data. Portability of pipeline should enable the analysis of this data without downloading it locally, which can take several weeks and requires lots of local space.
2. Pipeline should be able to handle large volumes of data. The volume of genomic datasets is continuously increasing and performance of the pipeline should scale accordingly.
3. Pipeline should support reentrancy. Processing time of datasets is usually correlated with their size. Thus, processing large datasets can take considerable amount of time. If the execution of a pipeline is stopped for any reason, rerunning the pipeline should not reprocess already processed entities, but should resume from the point it stopped at.
4. Pipeline should be robust to disruptions in computation. Even in very sophisticated systems disruptions of computation can occur. Pipeline should be able to handle the exceptions and resubmit the interrupted tasks. For instance, usually each individual process of a pipeline has specific resource requirements, such as number of processors and memory. Sometimes specified resources can become insufficient due to abnormal size of an intermediate file, and consequently the task can stop running. Pipeline should understand the nature of the error message and increase the specified resource requirement for that specific task instance.
5. Pipeline should be reproducible. Running the pipeline in two different execution environments (at different times) with the same input and parameters should result in the exact same results.

¹⁴ <https://basespace.illumina.com/>

¹⁵ <https://www.sevenbridges.com/>

¹⁶ <https://www.dnanexus.com/>

¹⁷ <https://www.embassycloud.org/>

¹⁸ <https://cloud.google.com/>

6. Pipeline should be inspectable. All the steps of the developed pipeline should be open-source, human-readable and understandable.
7. Execution of pipeline should be traceable. Usually the final target files are the main point of interest in pipeline execution. However, in order to produce target files, vast amount of intermediate files are generated, and sometimes intermediate files play an important role in understanding unexpected content of target files. Furthermore, intermediate files can be used as inputs of third party tools in order to empower the data analysis with additional information. Thus, the pipeline should gracefully handle large numbers of intermediate files and automatically delete them when specified to do so.
8. It should be easy to install and run the pipeline. For instance, support of software containers and Conda environments should be provided to minimize the effort to set up the execution environment.
9. Pipeline should follow the best practices of the pipeline development process. Maintenance is crucial for longevity of the pipeline. To minimize the effort spent on maintenance, best practices should be applied. Best practices are evolved according to needs of the user and developer communities. Used framework should have a community who has already developed such pipelines and has experience with best practices to be followed.
10. Pipeline should be reusable and extensible. Modern pipelines should adapt to the new needs of the the user community. Hence, the pipeline should be open and eligible to modifications and extensions by developers in order to change used tools or to add new features.
11. Pipeline should support continuous integration. Although usually there is a main developer and maintainer of the pipeline, open-source projects are often also changed by other developers. To ensure the functionality and the quality of the project, automatic tests are performed after each modification and the changes are added to the repository only after all tests have passed.

2.3.1. Comparison of Most Popular Frameworks

Fortunately, the evolution of pipeline design already serves us good frameworks to develop pipelines which provide all of the listed properties. Nonetheless, we have different options concerning which framework to use for pipeline development, each pipeline framework has its own benefits and drawbacks, and there is no formula to prefer one over the another. A suitable framework can be chosen according to the specific goals of the project. The most popular frameworks to construct bioinformatics pipelines are Nextflow, Snakemake and Galaxy. Table 2.2 shows the overall comparative properties of these frameworks. As it can be seen from the table, there are no significant differences between the frameworks. All three of them support the main features of modern pipelines (e.g. dependency isolation, reentrancy, scalability, reproducibility), and that is the reason why they are most popular among others. However, Galaxy is different from the other two, being designed for users with little or no coding skills. Being less extensible, lacking flexibility of changing hard-coded tool requirements (e.g. needed computational resources) and exception handling makes this configuration based framework a weaker candidate for this project.

Nextflow and Snakemake are similar frameworks because of their design paradigm. However Nextflow, in comparison to Snakemake supports exception handling and has a better community providing best practices to easily develop a pipeline from scratch (Ewels et al., 2019).

Table 2.2. Comparison of three most famous pipeline frameworks. The evaluation of properties with count of stars (*) is highly subjective and current values originate from the performed literature review and personal experience of the author.

	Nextflow	Snakemake	Galaxy
Syntax	Implicit	Implicit	Explicit
Paradigm	Conventional	Conventional	Configuration
Interaction	Command-line	Command-line	GUI Workbench
Dependency isolation	Conda, Docker and Singularity	Conda and Singularity	Conda, Docker and Singularity
Dynamic Exception Handling	Supported	Not supported	Not supported
Reentrancy	Supported by using caching	Supported by tracking file names and modification times	Support as a built-in feature with less flexible options
Scalability	***	***	***
Reproducibility	***	***	***
Extensibility and Ease of development	***	***	*
Inspectability	***	***	**
Community, availability of best practices, CI and documentation	***	*	**
Flexibility	***	***	*

Additionally, Nextflow provides a better reentrancy feature based on caching, making debugging issues easier and intermediate files inspectable. Snakemake, on the other hand, makes it easier to iteratively develop and document data analysis projects, because all of the intermediate files are stored in an explicit folder structure.

2.3.2. Pipeline Design Decisions

Taking into consideration the existing needs of the project, we decided to use the implicit DSL, command-line based script wrapper Nextflow (Di Tommaso et al., 2017) for the following reasons:

- It is straightforward to install and run.
- It offers an excellent framework to develop executor and environment agnostic pipelines. Nextflow follows “develop once and run everywhere” approach which is extremely suitable for our needs since our pipeline should be able to run with many executor engines including cloud clusters.
- It provides good integration with containers like Docker and Singularity, and popular package manager Conda (Grüning et al., 2018).
- It has a very helpful community and easy-to-communicate channels to get help rapidly.
- There are open-source projects like nf-core¹⁹, which provide ready-to-use pipelines that we can adopt and make modifications according to our needs. Nf-core provides also a collection of best practices and reasonable defaults for pipeline developers.
- It has good error handling features which make pipelines robust. In bioinformatics pipelines each step has minimum resource requirement. In case the process (step) raises an error due to resource insufficiency, nextflow provides an option to automatically resubmit the task with increased resource requirements.
- It provides reentrancy feature using the caching, which enables to track intermediate files of a particular process. This feature is extremely useful in development stage when a lot of debugging actions are required.
- It provides a good feature for pipeline versioning which increases reproducibility.
- Continuous integration and continuous testing tools such as Travis CI²⁰ can be easily implemented with Nextflow pipelines.
- Steps (tasks) accept scripts of any scripting language as long as the needed language is installed (or provided within a container) and ready-to-use in the pipeline running environment.
- Distinction between intermediate files and final outputs is clear. Final outputs can easily be stored in a permanent location without explicitly removing intermediate results. This is especially important for quantification pipeline that can produce terabytes of intermediate results whereas final outputs are very small.

3. Quantification Pipeline

The long term goal of the eQTL Catalogue project is to contain not only eQTLs (gene expression QTLs) but also transcript usage (tuQTL), alternative splicing QTLs (sQTL) and exon expression

¹⁹ <https://nf-co.re/>

²⁰ <https://travis-ci.org/>

(eeQTL). To be able to detect these QTLs, we first need to quantify gene expression, transcript usage, alternative splicing usage and exon expression from RNA-seq data (Figure 3.1).

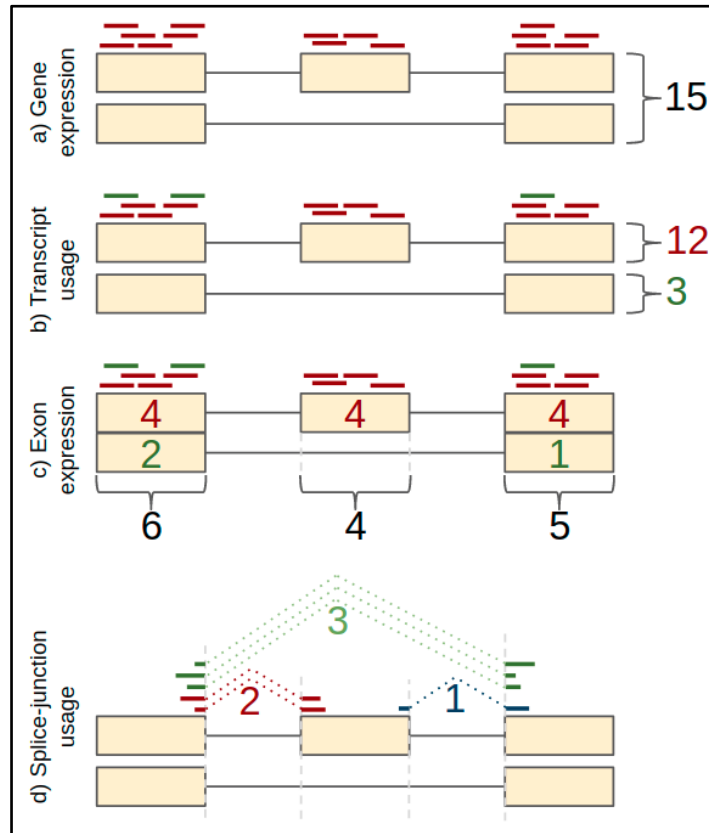


Figure 3.1. Description of counting reads as gene expression, transcript usage, exon expression and splice-junction usage. A gene described in example has 2 transcripts, 3 exons and 3 different splice junctions. a) All reads mapping to the gene are summed together to estimate gene expression. b) Reads are assigned to the transcripts that they are most likely to originate from. c) Expression level of each exon is quantified separately. d) Reads mapping to splice-junctions are used to distinguish between two alternatively spliced transcripts.

3.1. Quantification Methods

Several tools have been developed for quantification of these phenotypes (Figure 3.1) (Anders, Reyes, & Huber, 2012; Dobin et al., 2013; Kim, Langmead, & Salzberg, 2015; Liao, Smyth, & Shi, 2014). Every one of these tools serves to solve a particular problem to reach the common goal: to produce a phenotype matrix of a quantitative trait (Table 3.1). Additionally, there are already well established pipelines which use a subset of these tools and produce count matrix of one phenotype^{21,22}. However, currently there is no public, uniform pipeline which takes the RNA-seq raw data and quantifies multiple quantitative traits such as transcript usage, exon expression and alternative splicing usage in addition to gene expression. We adopted the nf-core rna-seq pipeline²³ for gene expression, and added the following new quantification methods:

²¹ <https://usegalaxy.org/u/chmy/w/rna-seq-differential-analysis>

²² <https://github.com/Novartis/EQP-cluster>

²³ <https://github.com/nf-core/rnaseq>

- Transcript usage
- Exon expression
- Alternative splicing usage

To quantify gene expression, exon expression and alternative splicing usage, RNA-seq reads should be aligned to the reference genome to determine the location from which they are originated. STAR (Dobin et al., 2013) and HISAT (Kim et al., 2015) are well-known RNA-seq aligners, using different algorithms to achieve the goal. STAR uses suffix arrays to provide fast aligning, however requires a large amount of random access memory (~27 GB of RAM) to function. HISAT, on the other hand, uses an indexing approach based on the Burrows-Wheeler transform (M. Burrows, 1994) and the Ferragina-Manzini (Ferragina & Manzini, 2000) index, and requires less memory, making it possible to align reads even in personal computers (Kim et al., 2015). Aligner tools take the raw RNA-seq data (FASTA/FASTQ format (Cock, Fields, Goto, Heuer, & Rice, 2010)) and the reference genome file (FASTA format) as inputs and output aligned sequence files (SAM/BAM format (H. Li et al., 2009)) and some additional files with metadata. To quantify transcript usage, we used Salmon (Patro, Duggal, Love, Irizarry, & Kingsford, 2017) which does not align reads to the reference genome, but uses the reference transcriptome instead (nucleotide sequences of all transcripts on the reference chromosomes, FASTA format).

Table 3.1. Example of quantified phenotype matrix of gene expression. First two columns contain phenotype information (ID and length of the phenotype). Starting from third column, each column name represents the sample ID and column values represent expression level of the corresponding phenotype.

gene_id	length	UCF018	UCB018	UCT018	UCB019	UCT019	UCB024
ENSG00000223972	1735	0	0	0	0	0	0
ENSG00000227232	1351	3	8	14	8	10	22
ENSG00000233750	3812	0	1	0	0	2	1
ENSG00000268903	755	1	7	9	11	0	25
ENSG00000279457	1397	11	31	16	29	29	28

3.1.1. Gene Expression Quantification

Gene expression is the most commonly used quantification method in RNA-seq analysis. It corresponds to the total number of RNA-seq reads mapping to the gene (Figure 3.1a). We use the rnaseq pipeline²⁴ developed by the nf-core (Ewels et al., 2019) community. This pipeline provides two alignment options: HISAT and STAR, preprocessing and quality assurance tools like fastqc²⁵, cutadapt (Martin, 2011), trim_galore, (Krueger, 2015), preseq²⁶, RSeQC (L. Wang, Wang, & Li,

²⁴ <https://github.com/nf-core/rnaseq>

²⁵ <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>

²⁶ <http://smithlabresearch.org/software/preseq/>

2012), picard tools²⁷, dupRadar (Sayols, Scherzinger, & Klein, 2016) and multiQC (Ewels, Magnusson, Lundin, & Källér, 2016). To summarise aligned reads, the pipeline uses the featureCounts tool: an efficient program for assigning sequence reads to genomic features. (Liao et al., 2014).

