Tools for cluster analysis of data from genome-wide association studies

Role of data representation and similarity measures

Johanne Håøy Horn

Master’s Thesis Autumn 2016
Tools for cluster analysis of data from genome-wide association studies

Johanne Håøy Horn

1st August 2016
Abstract

In the past couple of decades, genome-wide association studies (GWAS) have become a widely used approach for investigating the underlying genetic architecture of complex human diseases. Each particular GWAS will highlight multiple loci across the genome, in which genotypes occur at significantly different frequencies in cases versus controls. These loci are referred to as risk variants, and are believed to have some effect on the disease under investigation.

A primary goal of GWAS is to gain insight into the underlying etiology of the diseases subject to study. In this setting, one area of interest would be understanding how and to what extent diseases and traits are related on the genomic level. Cluster analysis, a machine learning technique that aims to group similar objects within a data set, can be used for the purpose of explorative analysis of the relationships between disease-associated data. However, the results and interpretation of a cluster analysis relies considerably on choices of data representation and distance measure, the details of which have not been thoroughly discussed in the context of GWAS data sets.

This thesis presents different methods for cluster analysis of disease-associated genomic tracks. Different ways to uniformly and computationally represent GWAS data are discussed, where the biological property linkage disequilibrium is taken into account. Further, we describe objective measures for defining similarity with respect to the different data representations. The goal of these different methods is to reflect a biological, rather than technical notion of similarity between disease-associated data.

A suite of tools and methods for cluster analysis of GWAS data sets is made available through the Genomic HyperBrowser, an open-source web-based analysis platform developed and maintained by the research group for biomedical informatics at the University of Oslo. The tools aim to provide a comprehensive and transparent way to explore the extent to which diseases share genetic components identified from GWAS. Examples of utilization, with published workflows and data sets, are provided.
Acknowledgements

This thesis is the result of many months of hard work, and would not have been the same without the help and guidance of many people. First and foremost, I would like to thank my supervisors, Geir Kjetil Sandve and Sigve Nakken for your guidance, support and inspirational input throughout the thesis project.

I would also like to thank the HyperBrowser development team. A special thank you goes to Sveinung Gundersen, for your help with different aspects of the HyperBrowser development process, including in-depth explanations, debugging and merging in code from the developer version. A big thank you also goes to Boris Simovski, for code help and countless restarts of my HyperBrowser instance. Further, I would like to thank the Ensembl dev team for excellent support and help with the scripts for LD computations.

On a personal note, I am deeply grateful for the support from my friends and family. Especially the friendships made at the research group of biomedical informatics have made the past year unforgettable. A special thank you goes to Aksel Wester, Jonas Meier Strømme, Ivar Grytten and Ben Stidworthy for taking the time to read the thesis and source code, and for your suggestions of improvement.
# Contents

Abstract

Acknowledgements

List of Figures

List of Tables

Abbreviations

## 1 Introduction

1.1 Aims for thesis

1.2 Overview of chapters

## 2 Background

2.1 Unveiling the causes of complex disease

2.1.1 DNA, the building blocks of life

2.1.2 A global human reference genome

2.1.3 Genetic variation

2.1.4 Linkage Disequilibrium

2.1.5 Association of genetic variants with disease

2.1.6 The statistical and structural problem of GWAS

2.2 Computational approaches for analyzing GWAS

2.2.1 Cluster analysis definitions

2.2.2 Flat clustering

2.2.3 Hierarchical clustering

2.2.4 Linkage criterion

2.3 Review of methods for comparison of GWAS

2.3.1 Clustering using vectors of predefined bins

2.3.2 Clustering diseases based on shared genetic loci

2.3.3 Using regression and LD to find genetic correlation

2.3.4 Genetic investigation of five chronic inflammatory diseases

2.4 The Genomic HyperBrowser

2.4.1 Tools

2.4.2 Statistics

2.5 Representing genomic data

2.5.1 Genomic track data
2.5.2 GTrack representation of SNPs ........................................ 26
2.5.3 GTrack representation of linkage disequilibrium ............... 26
2.6 Collecting data for diseases and traits ............................... 27
  2.6.1 Sources of GWAS data ............................................ 27
  2.6.2 Sources of LD data .................................................. 28
  2.6.3 Creating GSuites within the HyperBrowser ...................... 29
  2.6.4 Software for a standardized summary statistic format ........... 29