3.1.2. Transcript Usage Quantification

To estimate the relative expressions of alternative transcripts (Figure 3.1b) we used Salmon (Patro et al., 2017). Quantifying with Salmon consists of three steps: building a salmon index, quantification of transcripts and merging the outputs. To build a salmon index, the only needed input is the reference transcriptome (FASTA file). For quantification, Salmon takes the built index and one raw RNA-seq (FASTQ) file as inputs and estimates the expression of each transcript. Salmon is able to quantify individual reads in the FASTQ file in parallel using multiple threads. Salmon quantification output contains transcript id, length, effective length, counts per kilobase million (TPM normalised count) and number of reads assigned to the transcript. At the end, when the transcript expressions of all RNA-seq samples are quantified, independent output files are merged into a single phenotype matrix file (Figure 3.2).

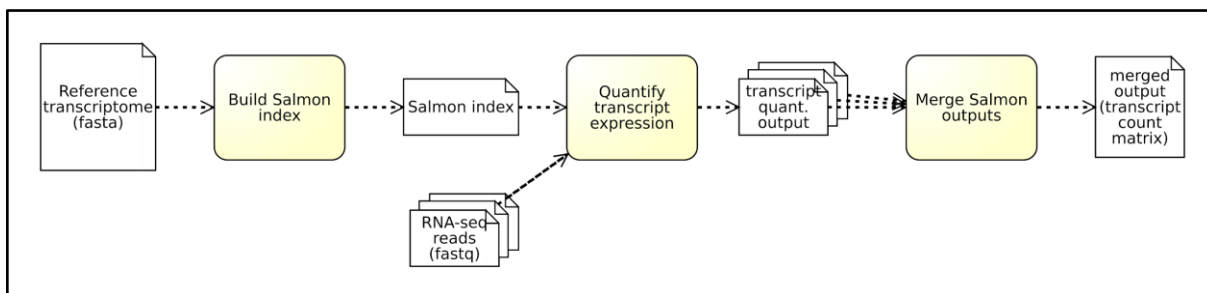


Figure 3.2. High-level representation of transcript usage quantification with Salmon.

3.1.3. Exon Expression Quantification

To quantify exon expression levels in RNA-seq data (Figure 3.1c) we used the DEXseq (Anders et al., 2012) package from Bioconductor. DEXseq takes an aligned reads (SAM) file and counts the number of reads mapped to a specific exon of the gene. Technically, DEXseq consists of a pair of python scripts: one to prepare DEXseq annotation file and another for counting reads mapped to exons. To build an annotation file, DEXseq needs Gene Transfer Format (GTF) file. It processes the exons in GTF file and creates a customized exon annotation file (General Feature Format - GFF) where exons do not overlap each other. The second script takes the built annotation file (GFF) and aligned RNA-seq reads (BAM), and counts the reads overlapping with new custom exons in the annotation file. Each aligned file is quantified individually, therefore the counting step produces output for each sample. When exon usage counts are quantified for all samples, output files are merged into a count matrix file (Figure 3.3).

²⁷ <http://broadinstitute.github.io/picard/>

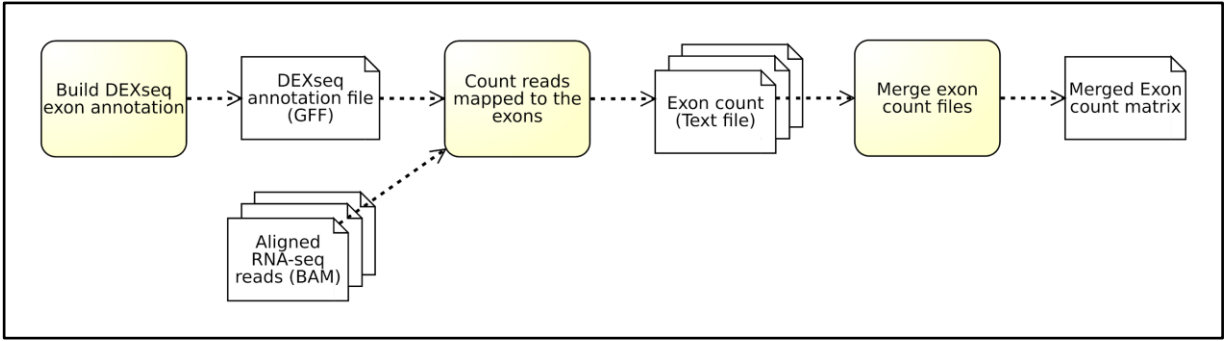


Figure 3.3. High-level representation of exon expression quantification with DEXseq.

3.1.4. Alternative Splicing Usage Quantification

Although multiple tools exist to quantify alternative splicing events (Goldstein et al., 2016; Griffith et al., 2010; Trapnell et al., 2012), we preferred to use LeafCutter (Y. I. Li et al., 2018) which directly measures splice-junction usage and does not rely on known transcript annotations. Splice-junction usage is a way to quantify alternative splicing from the RNA-seq data by looking at RNA-seq reads where one half of the read maps to one exon and the other half to another exon, revealing the intron that has been spliced out (Figure 3.1d). LeafCutter takes the aligned reads as input (BAM) and detects the differences in intron excisions for each input file and clusters them according to their junctions (Figure 3.4). This cluster of the different intron excisions is the final output (a.k.a. phenotype count matrix) of the LeafCutter tool.

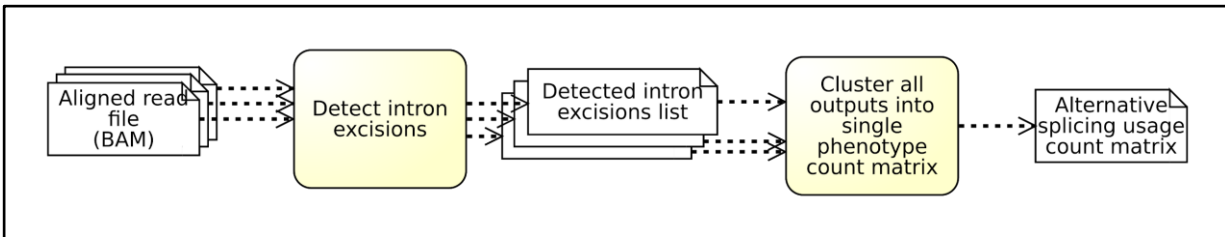


Figure 3.4. High-level representation of splice-junction usage quantification with LeafCutter.

3.2. Pipeline overview

The implemented RNA-seq quantification pipeline quantifies all the listed phenotypes in a parallel manner to achieve high efficacy in terms of cost and time. Raw RNA-seq sample reads go through Quality Control steps and are pre-processed to become ready for further processing. Pre-processed sample reads can go directly to transcript expression quantification, however, they should be aligned to the reference genome for gene expression, exon expression and alternative splicing usage quantifications. In the quantification step, prerequisite resources of each tool (featureCounts, DEXseq, Salmon and LeafCutter) are already prepared and are ready to be used (not shown in Figure 3.5). Each aligned sample is quantified individually and a count matrix is generated for each sample. When the outputs of all samples for one quantification type have been generated, they are merged together to form a merged phenotype count matrix (Figure 3.5).

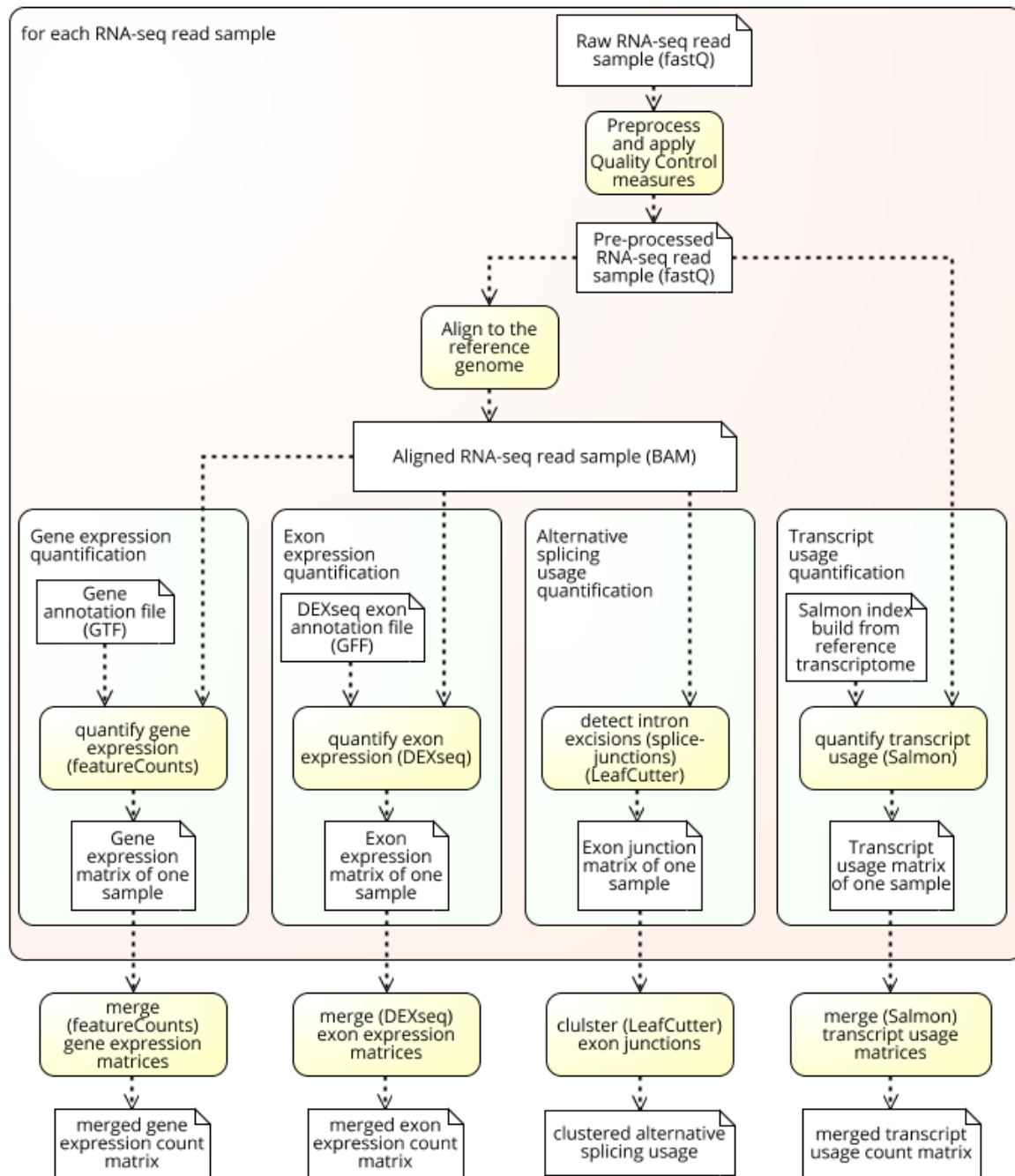


Figure 3.5. High-level representation of quantification pipeline including four different phenotype quantification method descriptions.

In addition to features that are provided by Nextflow as a framework, adapting the nf-core pipeline according to our needs has a number of benefits. First, since nf-core pipelines are actively maintained by the community, the tools used in the pipelines are up to date. Secondly, they provided well documented guidelines including best practices including a tool²⁸ to simplify the

²⁸ <https://github.com/nf-core/tools>

pipeline development process and a base container image to ease building software containers (Docker and Singularity) using a Conda environment recipe. Integrating the Travis CI²⁹ tool enabled to easily support the continuous integration of pipeline changes. Finally, the nf-core rnaseq pipeline³⁰ is one of the first developed pipelines by nf-core community, hence it continuously evolved applying the best known practices. It is currently used by many sequencing facilities such as SciLifeLab³¹ and Wellcome Sanger Institute³², which gives additional confidence to adopt this pipeline as a base for our pipeline.

We did not modify the existing gene expression quantification steps, but added three more quantification methods into the pipeline. We followed the best practices and added the necessary documentation about usage of the pipeline. The pipeline is freely available for download from GitHub³³.

4. Quality Control and Normalisation

Assessing data quality is essential in studies that contain hundreds of independent samples, because low quality samples can manifest as extreme outliers in the dataset. Outlier samples can in turn significantly reduce the power of detecting QTLs (Ellis et al., 2013) or skew the overall result of the analysis. Furthermore, sometimes two or more samples contaminate each other due to minor human errors in the laboratory, so that genetic material from the sample of one individual is present in a sample of another individual. This error is called sample cross-contamination and can also reduce power to detect QTLs. Therefore, in addition to pre-quantification quality control (QC) steps, we applied post-quantification QC measures such as *Principal Component Analysis* (Wold, Esbensen, & Geladi, 1987) and *Multidimensional Scaling* (Cox & Cox, 2000; Kruskal, 1964) to detect outliers, and *sex-specific gene expression analysis* (’t Hoen et al., 2013) and *sequence-genotype matching analysis* (Fort et al., 2017) to detect contaminated and swapped samples. To apply these QC measures, estimated feature counts should be normalised according to the quantification method. For instance, we used Transcript Per Million (TPM) (Wagner, Kin, & Lynch, 2012) method to normalise gene expression and transcript expression counts, and filtered out lowly expressed phenotypes (median(TPM) < 1), because lowly expressed phenotypes usually do not contribute much to the meaningful signal, but considerably increase the level of noise.

Although the script that produces these QC figures and tables is automated, we have decided to keep the process of identifying low quality samples and resolving conflicts between sample identities between RNA-seq and genotype data manual, because extracting thresholds for these decisions are often dataset specific and require human judgement.

²⁹ <https://travis-ci.org/>

³⁰ <https://github.com/nf-core/rnaseq>

³¹ <https://www.scilifelab.se/>

³² <https://www.sanger.ac.uk/>

³³ <https://github.com/kerimoff/rnaseq>

4.1. Principal Component Analysis (PCA)

PCA is a linear dimensionality reduction method which aims to collect most of the variance in a multidimensional dataset inside the principal components. As a result, it becomes possible to plot most of the variation and see if there are any samples in the dataset that look like obvious outliers. PCA is one of the most commonly used procedures to summarise a multivariate dataset and detect outliers in sample population.

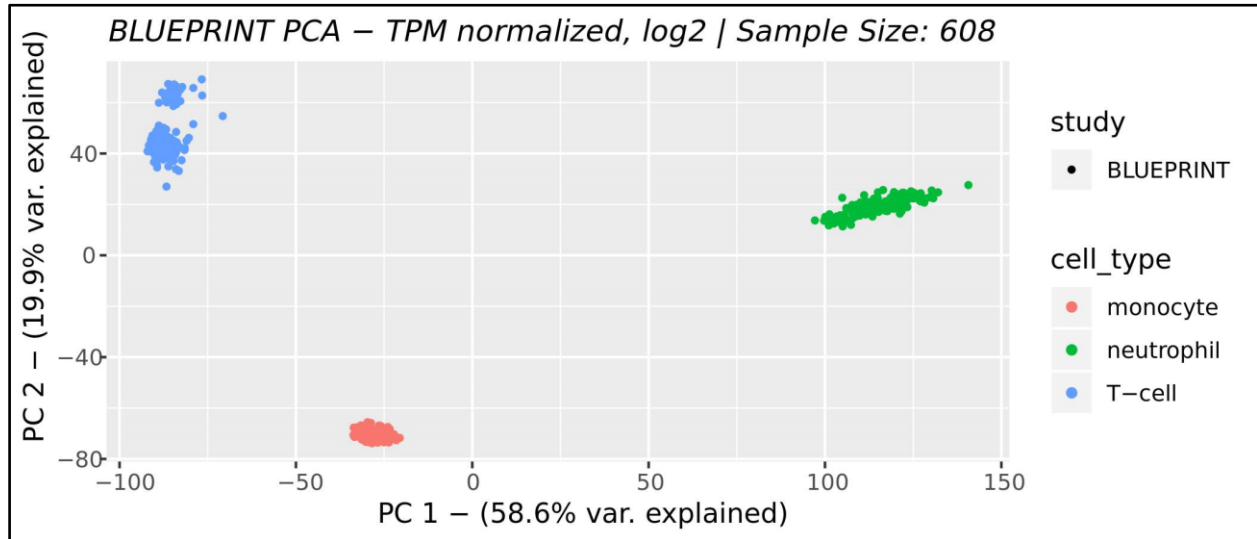


Figure 4.1. PCA plot example from BLUEPRINT dataset (no outliers).

The BLUEPRINT dataset consists of 608 samples from three distinct cell types (monocytes, T-cells and neutrophils) that form three distinct clusters in the PCA analysis (Figure 4.1). This dataset has no sign of any outlier sample which means PCA analysis did not find any poor quality samples to be eliminated from the dataset. Some studies contain several cell types or conditions, whereas others focus on only one cell type and have only one (naive) condition (Appendix 1). One such dataset is the collection of human induced pluripotent stem cells (iPSCs) generated by the HipSci project (Kilpinen et al., 2017). A PCA plot of the latter type of datasets usually look like Figure 4.2: one big cluster of samples without clear boundaries. The two outliers marked with a circle in plot are different than other samples. Principal component 1 values of these outliers differ from the cluster of other samples. This fact necessitates to look for the reason of this variance, which results in the decision of if the sample will be excluded from further processing with the QTL mapping pipeline or not. Usually, in outlier analysis, the reason of the variance is not evident (e.g. low sample quality, library preparation errors, sequencing errors, etc), and it is impossible to enhance the quality of the outlier samples (correct the unknown variance). Similarly, we could not find the reason of the variance of these two outliers and decided to exclude them from the dataset.

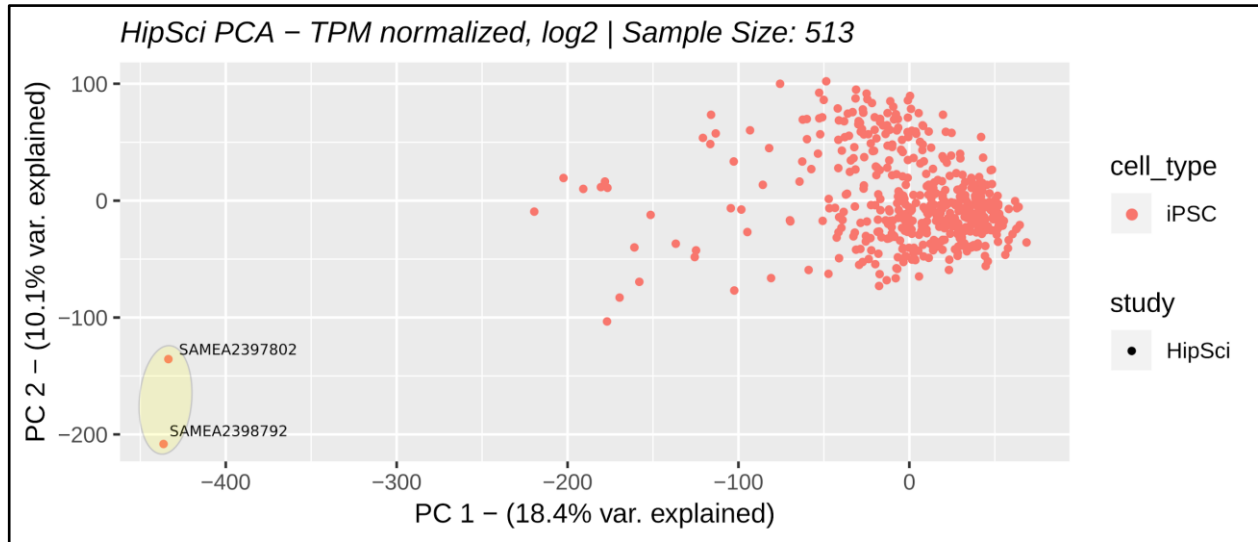


Figure 4.2. PCA plot example from HipSci dataset. Samples *SAMEA2397802* and *SAMEA2398792* are outliers.

PCA of the TwinsUK dataset (Buil et al., 2015) reveals four obvious clusters which represent the four cell types and tissues that were profiled (blood, fat, skin and lymphoblastoid cell lines (LCLs)) (Figure 4.3). Samples marked with a highlighted circles appear to be outliers with no obvious reason, and we decided to remove them from the dataset.

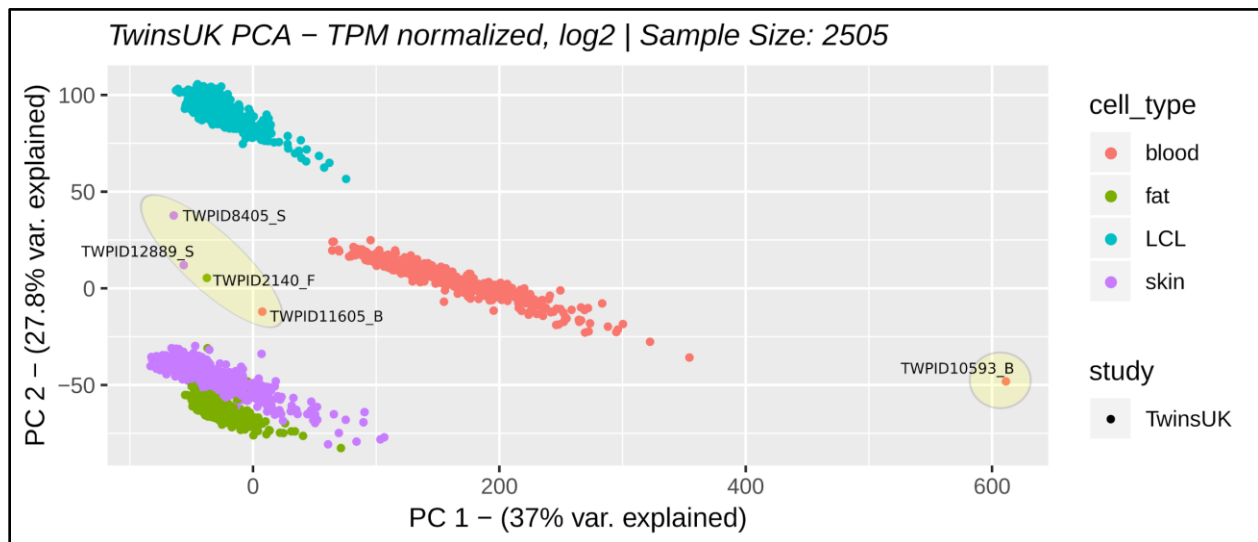


Figure 4.3. PCA plot example from TwinsUK dataset. Samples *TWPID8405_S*, *TWPID12889_S*, *TWPID2140_F*, *TWPID11605_B* and *TWPID10593_B* are outliers.

PCA is a well-recognised and easy to perform method to find outliers in the data. However, when the data is about biological signals, usually PCA explains only a fraction of information in two principal components. For instance, in Figure 4.2 only 28.5% of all variance is explained in the first two principal components. Although in this case it is sufficient to recognise outliers, sometimes it is beneficial to explore also more (e.g. third and fourth) principal components.

4.2. Multidimensional Scaling (MDS)

MDS is an exploratory technique used to identify unrecognized dimensions of the dataset (Mugavin, 2008). MDS reduces a multidimensional dataset to relatively simple, easy-to-visualize structures, which helps us to identify outliers after plotting and analysing it. In contrast to PCA, MDS can perform a non-linear dimensionality reduction using distances between each pair of samples. It can also force all of the data into a small number of dimensions (e.g. two dimensions) that simplifies visualisation. In contrast, since principal components are by definition orthogonal to each other, complex datasets are often not adequately summarised by the first two principal components. After reducing the phenotype count matrix into two dimensions using MDS, we explored outliers. TPM (Wagner et al., 2012) values were used in log₂-transformed (log₂(0.1 + TPM)) scale, after filtering out lowly expressed phenotypes (median(TPM) < 1). Pearson correlation was used as the correlation measure and distances between samples were defined as distance = 1 - correlation. We used isoMDS function from MASS R package (Cox & Cox, 2000; Ripley, 2007; Vernables & Ripley, 2002) with two desired dimensions (k=2) to summarise the data into. This technique was first used by Genotype-Tissue Expression (GTEx) Consortium to visualise gene expression variability among several tissues across individuals (Melé et al., 2015).

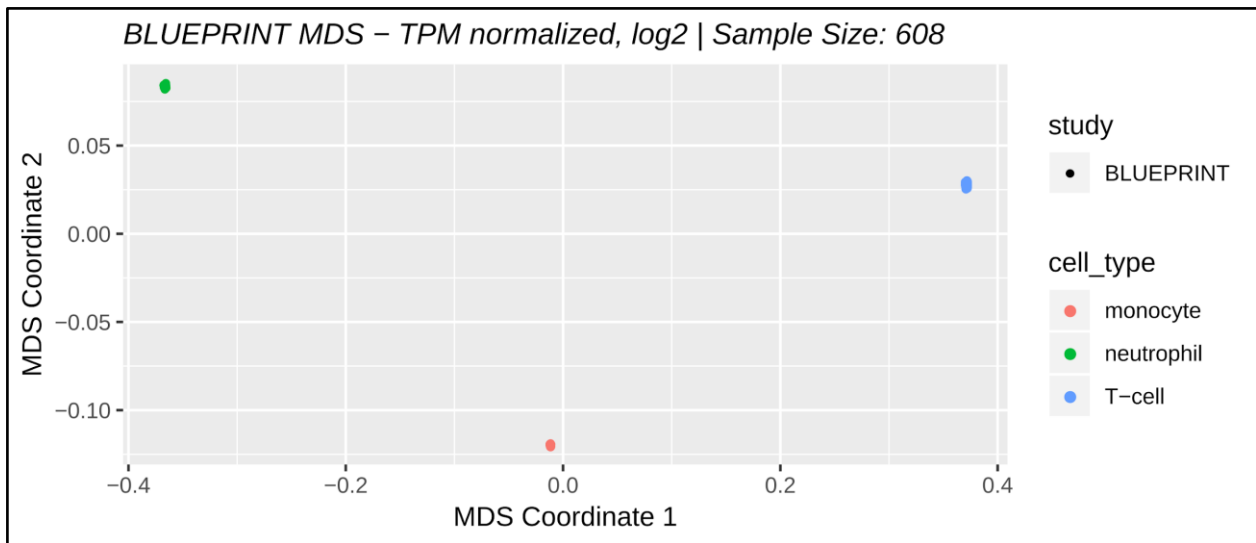


Figure 4.4. MDS plot example from BLUEPRINT dataset (no outliers)

In Figure 4.4 MDS plot of the BLUEPRINT (L. Chen et al., 2016) dataset is shown. Dataset contains three cell types and all three of them are clustered distinctly. There are no outliers to be analyzed further, and we consider data resulting in this kind of plot as high quality. However, in Figure 4.5 it can be clearly seen that samples marked with circles are located very far away from their respective clusters. These samples are considered as outliers and removed from dataset.