3 Methods .............................................................................. 31
  3.1 Clustering ........................................................................ 31
  3.2 Binary data representations .............................................. 32
    3.2.1 Binary Taxonomic Units ........................................... 32
    3.2.2 Definitions of binary features .................................... 33
    3.2.3 Properties of the different binary representations ........... 39
    3.2.4 Similarity measures .................................................. 40
    3.2.5 Standardized measures of distance ............................... 41
  3.3 Continuous data representations ........................................ 42
    3.3.1 Definitions of continuous features ............................... 42
    3.3.2 Correlation coefficients ............................................. 43
    3.3.3 Standardized measures of distance ................................ 44

4 Implementation .................................................................... 45
  4.1 Languages and frameworks .............................................. 45
    4.1.1 The Genomic HyperBrowser ....................................... 45
    4.1.2 Python libraries ...................................................... 45
    4.1.3 Perl .................................................................. 46
  4.2 LD computation ............................................................ 46
    4.2.1 Scripts for finding variants in LD ................................. 46
    4.2.2 A master file of variants in LD ................................. 47
  4.3 Tools for data modification and creation ................................ 47
    4.3.1 Genomic liftover .................................................... 47
    4.3.2 From sumstats to GTracks ....................................... 48
    4.3.3 LD track generation .................................................. 48
    4.3.4 Expanding a GSuite with LD ..................................... 49
  4.4 Tools for empirical exploration ........................................... 49
    4.4.1 Empirical exploration of point track properties ............... 49
    4.4.2 Empirical exploration of LD tracks ............................ 49
    4.4.3 Print rsIDs of tracks in GSuite .................................. 50
  4.5 Binary feature representations ......................................... 50
    4.5.1 Single base positions .............................................. 50
    4.5.2 Bins .................................................................. 50
    4.5.3 Genetic loci .......................................................... 51
    4.5.4 Haplotype blocks ..................................................... 52
    4.5.5 Bipartite matching scores ......................................... 52
    4.5.6 Pairwise comparison ................................................. 54
  4.6 Continuous vector representations ..................................... 54
    4.6.1 Vectors of values at overlapping positions .................... 55
    4.6.2 Vectors of overlap with other tracks ......................... 55
4.6.3 Pairwise comparison ........................................ 55
4.7 Disease-by-disease comparison ................................. 56
  4.7.1 Triangular matrix creation ............................... 56
  4.7.2 Heatmap .................................................. 56
  4.7.3 Clustering ............................................... 57
  4.7.4 Tool overview .......................................... 57

5 Results .......................................................... 59
  5.1 A suite of tools for comparison of diseases ................. 59
    5.1.1 Main purpose ......................................... 60
    5.1.2 Tools for clustering of binary representations ....... 60
    5.1.3 Tools for clustering of continuous vectors .......... 61
    5.1.4 Tools for empirical exploration ....................... 61
    5.1.5 Tools for data modification and creation ............ 62
  5.2 Use case .................................................... 62
    5.2.1 GSuite creation ...................................... 63
    5.2.2 Exploration of data properties ......................... 63
    5.2.3 Comparison of significant SNPs ....................... 64
    5.2.4 Comparison of GWAS summary statistics .............. 64