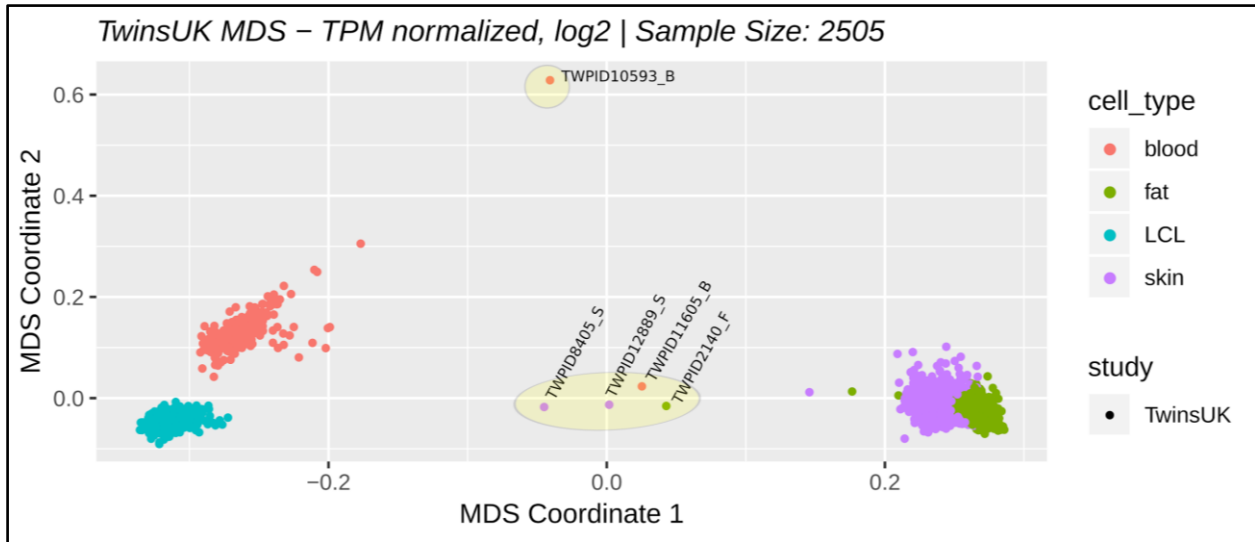


Figure 4.5. MDS plot example from TwinsUK dataset. Samples TWPID2140_F, TWPID10593_B, TWPID11605_B, TWPID12889_S and TWPID8405_S are outliers.

As expected, the same samples appear to be outliers (TWPID2140_F, TWPID10593_B, TWPID11605_B, TWPID12889_S and TWPID8405_S) in both MDS (Figure 4.5) and PCA (Figure 4.3) analysis of TwinsUK dataset. This scenario gives additional confidence to eliminate these outlier samples from the dataset.

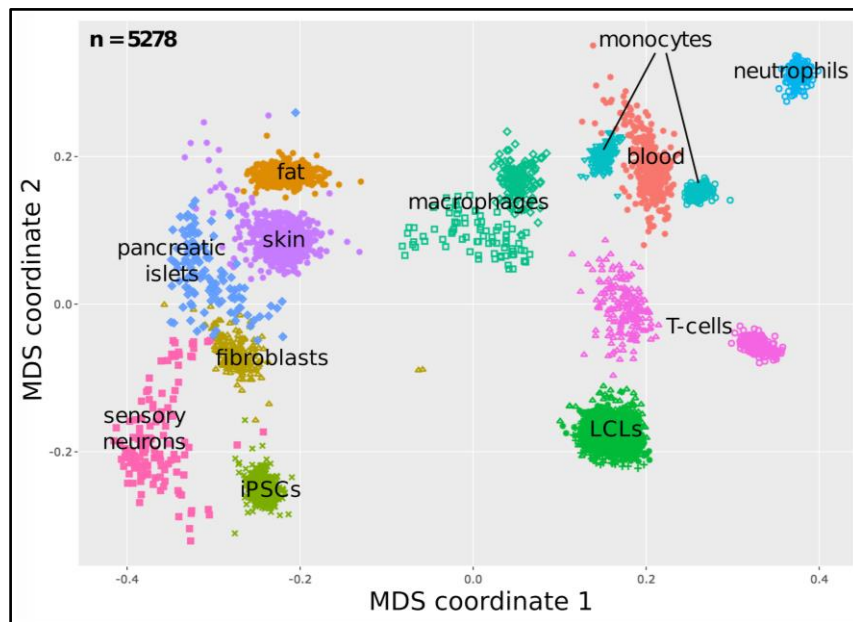


Figure 4.6. MDS plot of the 5278 distinct samples from seven datasets. Contains. Cell type and tissue specific clusters are clearly explaining most of the variance between samples. Clusters of Monocytes and T-cells are divided into two subclusters, showing additional variance depending on differences between studies' datasets.

To get a wider overview about variance between datasets we merged the several datasets and plotted the MDS output after decreasing the number of dimensions to two. These kinds of plots help us to understand the main contributing factors of the variance into the datasets. In order to

make the plot interpretable we only use samples with the “naive” condition (without any stimuli) and do not use datasets with very large number of biological contexts (e.g. GTEx). This filtering operation enables to decrease the noise and see the samples clustered according to some biological property. For instance, Figure 4.6 is a visualisation of MDS analysis of 5278 samples from 7 datasets. 12 tissues and cell types are clustered distinctly, explaining the source of the main variance between samples. However, monocytes and T-cells show extra variance in comparison to other cell type specific clusters. Each of these clusters are divided into two distinct subclusters, explaining variance derived from differences between datasets.

4.3. Sex-specific Gene Expression Analysis

In addition to outlier samples, other common data quality issues are sample swaps (samples between two study participants have accidentally been swapped and mislabeled) and cross-contamination between samples (RNA sample from individual A has been contaminated with RNA from individual B). It is not possible to spot the sample swaps or cross contamination of samples with PCA and MDS analyses, if affected samples are the same kind (same condition and cell type). One strategy to detect this type of data quality issues is to focus on genes that are exclusively expressed by one sex. ‘t Hoen et al. (‘t Hoen et al., 2013) proposed to plot the expression of genes from the Y chromosome against the expression of the XIST gene which is only expressed in females.

We generate a scatter plot with the XIST gene (ENSG00000229807 - found only in females) expression in horizontal axis and the Y chromosome gene expression (found only in males) in vertical axis, and set the color of each sample according to its donor’s sex. Plot from the GEUVADIS study (Lappalainen et al., 2013) is a good example of non-contaminated, correctly labeled lymphoblastoid samples (Figure 4.7). In this case, all of the male samples express genes from the Y chromosome and no XIST gene, whereas female samples express XIST gene but not the genes from the Y chromosome. The perfect separation between male and female samples also demonstrate how this analysis can be used to impute sex for RNA-seq samples if this data is not present in the original dataset. When there is cross-contamination between samples, the sex-specific gene expression plot looks like the one from the BLUEPRINT (L. Chen et al., 2016) study (Figure 4.7). The blue dots (male samples) marked with circles expressed XIST gene which should only be expressed in females. Conversely, red dots (female samples) marked with circles also expressed genes from the Y chromosome which should only be present in males. This suggests that the two female samples (S003Q3B1_mRNA and S003Q3B1_RNA) have been cross-contaminated by RNA from one or more male samples and the male samples (S003P5B1_mRNA and S003P5B1_RNA) have been cross-contaminated by RNA from one or more female samples (expressing the XIST gene). This could be caused by pipetting error while the samples were being processed in the lab. Consequently, these samples should be excluded from further analysis.

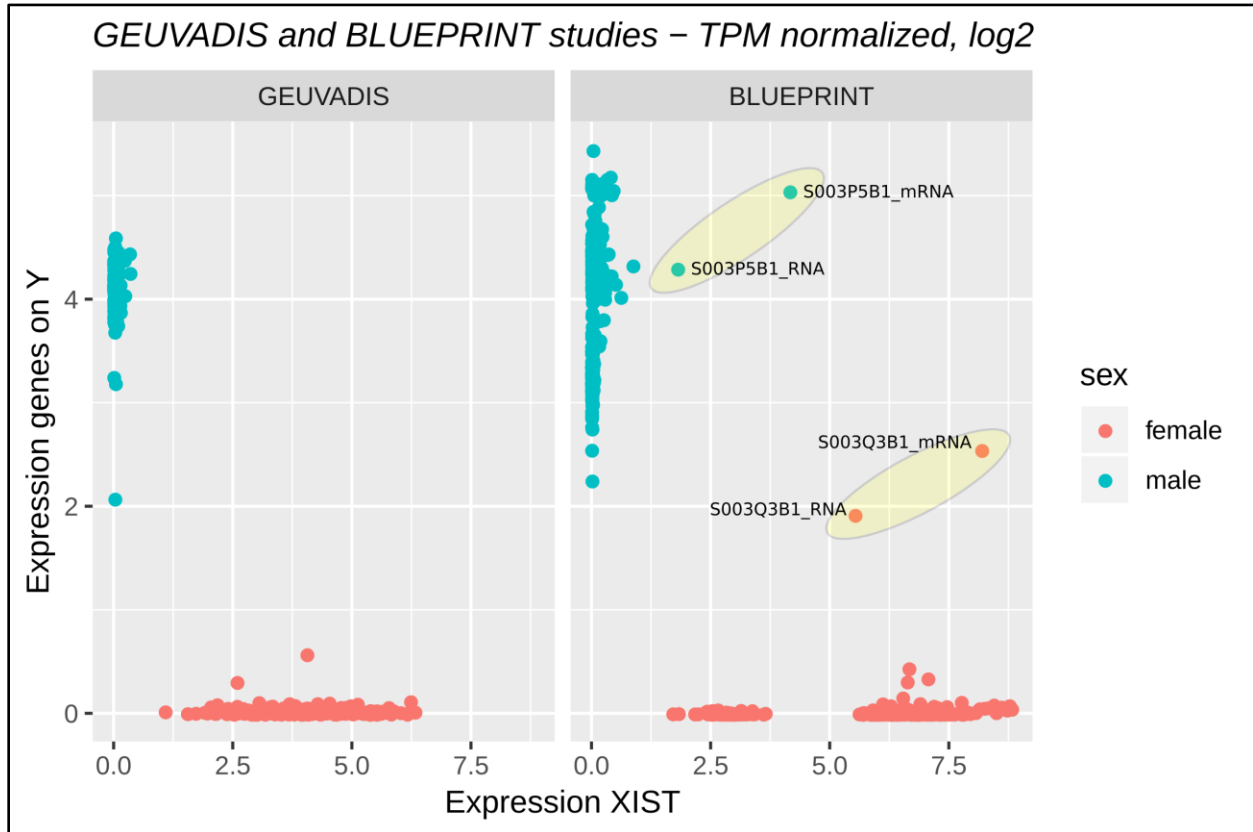


Figure 4.7. Sex-specific gene expression plots of GEUVADIS (on the left side, not contaminated) and BLUEPRINT (on the right side, contaminated) datasets

Sex-specific gene expression analysis is only effective if the metadata of dataset has sex information and the sample swap or contamination happened between samples belonging to donors with different sexes (e.g. male and female).

4.4. Sequence-genotype Matching (MBV analysis)

Another possibility to detect sample swaps is to directly compare the sample genotypes in VCF format to the genotype information that is present in the RNA-seq reads. MBV (Match BAM to VCF) is a method to detect mislabeling and technical bias in datasets that contain both genotype and sequencing (e.g. RNA-seq) data (Fort et al., 2017). This tool takes an aligned RNA-seq file (sample being tested) and a VCF file (Danecek et al., 2011) containing genotypic information of multiple samples as input and filters out undercovered variants according to provided parameter value. The variants with enough coverage are divided into two groups as homozygous and heterozygous sites. Finally, the proportion of consistent reads between the tested sample and each genotype sample in VCF file are measured for each group as shown in Formula 4.1.

The MBV tool creates one output file per each sample in the dataset. All plots shown in Figure 4.8 are generated from the MBV analysis output of Schwartzentruber et al. (Schwartzentruber et al., 2018) dataset. Each dot on the plot represents one genotype sample in the VCF file and each plot

$Count_{HomRef}$ = Number of homozygous sites REF / REF for which only REF alleles are seen at this position in the overlapping reads (1)

$Count_{HomAlt}$ = Number of homozygous sites ALT / ALT for which only ALT alleles are seen at this position in the overlapping reads (2)

$Count_{HetBoth}$ = Number of heterozygous sites REF / ALT or ALT / REF for which both REF and ALT alleles are seen at this position (3)

$$\text{The Consistency of Homozygous sites} = \frac{Count_{HomRef} + Count_{HomAlt}}{\text{Total number of homozygous sites in VCF}} \quad (4)$$

$$\text{The Consistency of Heterozygous sites} = \frac{Count_{HetBoth}}{\text{Total number of heterozygous sites in VCF}} \quad (5)$$

Formula 4.1. Calculation method of consistent reads' proportions grouped by site type (homozygous and heterozygous groups). Summarised according to supplementary material of (Fort et al., 2017) study.

generated according to analysis of one RNA-seq file (sample being tested). Colors of dots indicate if the genotype sample is a match to a tested sample. Hence, a green dot on each plot representing the best match genotype sample, and other genotype samples are red. X axis represents consistency of the heterozygous sites calculated with Formula 4.1-5. Y axis represents consistency of homozygous sites calculated with Formula 4.1-4. For example, the plot in Figure 4.8a is a good example of a non-contaminated, correctly labeled sample. The RNA-seq sample SAMEA3234534 is tested (consistency of both homozygous and heterozygous sites are calculated, Formula 4.1) against all of the genotype samples in the VCF file and only one genotype sample (HPSI0513i-oarz_22) is found to have highly consistent matching sites to the tested sample. In this example, more than 97% of homozygous sites in the tested sample (SAMEA3234534) are also homozygous in HPSI0513i-oarz_22 genotype sample. Heterozygous sites also show same level of consistency (>97%) between tested sample (SAMEA3234534) and matched genotype sample (HPSI0513i-oarz_22). Yet, consistency between the tested sample and other genotype samples in the VCF file are relatively low (consistency of heterozygous sites < 50%, consistency of homozygous sites < 80%). Therefore, it can be concluded that the tested sample has only one genotype match in the VCF file and the sample is not cross-contaminated with genetic material of other samples.

There are at least three possible conflicts between RNA-seq data and genotype data. Firstly, the genotypic data corresponding to the tested sample might not be present in the VCF file (Figure 4.8b). This technical issue sometimes happens when genotypic data (samples in VCF file) of needed samples are extracted from a larger VCF file. It should be analysed further and fixed if possible. If not possible to fix, the sample missing genotypic data should be eliminated from the dataset. Secondly, the MBV tool is good for detecting cross-contamination between samples (Figure 4.8c). The tested sample is fully matched with one genotype sample but also partially matched with some other genotype. It usually indicates possible cross-contamination in the laboratory. Both of these potentially contaminated samples should be analysed in order to decide if to eliminate one or both of the samples from the dataset. Finally, the MBV output plot can look like as Figure 4.8d. In the top right corner we see only red dot instead of green. That is because the green dot is overwritten and located under the red one, which means that the tested sample is perfectly matched with more than one genotype in the VCF file. This indicates that there are

duplicate samples in the dataset. For QTL analysis one of these duplicate samples usually needs to be excluded, because they cannot be considered as independent samples.

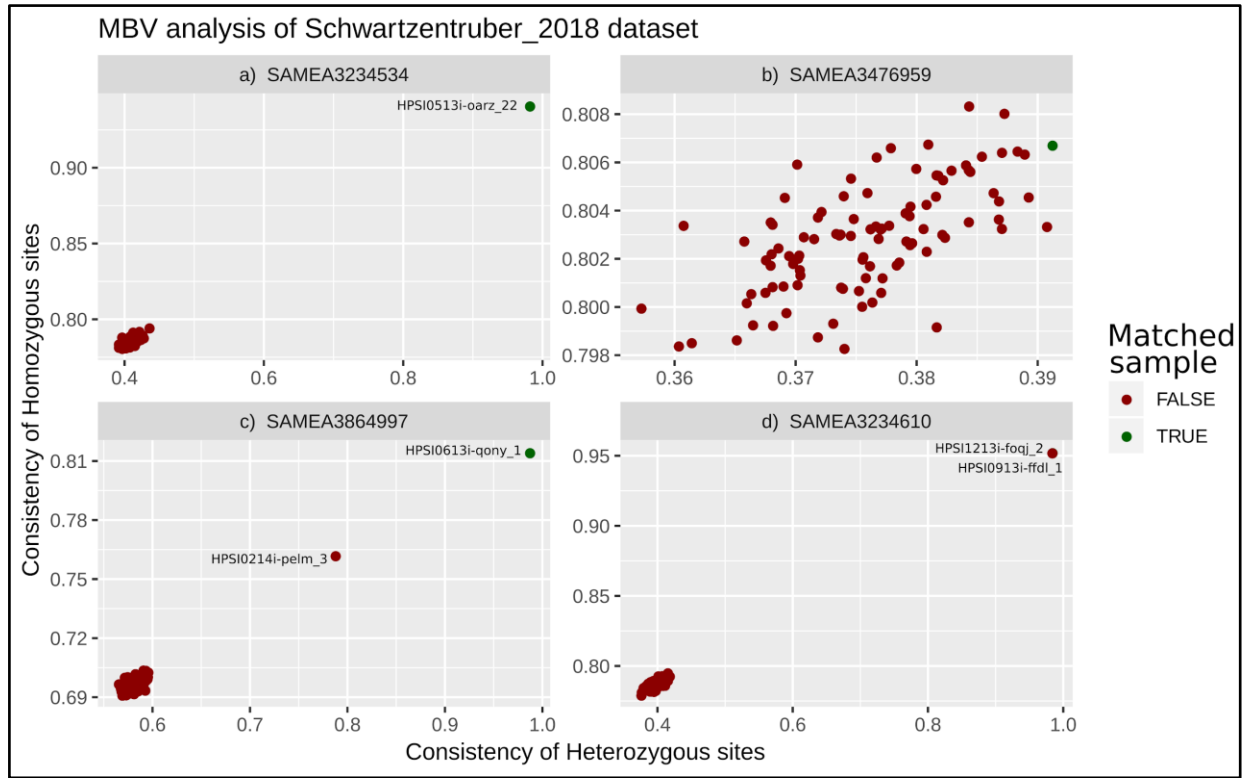


Figure 4.8. Examples of sequence-genotype matching analysis output plots from Schwartzentruber et al. (2018) dataset. X axis represents consistency of heterozygous sites (Formula 4.1-5). Y axis represents consistency of homozygous sites (Formula 4.1-4). **a)** Non-contaminated sample (SAMEA3234534) matches only one genotype (HPSI0513i-oarz_22) in the VCF file. **b)** The tested sample (SAMEA3476959) did not match to any genotype in the VCF file. **c)** Tested sample (SAMEA3864997) fully matches to one genotype (HPSI0613i-qony_1), but also partially matches another genotype (HPSI0214i-pelm_3) in the VCF file. **d)** Tested sample (SAMEA3234610) fully matches two genotypes (HPSI1213i-foqj_2 and HPSI0913i-ffdl_1) in the VCF file.

Looking through all of the plots manually can be misleading, because it is easy to miss two overlapping samples on a plot. Instead, we analysed the distance between two best matching samples to detect potential contaminated and duplicated samples. For that reason, we generate a table containing all tested samples (sample_id), best matching genotype to the tested sample (mbv_genotype_id), heterozygous and homozygous consistent fractions of the best match

Table 4.1. Example of minimum distance between best matching samples from BLUEPRINT study

sample_id	mbv_genotype_id	het_consistent_frac	hom_consistent_frac	het_min_dist	hom_min_dist	dist
S003Q3B1_mRNA	S003Q3	0.989	0.688	0.173	0.09	0.196
S003Q3B1_RNA	S003Q3	0.99	0.735	0.223	0.108	0.248
S003P5B1_mRNA	S003P5	0.99	0.775	0.347	0.142	0.374
S003P5B1_RNA	S003P5	0.991	0.835	0.414	0.169	0.447
S0026AB7	S0026A	0.996	0.89	0.494	0.231	0.545
S007PQB5	S007PQ	0.995	0.909	0.495	0.23	0.546
S0041CB7	S0041C	0.994	0.898	0.5	0.22	0.547

(het_consistent_frac and hom_consistent_frac), heterozygous and homozygous minimum distance to the second best genotype match to tested sample (het_min_dist and hom_min_dist), and overall distance from the best match to the second best match (dist) (Table 4.1). Table 4.1 represents the first 7 rows of a sequence-genotype matching analysis output table, ascendingly ordered by minimum distance between the best genotype match and the second best genotype match (dist) to the tested sample. These samples are potentially contaminated samples and need further analysis. To visualise the measurement of minimum distance between the best two matches, scatter plots of first (S003Q3B1_mRNA) and sixth (S007PQB5) samples in the table are shown in Figure 4.9.

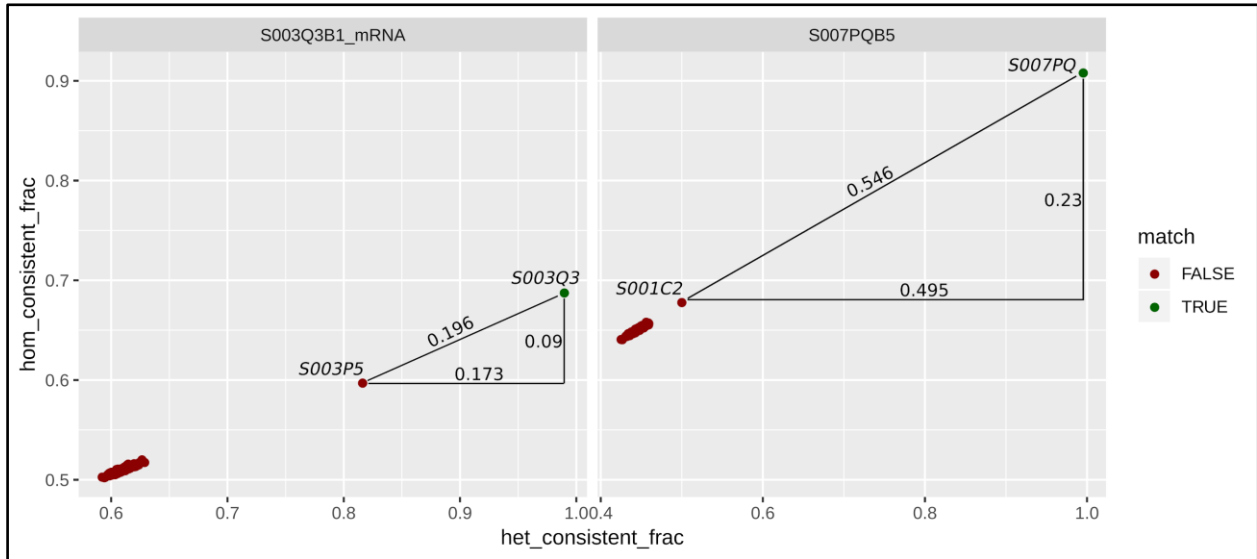


Figure 4.9. Plot of the two potentially contaminated samples' sequence-genotype matching analysis. BLUEPRINT (L. Chen et al., 2016) study. Sample S003Q3B1_mRNA is fully matched to S003Q3 genotype and partially matched to S003P5 genotype. Distance between the two best genotype matches is 0.196. Sample S007PQB5 is fully matched to S007PQ genotype and, the second best genotype match is S001C2. Distance between the two best matches is 0.546

The tested sample S003Q3B1_mRNA in Figure 4.9 is an obvious contamination and should be eliminated from the dataset. As expected this sample from the BLUEPRINT (L. Chen et al., 2016) study appeared to be contaminated also in sex-specific gene expression analysis (Figure 4.7). However, the sample S007PQB5 does not seem to be a contaminated one, because the second best genotype match (S001C2) is located close to the cluster of unmatched genotypes. This case is open to interpretation of the bioinformatician and we decided to keep this sample.

5. QTL Mapping Pipeline

Computing association summary statistics with all required technical features (portability, scalability, reproducibility etc.) is the main goal of the QTL mapping pipeline. After quality control steps are performed and samples with poor quality are eliminated, the phenotype count matrix should be normalised according to the quantification method. The normalised phenotype matrix is considered ready for QTL mapping. We developed a pipeline which takes phenotype count matrix

(the output of the quantification pipeline, quality controlled, normalised), metadata files and genotype information as input to uniformly map a wide range of molecular QTLs. The pipeline is based on the widely used QTLtools software package (Delaneau et al., 2017) and is freely available for download from GitHub³⁴.

5.1. Description of QTL Mapping Process

QTL mapping is the process of finding statistically significant associations between phenotypes and genetic variants located nearby (within a specific window around phenotype, a.k.a *cis* window), which is usually found using linear regressions. The process requires a large number of association tests to find all potential phenotype-variant associations in a *cis* window and produces p-values for each performed test. Because of the high number of existing phenotypes and the number of tests to be performed, accounting for multiple testing is essential to assess the significance of discovered associations. QTLtools (Delaneau et al., 2017) uses a fast and efficient permutation algorithm where the null distribution of association for phenotype is modeled based on the beta distribution (Ongen, Buil, Brown, Dermitzakis, & Delaneau, 2016). This enables to accurately estimate adjusted p-values in short running times.

QTLtools provides two options to run the QTL mapping in *cis* window, namely nominal run and permutation run. The nominal run calculates only nominal p-values of the associations, given the null hypothesis of not having any association between a phenotype and variant. Permutation run, on the other hand, performs a permutation based analysis in order to adjust nominal p-values according to a fitted beta distribution from thousands of random permutations of the genotype data. Permutation run accounts for the number of genetic variants tested in the *cis* window and allows us to identify phenotypes that have at least one statistically significant QTL at a pre-determined false discovery rate.

5.2. Pipeline Overview

To develop the QTL mapping pipeline, we adopted the development style and some resources of the nf-core framework (Ewels et al., 2019), as we did in quantification pipeline development. Nf-core initiated the idea of having bioinformatics pipelines in one unified pool. Nf-core pipelines are containerized, easy-to-use, open-source, tested and continuously maintained by the contributor developers. They also provide pipeline development tools³⁵ to encourage developers to contribute new pipelines and widen the community. We developed the QTL mapping pipeline (qtlmap) to be added to nf-core set of pipelines. The overview of the pipeline and brief explanation of each step is described in Figure 5.1.