6 Discussion ...................................................... 69
  6.1 Data representation and resources .......................... 69
    6.1.1 Data formats ......................................... 69
    6.1.2 Software for LD computation .......................... 69
    6.1.3 Potential data confounders ............................ 71
  6.2 Methods related to cluster analysis ........................ 72
    6.2.1 Feature representations ................................ 73
    6.2.2 Similarity and correlation ............................ 74
    6.2.3 Distance conversion ................................... 75
    6.2.4 Clustering ............................................ 76
  6.3 Design principles .......................................... 76
    6.3.1 Reproducibility ....................................... 76
    6.3.2 Transparency ........................................ 77
    6.3.3 Usability ............................................. 78
  6.4 Different methods for comparison of diseases ............... 78
    6.4.1 Pleiotropy versus genetic correlation .................. 78
    6.4.2 Related methods ....................................... 79
  6.5 Developed software ......................................... 80
    6.5.1 Source code ........................................... 80
    6.5.2 Code integration in the HyperBrowser ................. 81
    6.5.3 An alternative path for development ................... 82
  6.6 Weaknesses in implementation ................................ 83
    6.6.1 No automated tests .................................... 83
    6.6.2 Use of static data files ............................... 83
    6.6.3 Time and space complexity ............................. 84

7 Conclusion and future work .................................. 85
  7.1 Conclusion ................................................ 85
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2 Future work</td>
<td>85</td>
</tr>
<tr>
<td>Appendices</td>
<td>87</td>
</tr>
<tr>
<td>A Linkage criteria</td>
<td>89</td>
</tr>
<tr>
<td>B Data sources and formats</td>
<td>91</td>
</tr>
<tr>
<td>B.1 NCBI: dbSNP</td>
<td>91</td>
</tr>
<tr>
<td>B.2 NHGRI-EBI GWAS Catalog</td>
<td>91</td>
</tr>
<tr>
<td>B.3 LD tracks</td>
<td>92</td>
</tr>
<tr>
<td>C Source code</td>
<td>97</td>
</tr>
<tr>
<td>C.1 GitHub repository</td>
<td>97</td>
</tr>
<tr>
<td>C.2 LD scripts</td>
<td>97</td>
</tr>
<tr>
<td>D Analyses and experimental data sets</td>
<td>99</td>
</tr>
<tr>
<td>D.1 Use case of seven diseases and traits</td>
<td>99</td>
</tr>
<tr>
<td>D.1.1 Data from the GWAS Catalog</td>
<td>99</td>
</tr>
<tr>
<td>D.1.2 Summary statistics</td>
<td>99</td>
</tr>
<tr>
<td>D.2 Example data for 40 diseases and traits</td>
<td>100</td>
</tr>
<tr>
<td>E Using the Ensembl Variation API</td>
<td>109</td>
</tr>
<tr>
<td>E.1 Setup and installation on OS X</td>
<td>110</td>
</tr>
<tr>
<td>E.1.1 Prerequisites</td>
<td>110</td>
</tr>
<tr>
<td>E.1.2 Core dependencies</td>
<td>111</td>
</tr>
<tr>
<td>E.1.3 DBD::mysql</td>
<td>111</td>
</tr>
<tr>
<td>E.1.4 Cloning the Ensembl repos</td>
<td>111</td>
</tr>
<tr>
<td>E.1.5 Compilation</td>
<td>113</td>
</tr>
<tr>
<td>E.1.6 Setting the environment</td>
<td>113</td>
</tr>
<tr>
<td>E.1.7 Check installation status</td>
<td>113</td>
</tr>
<tr>
<td>E.2 Scripts and local setup</td>
<td>114</td>
</tr>
<tr>
<td>E.2.1 API calls without FTP connection</td>
<td>114</td>
</tr>
<tr>
<td>E.3 Challenges with the Ensembl APIs</td>
<td>115</td>
</tr>
<tr>
<td>E.3.1 Running the scripts</td>
<td>115</td>
</tr>
<tr>
<td>E.3.2 Implementation and documentation</td>
<td>117</td>
</tr>
</tbody>
</table>
## List of Figures

2.1 Structure of DNA in the chromosome ........................................... 4  
2.2 Examples of genetic variations .................................................. 6  
2.3 Linkage disequilibrium for different thresholds of $r^2$ ................. 9  
2.4 SNP structure within and between populations ............................ 10  
2.5 Two examples of distance measures in vector space ................... 14  
2.6 Examples of hierarchical clustering dendrograms ..................... 15  
2.7 Linkage criteria for hierarchical clustering ............................... 16  
2.8 Correlation of genetic loci between diseases and traits ............... 19  
2.9 Genetic correlation between diseases and traits ....................... 21  
2.10 Code example for a HyperBrowser statistic ............................. 23  
2.11 Code example for a tool execution ........................................ 25  
2.12 Graph visualization of linkage disequilibrium .......................... 27  

3.1 Matching of single base positions .......................................... 33  
3.2 Matching of bins ................................................................. 34  
3.3 Difference in matching for three feature representations ............. 35  
3.4 Matching of haplotypes ......................................................... 36  
3.5 Bipartite graph representation ............................................... 37  
3.6 Bipartite matching of SNPs ..................................................... 39  

5.1 Example output from tools for empirical exploration ................... 63  
5.2 Matching of haplotypes with cosine ....................................... 65  
5.3 Bipartite matching with McConnaughey .................................. 65  
5.4 Matching of 500k bins with Forbes ...................................... 65  
5.5 Correlation of p-values ......................................................... 66  
5.6 Correlation of z-values ......................................................... 66  

6.1 Distance between variants in high linkage disequilibrium ............ 74  

D.1 Dendrogram for feature representation of bins .......................... 103  
D.2 Dendrogram for features of genetic loci overlap ........................ 103  
D.3 Heatmap for feature representation of bins .............................. 104  
D.4 Heatmap for feature representation of haplotypes ...................... 104  
D.5 Heatmap for feature representation of matching scores ............... 105  
D.6 Heatmap for features of genetic loci overlap ............................ 105  
D.7 Heatmap with Jaccard similarity ........................................... 106  
D.8 Heatmap with Simpson similarity ......................................... 106
D.9 Heatmap with McConnaughey similarity .......................... 107
List of Tables

2.1 GTrack specifications ................................................. 26
2.2 Required headers in the sumstat file format .................. 30

3.1 Expression of BTU summaries ...................................... 32
3.2 Properties of the different binary feature representations . 39

4.1 Example of cost matrix for bipartite matching ............... 53
4.2 Overview of feature representations used in tools ........... 58

B.1 Genome build for dbSNP data ..................................... 91
B.2 Attributes related to a study in the GWAS Catalog .......... 92
B.3 Attributes related to a SNP in the GWAS Catalog .......... 93
B.4 Example of GWAS SNP track element ......................... 94
B.5 LD track locations .................................................... 95

D.1 Galaxy histories for use case ...................................... 100
D.2 Original and filtered SNP counts for use case data .......... 101
D.3 Overview of sources for GWAS summary statistics .......... 102

E.1 Overview of documentation for LD computations ............ 110
E.2 Location of 1000G VCF files for genotype data ............... 114
Abbreviations

**1000G**  The 1000 Genomes Project
**bp:**  base pair
**BTU:**  Binary Taxonomic Unit
**DNA:**  deoxyribonucleic acid
**GWAS:**  Genome-Wide Association Study
**GWAS Catalog:**  the NGRHI-EBI GWAS Catalog
**HapMap:**  The International HapMap Project
**hg19:**  genome build GRCh37
**hg38:**  genome build GRCh38
**kb:**  kilobases, a thousand base pairs
**LAPJV:**  Jonker-Volgenant Algorithm for Linear Assignment Problem
**LD:**  linkage disequilibrium
**NCBI:**  National Center for Biotechnology Information
**Mb:**  megabases, a million base pairs
**rsID:**  reference SNP cluster ID
**SNP:**  single nucleotide polymorphism
**SNV:**  single nucleotide variant
Chapter 1

Introduction

The human genome, with its billions of base pairs, is the genetic foundation for all human biology. Within it lies information of our medical past, present and future. Chance, hereditary history and lifestyle choices result in unique combinations of variation in different genomes, so that no two individuals are exactly identical on the genomic level. Carriers of a certain disease, however, might have genetic variants in common that explain the cause and course of the disease in question.

Large-scale association studies conducted on the genome-wide level can be used to find patterns of common genetic variants in groups of individuals that carry a disease. These associated variants are hypothesized to have an effect on the traits or disease under investigation, but their function and importance are often unknown. Explorative investigation of relationships between different disease-associated data sets might in this context reveal shared genetic components, which could be subject to further in-depth analysis.