³⁴ <https://github.com/kerimoff/qtlmap>

³⁵ <https://github.com/nf-core/tools>

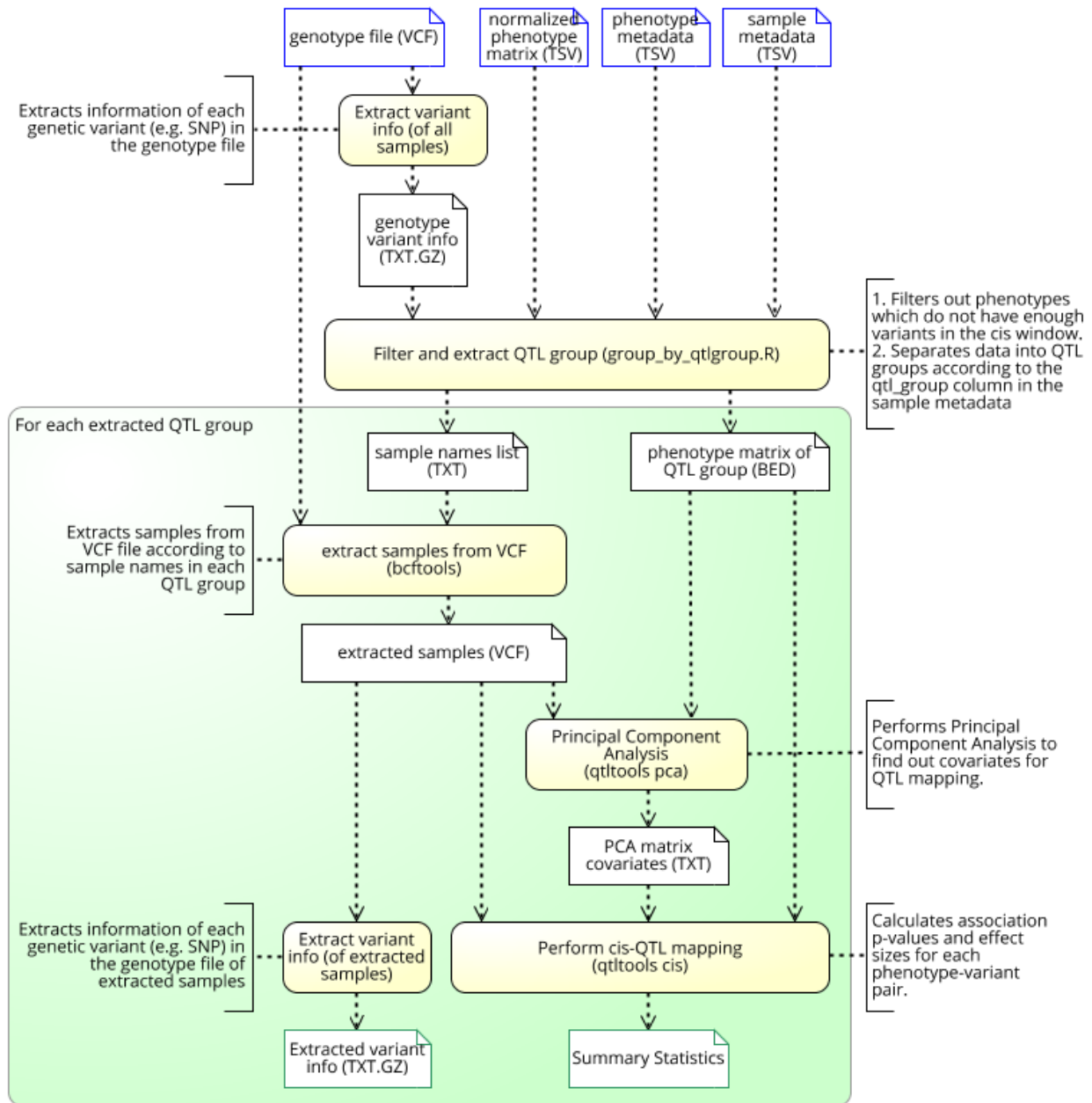


Figure 5.1. High level representation of qtlmap pipeline. Shapes with yellow background are the steps (processes) and shapes with white background symbolize data objects (files).

The pipeline starts execution with checking if all the mandatory inputs are provided (not shown in the Figure 5.1). Then, if the all provided inputs are valid, information (coordinates, alleles, etc.) about all the variants in the provided VCF file are extracted. Genotypic variant information together with sample metadata files and the phenotype count matrix are passed into the custom script *group_by_qtlgroup.R*³⁶ in order to filter out the problematic phenotypes and group samples by QTL group. After asserting that all needed columns are available in datasets, we find variants

³⁶ https://github.com/kerimoff/qtlmap/blob/master/bin/group_by_qtlgroup.R

(from variant information) overlapping with *cis* window of phenotypic features (from phenotype metadata) (Figure 5.2). We introduced a filtering parameter *mincisvariant*³⁷ (with a default value of 5) to avoid processing phenotypes with very few variants in the *cis* window, because QTLtools simply stop working (e.g. raises error) if there are no overlapping variants in *cis* window of the phenotype. Hence, by default, if the phenotype has less than 5 variants in the *cis* window, this phenotype is filtered out from the phenotype count matrix. After removing phenotypes with very few variants (less than *mincisvariant*) from the phenotype matrix, we divide it into submatrices according to *qtl_group* information provided in the sample metadata file. QTL groups usually represent different tissues, cell types or other biological contexts (e.g. experimental stimulation) present in the datasets. These submatrices are processed individually and in parallel through the next steps of the pipeline.



Figure 5.2. Finding genomic variants overlapping with the *cis* window of the phenotype. In example, *cis* distance is 1 million sequence bases and the length of *cis* window is 2 million bases. Genomic variants which overlapped with the *cis* window of the phenotype are shown in green. Genomic variants remaining out of the *cis* window represented in red color. The phenotype has 9 overlapping variants in its *cis* window, which is more than 5 (default value of *mincisvariant* parameter). Hence this phenotype will be processed in QTL mapping procedure.

Genotypic information of samples represented in each QTL group are also extracted from the VCF file, forming VCF files containing only samples belonging to each QTL group. First output of the pipeline is the variant information extracted from each of these QTL group specific VCF files. QTLtools uses PCA covariates in the QTL mapping process. PCA of the phenotype matrix is used in order to remove technical variance between samples and increases the power of detecting significant associations. On the other hand, PCA of genotypic information is used to take origin of the genotypic information (e.g. population structure) into account in the QTL mapping process. We used six principal components of the phenotype matrix and three principal components of genotypic data as covariates in QTL mapping process. In order to map the QTLs, PCA covariates matrix together with QTL group specific VCF file and phenotype matrix are provided as inputs to the QTLtools. QTLtools process these inputs in parallel according to specified number of batches³⁸. Ultimately, individually calculated outputs of batches are merged into a single summary statistics file for each QTL group.

³⁷ <https://github.com/kerimoff/qtlmap/blob/master/docs/usage.md#--mincisvariant>

³⁸ https://github.com/kerimoff/qtlmap/blob/master/docs/usage.md#--n_batches

5.3. Pipeline Implementation Details

There are a number of tools available to map QTLs such as QTLtools and MatrixEQTL (Delaneau et al., 2017; Shabalin, 2012). Moreover, the MOLGENIS (van der Velde et al. 2019) project provides a set of scripts and step-by-step guidelines to use them in order to map QTLs³⁹. Another reason why we decided to use QTLtools is that it provides options to facilitate parallelisation on compute cluster for both *cis* mapping running options (nominal⁴⁰ and permutation⁴¹ runs). Especially in permutation run this parallelisation feature increases efficacy of the analysis, because it does multiple permutation tests to adjust p-values which is computationally intensive. For instance, by default we perform ten thousand (10,000) permutations in one megabase (1 Mb) *cis* distance to obtain accurate adjusted p-values. Moreover, we also use PCA⁴² module of QTLtools to calculate covariates. Besides the preprocessing steps needed for QTL mapping, our pipeline is smoothly managing these multiple parallel tasks by processing them in specified number of batches (one parallel node per batch) and finally merging them into a single output file.

Hardware requirements of each step of the pipeline is configured in a base configuration file⁴³ and the set values are the result of personal experience of researchers involved in the project. The current version of the pipeline can only perform the *cis*-QTL mapping, however, we are working to add *trans*-QTL mapping option also in the future.

5.3.1. Containerisation and Conda support

To provide maximum level of dependency isolation, Nextflow provides very good integration with both Conda and containerisation (e.g. Docker and Singularity). We analysed all the dependencies for this pipeline and created a Conda software recipe (Figure 2.1). User can create a “Conda environment” with this recipe in an execution environment (e.g. HPC or PC) and run the pipeline without worrying about software dependencies. Nf-core provided a nice template to build a software container using a Conda environment recipe. Specifically they provide a base image which contains the operating system and Conda installed. I have built the needed containers using Dockerfile and Singularity file which are also provided by nf-core and pushed them to Docker Hub and Singularity-Hub, respectively. The address of the Docker container has been entered into pipeline settings, hence, users only need to specify if they want to run the pipeline with the container (Docker or Singularity) and the pipeline will automatically download the container from Docker Hub and use it. Since Singularity supports direct usage of Docker containers, I have included only the address of the Docker container into the pipeline.

³⁹ <https://github.com/molgenis/systemsgenetics/wiki/eQTL-mapping-analysis-cookbook-for-RNA-seq-data>

⁴⁰ https://qtltools.github.io/qtltools/pages/mode_cis_nominal.html

⁴¹ https://qtltools.github.io/qtltools/pages/mode_cis_permutation.html

⁴² https://qtltools.github.io/qtltools/pages/mode_pca.html

⁴³ <https://github.com/kerimoff/qtlmap/blob/master/conf/base.config>

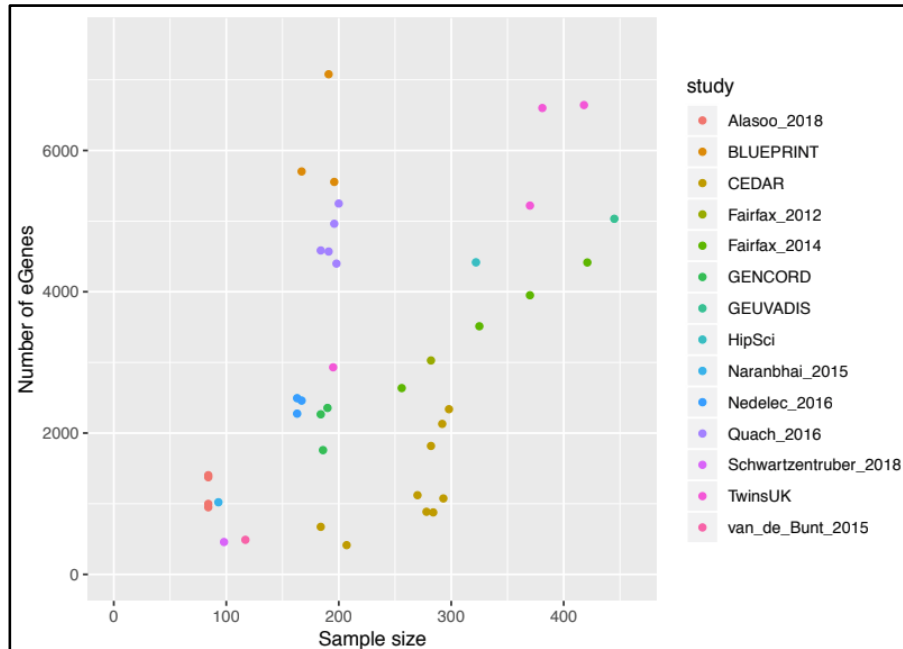


Figure 5.3. Correlation of eGenes with sample size within each QTL group. **Imputed genotypes:** 1000 Genomes Phase 3*; **Genotype filtering:** $MAF > 0.01$ & $R^2 > 0.4$; **Covariates:** 6 expression PCs + 3 genotype PCs; **Cis window:** +/- 1Mb from gene center; **FDR correction:** 10,000 permutation + Benjamini-Hochberg $FDR < 0.05$. Figure prepared by Kaur Alasoo for presentation purposes of quarterly results of the project.

We processed more than 40 QTL groups from more than 15 studies (Appendix 1) with this pipeline. To verify that the results make sense, we plotted the number of eQTLs detected in each biological context (`qtl_group`) against the sample size of that context and discovered more than 9000 eGenes (genes which have at least one significant SNP in the *cis* window) (Figure 5.3). As expected, we found a linear relationship between the number of eGenes and the sample size.

5.4. Pipeline Input and Output Description

Although the pipeline is developed to be user-friendly and ready-to-use, there is a data preparation step which requires a little effort to ensure having mandatory format and content of the input files.

5.4.1. Input Preparation

We decided to use simple text files (e.g. tab separated value, TSV) as the main format of tabular input files for our pipeline. Pipeline requires 4 input files to map the QTLs.

1. **Phenotype count matrix** of the quantitative traits: tab separated file containing normalised phenotype counts (e.g. gene expression matrix). Columns and rows represent samples and phenotypes, respectively.
2. **Phenotype metadata:** tab separated file containing metadata of phenotypes appearing in count matrix as rows. This table should have at least the following information: *phenotype_id* (this column links the metadata to count matrix), *chromosome*, *phenotype_pos*, *strand*. The chromosome and position of each phenotype is used to define the genomic region around the phenotype that is used for QTL mapping (*cis* window).

3. **Sample metadata:** tab separated file containing metadata of samples appearing in count matrix as columns. This table should have at least the following information: *sample_id* (this column links the metadata to the count matrix), *genotype_id*, *qtl_group*
4. **Variant Call Format (VCF) file:** genotype information of samples to map the QTLs to.

Input files are coupled to each other through specific parameters (Figure 5.4). Phenotype count matrix is the main data table for the pipeline. Pipeline manipulates phenotype count matrix according to metadata files and maps it to VCF file. Although, it is important to have the mandatory columns in each metadata file, preparing them is straightforward. The count matrix dataset will be grouped into subsets according to *qtl_group* information in the sample metadata file.