1.1 Aims for thesis

The goal of this thesis is to create a suite of tools for comparative analysis of diseases and traits, as represented by GWAS data sets. Clustering, an unsupervised machine learning algorithm, is the chosen method for comparing disease-associated data.

The focus will be on different methods for representation of GWAS data, and how similarity further can be defined for a given representation. The aim of these methods is to capture a biological, rather than technical notion of similarity between disease-associated data, so that cluster analysis can be used to discover relationships of true biological meaning.
1.2 Overview of chapters

In Chapter 2, we present different biological concepts that are important for this thesis, in addition to an introduction to cluster analysis and a review of articles which present methods for comparisons of disease-associated data. The Genomic HyperBrowser framework, through which the methods of this thesis have been made available, is also described. We also present ways to represent, collect and store genomic data sets. In Chapter 3, we present a range of methods for representation of disease-associated data. For these representations, measures of similarity and correlation are described, and how they can be converted to distance for use in a cluster analysis. Implementation choices and code for the developed tool suite is described in Chapter 4, and the resulting software presented in Chapter 5. Chapter 5 also contains a use case demonstrating functionality of tools and examples of output. Chapter 6 contains discussions regarding data, the methods presented for use in the cluster analysis, and the tool implementation. Finally, Chapter 7 presents the conclusion and possible future work. Detailed explanations of data sources and formats, as well as some example figures and link to the source code are given in the Appendices. An in-depth description of setup and use of the Ensembl API is also included as appendix.
Chapter 2

Background

2.1 Unveiling the causes of complex disease

Since the discovery of DNA in 1869, great advances in the field of genetics have been made. In particular, the last couple of decades have seen an unprecedented development of new biotechnological tools which have contributed to an improved understanding of human biology and complex disease. This section gives a brief introduction to discoveries within the fields of human genetics and genomics, that together form the foundation upon which this thesis is built.

2.1.1 DNA, the building blocks of life

The story of DNA begins with a discovery made by the Swiss physician Friedrich Miescher in 1869, ten years after Charles Darwin published his famous book On the origin of species. While studying a subtype of white blood cells, Miescher came across a novel nuclear substance, which he named nuclein. Certain of its importance, he concluded his publication with the following: “Knowledge of the relationship between nuclear substances, proteins and their closest conversion products will gradually help to lift the veil which still utterly conceals the inner processes of cell growth” (Dahm, 2005). However, nearly a century passed before interest in nuclein, later identified as deoxyribonucleic acid (DNA), became widespread. The majority of scientists at that time believed DNA was too simple for storing all the genetic information of organisms. This changed after Avery et al. in 1944 and Hershey and Chase in 1952 demonstrated that DNA indeed was the carrier of genetic information. Their research was followed up by the deduction of the double helix structure of DNA in 1953 by Watson and Crick, all of which provided important insights in how DNA works (Dahm, 2005; Watson and Crick, 1953).

DNA consists of two strands, or chains, of small molecules called nucleotides. The strands form a double helix, where the nucleotides on the
2.1. UNVEILING THE CAUSES OF COMPLEX DISEASE

same position at each strand are paired together through hydrogen bonds. Nucleotides are often referred to as bases, from the nitrogenous base of the nucleotide in question, and the nucleotides paired together form base pairs. There are four different types of nitrogenous bases in DNA, and thus four types of nucleotides: adenine, guanine, cytosine, thymine, denoted by A, G, C and T, respectively. The bases on the opposing strands are complementary paired: A with G and C with T (Sung, 2010). See Figure 2.1 for an illustration of the DNA helix and its base pairs.

DNA is organized in structures called chromosomes (Figure 2.1). Together, all chromosomes in an organism, i.e. all of its DNA, constitute what we call the organism’s genome. The human genome consists of more than 3 billion base pairs, representing all the genetic material in a human being, and is stored in its entirety as a local copy in each of our cells. Humans are diploid and carry two copies of each chromosome, one from each parent, forming 23 chromosome pairs.