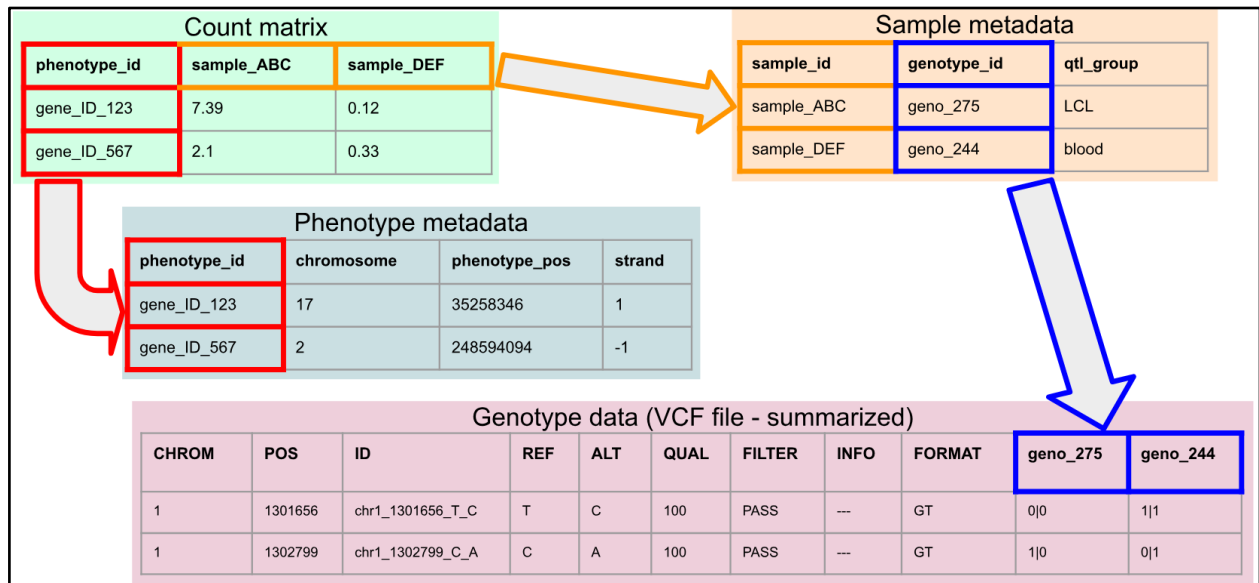


Figure 5.4. Description of relations between required input files for QTL mapping with pipeline. Count matrix has a *phenotype_id* column which corresponds to column with the same name in phenotype metadata file. Other column names of count matrix (*sample_ABC* and *sample_DEF*) corresponds to the values of *sample_id* column in sample metadata file. Values of the *genotype_id* column in sample metadata file corresponds to *genotype_ids* in VCF file. Count matrix is divided into submatrices (in this case, each of two submatrices contains only one sample) according to values of the *qtl_group* column in sample metadata file and processed individually. *chromosome*, *phenotype_pos* and *strand* information of each phenotype (in phenotype metadata file) is used to define the *cis* window that is used for QTL mapping.

For instance, TwinUK dataset contains 1364 samples originated from 4 biological contexts (in this case cell-type and tissues), namely LCL, skin, fat and blood (Table 1.2 and Appendix 1). Although the phenotype count matrix of this dataset will contain phenotype expression levels of 1364 samples, this matrix will be divided into 4 distinct matrices according to *qtl_group* value in the sample metadata file of the dataset. These 4 matrices will be processed individually in order to map the QTLs.

Finally, the genotype ids present in the VCF file should correspond to values of *genotype_id* column in sample metadata file. For a complete set of available parameters see the pipeline documentation⁴⁴.

5.4.2. Description and Interpretation of Pipeline Output

The main final output of the QTL mapping pipeline is a sorted and indexed text file of summary statistics. This file summarises the associations between phenotypes and genetic variants. The output of the nominal run contains the nominal phenotype information (id, chromosome, strand, start and end positions), genetic variant information (id, chromosome, start and end positions), test information (number of variants tested per phenotype, distance between the phenotype and associated variant) and association details (nominal p-value, effect size and a binary flag showing if the associated variant has the lowest p-value a.k.a. lead variant)⁴⁵.

6. Conclusions

In the context of the eQTL Catalog project, we implemented all the necessary steps to uniformly generate QTL summary statistics from RNA-seq and genotype data from multiple studies. The whole process can be summarised as three distinct steps:

1. Quantification of the required phenotypes (gene expression, transcript usage, exon expression and alternative splicing usage) from raw RNA-seq data (including pre-quantification quality control steps) with the quantification pipeline.
2. Post-quantification manual quality control (PCA, MDS, sex-specific gene expression and sequence-genotype matching analyses) and elimination of samples with poor quality.
3. QTL mapping pipeline input preparation and running.

In the first step we added new phenotype quantification methods (transcript usage, exon expressions and alternative splicing usage) to an existing RNA-seq pipeline developed by the nf-core framework. In order to isolate software dependencies, it fully supports package managers like Conda and container technologies such as Docker and Singularity. The output of the pipeline is a phenotype count matrix which represents expression levels of quantitative traits in the RNA-seq data.

The second step is detection and elimination of mislabeled, contaminated and poor quality samples, in order to increase the quality and precision of summary statistics. To detect outlier samples in the dataset we apply PCA and MDS analyses. For identification of mislabeling and contamination issues we analyse the sex-specific gene expression and the sequence-genotype

⁴⁴ <https://github.com/kerimoff/qtlmap/blob/master/docs/usage.md>

⁴⁵ https://qtltools.github.io/qtltools/pages/mode_cis_nominal.html

matching levels of samples. If the reason of the issue is not evident or correctable, we eliminate the concerned samples from the dataset.

After post-quantification quality control measures, the phenotype count matrix together with phenotype metadata, sample metadata and genotype (VCF) files are given into the QTL mapping pipeline in order to produce summary statistics. This pipeline is also designed to be containerised, easy-to-use, parallelly executable, scalable, open-source and ready-to-use, as all other pipelines in the nf-core framework. We implemented it to ease the QTL mapping analysis process for users and uniformly process context specific datasets of multiple studies. Although the pipeline is much simpler to use than existing QTL mapping tools, some data preparation effort is still needed. Currently, the pipeline performs only *cis* QTL mapping, but we are planning to add *trans*-QTL mapping in the near future.

The next step in development of the pipeline is consulting with the nf-core team and decide if this pipeline is suitable to be added to nf-core set of reference pipelines. Continuous integration and HTML reports are also features to be added to the pipeline. Additionally, we are looking for a suitable method to merge similar biological contexts across different studies (e.g. blood tissue samples from two different studies) and process them as a unified QTL group. Successful implementation of this approach can substantially increase the statistical power of the QTL mapping, facilitating the discovery of weaker *trans*-eQTLs, for which most individual studies are currently underpowered.

7. References

- 1000 Genomes Project Consortium, Auton, A., Brooks, L. D., Durbin, R. M., Garrison, E. P., Kang, H. M., ... Abecasis, G. R. (2015). A global reference for human genetic variation. *Nature*, 526(7571), 68–74.
- Alasoo, K., Rodrigues, J., Mukhopadhyay, S., Knights, A. J., Mann, A. L., Kundu, K., ... Gaffney. (2018). Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response. *Nature Genetics*, 50(3), 424–431.
- Allaire, J., Cheng, J., Xie, Y., McPherson, J., Chang, W., Allen, J., ... Arslan, R. (2016). rmarkdown: Dynamic Documents for R. *R Package Version, 1*, 9010.
- Anders, S., Reyes, A., & Huber, W. (2012). Detecting differential usage of exons from RNA-seq data. *Genome Research*, 22(10), 2008–2017.
- Ayoubi, T. A., & Van De Ven, W. J. (1996). Regulation of gene expression by alternative promoters. *FASEB Journal: Official Publication of the Federation of American Societies for Experimental Biology*,

10(4), 453–460.

Baggerly, K. A., & Coombes, K. R. (2009). DERIVING CHEMOSENSITIVITY FROM CELL LINES: FORENSIC BIOINFORMATICS AND REPRODUCIBLE RESEARCH IN HIGH-THROUGHPUT BIOLOGY. *The Annals of Applied Statistics*, 3(4), 1309–1334.

Brass, A. (2000). Bioinformatics education--a UK perspective. *Bioinformatics*, 16(2), 77–78.

Buil, A., Brown, A. A., Lappalainen, T., Viñuela, A., Davies, M. N., Zheng, H.-F., ... Dermitzakis, E. T. (2015). Gene-gene and gene-environment interactions detected by transcriptome sequence analysis in twins. *Nature Genetics*, 47(1), 88–91.

Buniello, A., MacArthur, J. A. L., Cerezo, M., Harris, L. W., Hayhurst, J., Malangone, C., ... Parkinson, H. (2018). The NHGRI-EBI GWAS Catalog of published genome-wide association studies, targeted arrays and summary statistics 2019. *Nucleic Acids Research*. <https://doi.org/10.1093/nar/gky1120>

Bush, W. S., & Moore, J. H. (2012). Chapter 11: Genome-wide association studies. *PLoS Computational Biology*, 8(12), e1002822.

Carillo, K., & Okoli, C. (2008). The Open Source Movement: A Revolution in Software Development. *Journal of Computer Information Systems*, 49(2), 1–9.

Chen, J., & Weiss, W. A. (2015). Alternative splicing in cancer: implications for biology and therapy. *Oncogene*, 34(1), 1–14.

Chen, L., Ge, B., Casale, F. P., Vasquez, L., Kwan, T., Garrido-Martín, D., ... Soranzo, N. (2016). Genetic Drivers of Epigenetic and Transcriptional Variation in Human Immune Cells. *Cell*, 167(5), 1398–1414.e24.

Cock, P. J. A., Fields, C. J., Goto, N., Heuer, M. L., & Rice, P. M. (2010). The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, 38(6), 1767–1771.

Cox, T. F., & Cox, M. A. A. (2000). *Multidimensional scaling*. Chapman and hall/CRC.

Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012). Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. In *Proceedings of the ACM 2012 Conference on*

- Computer Supported Cooperative Work* (pp. 1277–1286). New York, NY, USA: ACM.
- Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., DePristo, M. A., ... 1000 Genomes Project Analysis Group. (2011). The variant call format and VCFtools. *Bioinformatics*, 27(15), 2156–2158.
- Delaneau, O., Ongen, H., Brown, A. A., Fort, A., Panousis, N. I., & Dermitzakis, E. T. (2017). A complete tool set for molecular QTL discovery and analysis. *Nature Communications*, 8, 15452.
- Denk, F. (2017). Don't let useful data go to waste. *Nature*, 543(7643), 7.
- Dermitzakis, E. T. (2008). From gene expression to disease risk. *Nature Genetics*, 40(5), 492–493.
- Di Tommaso, P. (n.d.). *awesome-pipeline*. Github. Retrieved from <https://github.com/pditommaso/awesome-pipeline>
- Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35(4), 316–319.
- Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., ... Gingeras, T. R. (2013). STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1), 15–21.
- Ellis, S. E., Gupta, S., Ashar, F. N., Bader, J. S., West, A. B., & Arking, D. E. (2013). RNA-Seq optimization with eQTL gold standards. *BMC Genomics*, 14, 892.
- Emilsson, V., Thorleifsson, G., Zhang, B., Leonardson, A. S., Zink, F., Zhu, J., ... Stefansson, K. (2008). Genetics of gene expression and its effect on disease. *Nature*, 452(7186), 423–428.
- Ewels, P., Magnusson, M., Lundin, S., & Käller, M. (2016). MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics*, 32(19), 3047–3048.
- Ewels, P., Peltzer, A., Fillinger, S., Alneberg, J., Patel, H., Wilm, A., ... Nahnsen, S. (2019). *nf-core: Community curated bioinformatics pipelines*. *bioRxiv*. <https://doi.org/10.1101/610741>
- Fairfax, B. P., Makino, S., Radhakrishnan, J., Plant, K., Leslie, S., Dilthey, A., ... Knight, J. C. (2012). Genetics of gene expression in primary immune cells identifies cell type-specific master regulators and roles of HLA alleles. *Nature Genetics*, 44(5), 502–510.
- Ferragina, P., & Manzini, G. (2000). Opportunistic data structures with applications. In *Proceedings 41st*

Annual Symposium on Foundations of Computer Science (pp. 390–398).

Fjukstad, B., & Bongo, L. A. (2017). A Review of Scalable Bioinformatics Pipelines. *Data Science and Engineering*, 2(3), 245–251.

Fort, A., Panousis, N. I., Garieri, M., Antonarakis, S. E., Lappalainen, T., Dermitzakis, E. T., & Delaneau, O. (2017). MBV: a method to solve sample mislabeling and detect technical bias in large combined genotype and sequencing assay datasets. *Bioinformatics*, 33(12), 1895–1897.

Giambartolomei, C., Vukcevic, D., Schadt, E. E., Franke, L., Hingorani, A. D., Wallace, C., & Plagnol, V. (2014). Bayesian test for colocalisation between pairs of genetic association studies using summary statistics. *PLoS Genetics*, 10(5), e1004383.

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., ... Myers, J. (2007). Examining the Challenges of Scientific Workflows. *Computer*, 40(12), 24–32.

Goecks, J., Nekrutenko, A., Taylor, J., & Galaxy Team. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8), R86.

Goldstein, L. D., Cao, Y., Pau, G., Lawrence, M., Wu, T. D., Seshagiri, S., & Gentleman, R. (2016). Prediction and Quantification of Splice Events from RNA-Seq Data. *PLoS One*, 11(5), e0156132.

Goodstadt, L. (2010). Ruffus: a lightweight Python library for computational pipelines. *Bioinformatics*, 26(21), 2778–2779.

Griffith, M., Griffith, O. L., Mwenifumbo, J., Goya, R., Morrissy, A. S., Morin, R. D., ... Marra, M. A. (2010). Alternative expression analysis by RNA sequencing. *Nature Methods*, 7(10), 843–847.

Grüning, B., Dale, R., Sjödin, A., Chapman, B. A., Rowe, J., Tomkins-Tinch, C. H., ... Bioconda Team. (2018). Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nature Methods*, 15(7), 475–476.

GTEx Consortium. (2013). The Genotype-Tissue Expression (GTEx) project. *Nature Genetics*, 45(6), 580–585.

Ioannidis, J. P. A., Allison, D. B., Ball, C. A., Coulibaly, I., Cui, X., Culhane, A. C., ... van Noort, V.

(2009). Repeatability of published microarray gene expression analyses. *Nature Genetics*, 41(2), 149–155.

Kandukuri, B. R., V., R. P., & Rakshit, A. (2009). Cloud Security Issues. In *2009 IEEE International Conference on Services Computing* (pp. 517–520).

Kanwal, S., Khan, F. Z., Lonie, A., & Sinnott, R. O. (2017). Investigating reproducibility and tracking provenance - A genomic workflow case study. *BMC Bioinformatics*, 18(1), 337.