Some parts of the DNA, called coding regions, can be transcribed into proteins. The DNA sequence that forms the basis for this kind of functional and physical unit, is called a gene, and different tissues in the human body, such as hair and teeth, come from different genes being expressed in the cells. Despite the importance of protein-coding DNA regions in direct gene regulation, they make up less than 3% of the genome (Sung, 2010). The rest of the genome, often referred to as non-coding, does not directly encode proteins, but can still have functional roles. It is predicted that ~80% of the genome have some biochemical function, most of which is located outside the protein-coding regions (The ENCODE Project Consortium, 2012). Specific details concerning these functional elements, such as how and where they function, is yet to be discovered.

The terms locus and allele are commonly used when discussing the properties of DNA. A locus, plural loci, refers to a location in a specific pair of chromosomes in the genome. It can refer to both a single position or a larger continuous region on the chromosomes. In humans, a locus will be occupied by at least two alleles, one on each chromosome, but the

Figure 2.1: Structure of DNA in the chromosome

Illustration: ©Johan Jarnestad / The Royal Swedish Academy of Sciences
definition is usually applied when we want to refer to one of two genetic variants (Sung, 2010).

### 2.1.2 A global human reference genome

In 2001, the International Human Genome Sequencing Consortium released the first draft of a human genome assembly, which was further improved upon and reported as nearly complete in 2004. Although incomplete, it served as a basis for understanding the genetic instructions of human physiology and evolution (The International Human Genome Sequencing Consortium, 2004). Acknowledging the need of a high quality reference genome, The Genome Reference Consortium (GRC) was formed to address the issues of the initial human genome assembly. Their efforts have lead to several releases of a regularized, publicly available and globally used human reference genome (Church et al., 2011).

The latest major reference genome release, introduced in December 2013 by GRC, and last updated in October 2015, is the genome build GRCh38, often referred to as hg38. However, many research initiatives still use the GRCh37 or hg19 reference genome, released in 2009. The biggest difference between hg19 and hg38, is improvements of erroneous and complicated assembly regions, which were discovered in research initiatives subsequent to the hg19 assembly. A number of bases from hg19 never seen in any individuals were updated, and several gaps were reduced or closed. Both hg19 and hg38 contain alternate loci, i.e. alternate sequence representations for genomic loci, aligned on the reference genome, but there are more such sequences in the hg38 assembly. From a computational perspective, the biggest practical difference between hg19 and hg38, is changes in the chromosome coordinates. If different data sets contain genetic and functional elements that are interchangeably mapped to both hg19 and hg38, the elements need to be lifted over to the same reference genome before they can be used in the same analysis.

With the advent of next-generation sequencing (NGS) in 2005, exploration of large scale DNA sequences in a cost-efficient and fast manner became possible. The technology has played an important part in the reference genome assemblies that followed the initial 2004 release. Using massive parallel sequencing, NGS allows an entire genome to be sequenced in less than one day (Grada and Weinbrecht, 2013). Together with the reference genome, it becomes not only possible to assemble the genome of a person within reasonable time and cost, but also to identify the genetic variation of individuals within an entire population.

---

2[https://genomeref.blogspot.no/2013/12/announcing-grch38.html](https://genomeref.blogspot.no/2013/12/announcing-grch38.html)
2.1. UNVEILING THE CAUSES OF COMPLEX DISEASE

2.1.3 Genetic variation

The genomes of any two human beings will differ in millions of ways due to a range of genetic variants. This variation reflects the evolution and architecture of the human genome through generations. A mutation refers to the event when a locus is altered within a single cell during DNA replication, and the origin of genetic variants, which appears in many forms. The most common type of DNA variation is single nucleotide variants (SNVs), where a single nucleotide is altered. All other variants than SNV are in this thesis broadly referred to as structural variants, including insertion-deletions, block substitutions, inversions of DNA sequences and copy number differences (Frazer et al., 2009). Figure 2.2 illustrate some forms of variants.