Kilpinen, H., Goncalves, A., Leha, A., Afzal, V., Alasoo, K., Ashford, S., ... Gaffney, D. J. (2017). Common genetic variation drives molecular heterogeneity in human iPSCs. *Nature*, 546(7658), 370–375.

Kim, D., Langmead, B., & Salzberg, S. L. (2015). HISAT: a fast spliced aligner with low memory requirements. *Nature Methods*, 12(4), 357–360.

Kimura, K., Wakamatsu, A., Suzuki, Y., Ota, T., Nishikawa, T., Yamashita, R., ... Sugano, S. (2006). Diversification of transcriptional modulation: large-scale identification and characterization of putative alternative promoters of human genes. *Genome Research*, 16(1), 55–65.

Köster, J., & Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19), 2520–2522.

Krueger, F. (2015). Trim galore. *A Wrapper Tool around Cutadapt and FastQC to Consistently Apply Quality and Adapter Trimming to FastQ Files*.

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1), 1–27.

Kulkarni, N., Alessandri, L., Panero, R., Arigoni, M., Olivero, M., Ferrero, G., ... Calogero, R. A. (2018). Reproducible bioinformatics project: a community for reproducible bioinformatics analysis pipelines. *BMC Bioinformatics*, 19(Suppl 10), 349.

Kurtzer, G. M., Sochat, V., & Bauer, M. W. (2017). Singularity: Scientific containers for mobility of compute. *PloS One*, 12(5), e0177459.

Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., ... International Human Genome Sequencing Consortium. (2001). Initial sequencing and analysis of the human genome. *Nature*,

409(6822), 860–921.

Lappalainen, T., Sammeth, M., Friedländer, M. R., 't Hoen, P. A. C., Monlong, J., Rivas, M. A., ... Geuvadis Consortium. (2013). Transcriptome and genome sequencing uncovers functional variation in humans. *Nature*, 501(7468), 506–511.

Leipzig, J. (2017). A review of bioinformatic pipeline frameworks. *Briefings in Bioinformatics*, 18(3), 530–536.

Leipzig, J. (2018). Computational Pipelines and Workflows in Bioinformatics. In S. Ranganathan, M. Gribskov, K. Nakai, & C. Schönbach (Eds.), *Encyclopedia of Bioinformatics and Computational Biology* (pp. 1151–1162). Oxford: Academic Press.

Lepik, K., Annilo, T., Kukuškina, V., Kisand, K., Kutalik, Z., Peterson, P., ... eQTLGen Consortium. (2017). C-reactive protein upregulates the whole blood expression of CD59 - an integrative analysis. *PLoS Computational Biology*, 13(9), e1005766.

Liao, Y., Smyth, G. K., & Shi, W. (2014). featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics*, 30(7), 923–930.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., ... 1000 Genome Project Data Processing Subgroup. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16), 2078–2079.

Liu, C., Gershon, E., & Kelsoe, J. (2017). From Gene Expression To Disease Association. *European Neuropsychopharmacology: The Journal of the European College of Neuropsychopharmacology*, 27, S416.

Li, Y., & Chen, L. (2014). Big biological data: challenges and opportunities. *Genomics, Proteomics & Bioinformatics*, 12(5), 187–189.

Li, Y. I., Knowles, D. A., Humphrey, J., Barbeira, A. N., Dickinson, S. P., Im, H. K., & Pritchard, J. K. (2018). Annotation-free quantification of RNA splicing using LeafCutter. *Nature Genetics*, 50(1), 151–158.

Mackay, T. F. C. (2009). Q&A: Genetic Analysis of Quantitative Traits. *Journal of Biology*.

<https://doi.org/10.1186/jbiol133>

Mangul, S., Mosqueiro, T., Duong, D., Mitchell, K., Sarwal, V., Hill, B., ... Blekhman, R. (2018). A comprehensive analysis of the usability and archival stability of omics computational tools and resources. *bioRxiv*. <https://doi.org/10.1101/452532>

Marioni, J. C., Mason, C. E., Mane, S. M., Stephens, M., & Gilad, Y. (2008). RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, *18*(9), 1509–1517.

Martin, M. (2011). Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal*, *17*(1), 10–12.

M. Burrows, D. J. W. (1994). A block-sorting lossless data compression algorithm. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.8069>

Melé, M., Ferreira, P. G., Reverter, F., DeLuca, D. S., Monlong, J., Sammeth, M., ... Guigó, R. (2015). Human genomics. The human transcriptome across tissues and individuals. *Science*, *348*(6235), 660–665.

Merkel, D. (2014). Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux Journal*, *2014*(239). Retrieved from <http://dl.acm.org/citation.cfm?id=2600239.2600241>

Mugavin, M. E. (2008). Multidimensional scaling: a brief overview. *Nursing Research*, *57*(1), 64–68.

Nédélec, Y., Sanz, J., Baharian, G., Szpiech, Z. A., Pacis, A., Dumaine, A., ... Barreiro, L. B. (2016). Genetic Ancestry and Natural Selection Drive Population Differences in Immune Responses to Pathogens. *Cell*, *167*(3), 657–669.e21.

Ongen, H., Buil, A., Brown, A. A., Dermitzakis, E. T., & Delaneau, O. (2016). Fast and efficient QTL mapper for thousands of molecular phenotypes. *Bioinformatics*, *32*(10), 1479–1485.

Patro, R., Duggal, G., Love, M. I., Irizarry, R. A., & Kingsford, C. (2017). Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*, *14*(4), 417–419.

Perez, F., & Granger, B. E. (2007). IPython: A System for Interactive Scientific Computing. *Computing in Science & Engineering*. <https://doi.org/10.1109/mcse.2007.53>

Piccolo, S. R., & Frampton, M. B. (2016). Tools and techniques for computational reproducibility.

GigaScience, 5(1), 30.

Quach, H., Rotival, M., Pothlichet, J., Loh, Y.-H. E., Dannemann, M., Zidane, N., ... Quintana-Murci, L. (2016). Genetic Adaptation and Neandertal Admixture Shaped the Immune System of Human Populations. *Cell*, 167(3), 643–656.e17.

Ramsköld, D., Wang, E. T., Burge, C. B., & Sandberg, R. (2009). An abundance of ubiquitously expressed genes revealed by tissue transcriptome sequence data. *PLoS Computational Biology*, 5(12), e1000598.

Ripley, B. D. (2007). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Sayols, S., Scherzinger, D., & Klein, H. (2016). dupRadar: a Bioconductor package for the assessment of PCR artifacts in RNA-Seq data. *BMC Bioinformatics*, 17(1), 428.

Schwartzentruber, J., Foskolou, S., Kilpinen, H., Rodrigues, J., Alasoo, K., Knights, A. J., ... HIPSCI Consortium. (2018). Molecular and functional variation in iPSC-derived sensory neurons. *Nature Genetics*, 50(1), 54–61.

Shabalin, A. A. (2012). Matrix eQTL: ultra fast eQTL analysis via large matrix operations. *Bioinformatics*, 28(10), 1353–1358.

Silver, A. (2017). Software simplified. *Nature*, 546(7656), 173–174.

Siva, N. (2008). 1000 Genomes project. *Nature Biotechnology*, 26(3), 256.

Stephens, Z. D., Lee, S. Y., Faghri, F., Campbell, R. H., Zhai, C., Efron, M. J., ... Robinson, G. E. (2015). Big Data: Astronomical or Genomical? *PLoS Biology*, 13(7), e1002195.

Stodden, V., Borwein, J., & Bailey, D. H. (2013). Setting the default to reproducible. *Computational Science Research. SIAM News*, 46(5), 4–6.

't Hoen, P. A. C., Friedländer, M. R., Almlöf, J., Sammeth, M., Pulyakhina, I., Anvar, S. Y., ...

Lappalainen, T. (2013). Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories. *Nature Biotechnology*, 31(11), 1015–1022.

Trapnell, C., Roberts, A., Goff, L., Pertea, G., Kim, D., Kelley, D. R., ... Pachter, L. (2012). Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nature*

Protocols, 7(3), 562–578.

van der Velde, K., Imhann, F., Charbon, B., Pang, C., van Enkevort, D., Slofstra, M., Barbieri, R., Alberts, R., Hendriksen, D., Kelpin, F., de Haan, M., de Boer, T., Haakma, S., Stroomberg, C., Scholtens, S., van de Geijn, G., Festen, E., Weersma, R. and Swertz, M. (2018). MOLGENIS research: advanced bioinformatics data software for non-bioinformaticians. *Bioinformatics*, 35(6), pp.1076-1078.

Vernables, W. N., & Ripley, B. D. (2002). *Modern applied statistics with S*. Springer, New York, New York, USA.

Visscher, P. M., Brown, M. A., McCarthy, M. I., & Yang, J. (2012). Five years of GWAS discovery. *American Journal of Human Genetics*, 90(1), 7–24.

Wagner, G. P., Kin, K., & Lynch, V. J. (2012). Measurement of mRNA abundance using RNA-seq data: RPKM measure is inconsistent among samples. *Theory in Biosciences = Theorie in Den Biowissenschaften*, 131(4), 281–285.

Wang, J., Zheng, J., Wang, Z., Li, H., & Deng, M. (2019). Inferring Gene-Disease Association by an Integrative Analysis of eQTL Genome-Wide Association Study and Protein-Protein Interaction Data. *Human Heredity*, 83(3), 117–129.

Wang, L., Wang, S., & Li, W. (2012). RSeQC: quality control of RNA-seq experiments. *Bioinformatics*, 28(16), 2184–2185.

Wang, Y., Liu, J., Huang, B. O., Xu, Y.-M., Li, J., Huang, L.-F., ... Wang, X.-Z. (2015). Mechanism of alternative splicing and its regulation. *Biomedical Reports*, 3(2), 152–158.

Wilkie, G. S., Dickson, K. S., & Gray, N. K. (2003). Regulation of mRNA translation by 5' - and 3'-UTR-binding factors. *Trends in Biochemical Sciences*, 28(4), 182–188.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1), 37–52.

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., ... Goble, C. (2013). The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research*, 41(Web Server issue), W557–W561.

Yung, C. K., Mihaiescu, G. L., Tiernay, B., Zhang, J., Gerthoffert, F., Yang, A., ... Stein, L. D. (2017).

Abstract 378: The Cancer Genome Collaboratory. *Cancer Research*, 77(13 Supplement), 378–378.

Zhernakova, D. V., Deelen, P., Vermaat, M., van Iterson, M., van Galen, M., Arindrarto, W., ... Franke,

L. (2016). Identification of context-dependent expression quantitative trait loci in whole blood. *Nature*

Genetics. <https://doi.org/10.1038/ng.3737>

8. Appendices

1. Full list of studies processed in a context of eQTL Catalog project (List is prepared by main researcher of the project Kaur Alasoo for documentation purposes). All datasets also have a “naive” stimulation, where there is no any stimulation applied. For instance, dataset of Alasoo_2018 study has 3 stimultions in addition to naive, hence it has 4 qtl_groups.

Study name	Number of donors	Number of samples	Experiment type	Cell types or tissues	Stimulations	Number of <i>qtl_group</i> 's	Publication DOI
CEDAR	322	2388	microarray	T-cell, transverse_colon, monocyte, neutrophil, platelet, rectum, B-cell, ileum		8	http://dx.doi.org/10.1038/s41467-018-04365-8
Fairfax_2012	282	282	microarray	B-cell		1	http://dx.doi.org/10.1038/ng.2205
Fairfax_2014	424	1372	microarray	monocyte	IFN24, LPS2, LPS24	4	http://dx.doi.org/10.1126/science.1246949
Kasela_2017	297	553	microarray	T-cell		1	http://dx.doi.org/10.1371/journal.pgen.1006643
Naranbhai_2015	93	93	microarray	neutrophil		1	http://dx.doi.org/10.1038/ncomms8545
Raj_2014	515	984	microarray	T-cell, monocyte		2	http://dx.doi.org/10.1126/science.1249547
Alasoo_2018	84	336	RNA-seq	macrophage	IFNg, Salmonella, IFNg+Salmonella	4	http://dx.doi.org/10.1038/s41588-018-0046-7
BLUEPRINT	197	554	RNA-seq	monocyte, neutrophil, T-cell		3	https://doi.org/10.1016/j.cell.2016.10.026

GENCORD	195	560	RNA-seq	LCL, fibroblast, T-cell		3	https://doi.org/10.7554/eLife.00523
GEUVADIS	445	445	RNA-seq	LCL		1	https://doi.org/10.1038/nature12531
HipSci	322	322	RNA-seq	iPSC		1	https://doi.org/10.1038/nature22403
Nedelec_2016	168	493	RNA-seq	macrophage	Listeria, Salmonella	3	http://dx.doi.org/10.1016/j.cell.2016.09.025
Quach_2016	200	969	RNA-seq	monocyte	LPS, Pam3CSK4, R848, IAV	5	http://dx.doi.org/10.1016/j.cell.2016.09.024
Schwartzentruber_2018	98	98	RNA-seq	sensory_neuron		1	http://dx.doi.org/10.1038/s41588-017-0005-8
TwinsUK	433	1364	RNA-seq	fat, LCL, skin, blood		4	http://dx.doi.org/10.1038/ng.3162
van_de_Bunt_2015	117	117	RNA-seq	pancreatic_islet		1	https://doi.org/10.1371/journal.pgen.1005694
Ye_2018	261	573	RNA-seq	dendritic_cell	IFN, FLU	3	http://dx.doi.org/10.1101/gr.240390.118

9. License

Non-exclusive licence to reproduce thesis and make thesis public

I, Nurlan Kerimov,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, “Designing a robust and portable workflow for detecting genetic variants associated with molecular phenotypes across multiple studies”, supervised by Kaur Alasoo.
2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons’ intellectual property rights or rights arising from the personal data protection legislation.

Nurlan Kerimov
16/05/2019