Genetic variants can be divided into two groups: common and rare. Common variants are interchangeably referred to as polymorphisms. In order to distinguish a polymorphism from a rare variant, the terms minor allele and minor allele frequency (MAF) are used. The minor allele is the least common allele of a given genetic variant, and the MAF its frequency in a given population (Frazer et al., 2009). Different thresholds of MAF have been used to define whether a variant is common or not, typically with a lower bound of 1 or 5%. A SNV is called a single nucleotide polymorphism (SNP) if it occurs commonly in the population, defined as a MAF >1% in this thesis.

SNPs represent the most prevalent type of genetic variants together with short insertion-deletions, accounting for >99.9% of the variants found in a typical genome (The 1000 Genomes Project Consortium, 2015). For the majority of variants in the genome, their relationship to a particular disease or phenotypic trait, i.e. observable characteristics, remains unknown. Many such variants of unknown effect are hypothesized not to affect the individual carrying them (Frazer et al., 2009). A significant set of variants, however, are found to have clear functional implications, and can influence the physical appearance of an individual, such as skin and hair color. Presence of specific variants, either alone or in combinations, might also contribute to susceptibility of complex and common diseases (Shastry, 2009).

Several projects, often the result of international collaborations, have
aimed to find and describe patterns of common genetic variation, in effect identifying SNPs. One of the best known, the International HapMap Project, HapMap for short, which started in 2002 and concluded its final phase with a data release in 2009, set out to catalog all common genetic variants across the human genome (The International HapMap Consortium, 2003). The presence of a particular allele in an individual was determined by genotyping DNA samples, which in term was done for all individuals in the testing population. Variants with a MAF above 5% was defined as common, meaning that if more than 5% of the population was observed with the particular allele, it was denoted as a SNP. The HapMap Project made use of correlation between SNP genotypes, more specifically a statistical property of non-random association called linkage disequilibrium (LD), further described in Section 2.1.4. Carefully selected SNPs distributed across the entire genome were the subject of investigation in different individuals. These SNPs, identified as tag SNPs, were estimated to account for most human variation, as they represent commonly occurring combinations of alleles along a chromosome, referred to as haplotypes. The resulting data sets, including variants, frequencies, genotypes and haplotypes, were made publicly available (The International HapMap Consortium, 2005; Buchanan et al., 2012).

The 1000 Genomes Project (1000G), which started in 2008, also set out to provide a comprehensive resource on human genetic variation. Their expressed goal was to “provide a deep characterization of human genome sequence variation as a foundation for investigating the relationship between genotype and phenotype” (The 1000 Genomes Project Consortium, 2010). 1000G was one of the first projects to use NGS to sequence the genomes of a large number of people, and made use of high-density SNP microarrays to estimate genotypes and haplotypes. 1000G had a lower threshold of 1% MAF to identify polymorphisms. Overall, the project discovered, genotyped and phased 88 million variant sites, including structural variants, validating or contributing to 80 million of the 100 million known SNPs in the public dbSNP catalogue. 1000G also discovered ~64 million rare variants with MAF < 0.5% (The 1000 Genomes Project Consortium, 2010; The 1000 Genomes Project Consortium, 2015). The 1000 Genomes Project started with three pilot studies, which were completed in 2009. The main project that followed was split into three phases, with the results of phase 3 published in 2015. The data from 1000G has been used to make improvements in the hg38 genome assembly, compared to hg19, as mentioned in Section 2.1.2. Despite completion of its final planned phase in 2015, 1000G is an ongoing project, currently supported and extended by the International Genome Sample Resource. The 1000G data is seen as state of the art for genome-wide and worldwide genotype frequencies, in contrast to the more outdated HapMap phase 3 data sets.

In the HapMap Project, 269 individuals with ancestry from four populations in Africa, Asia and Europe were used.

In the 1000 Genomes project, 2,504 individuals from 26 populations worldwide were used.
2.1. UNVEILING THE CAUSES OF COMPLEX DISEASE

2.1.4 Linkage Disequilibrium

Linkage disequilibrium (LD), briefly mentioned in the previous section, is an important concept in the context of genetic variants. LD is the event when different loci are inherited together more often than would be expected by chance, i.e. “a non-random association of alleles at two or more loci” (Slatkin, 2008). The original definition of LD allowed for the possibility of loci in LD on different chromosomes, but the prevailing interpretation of LD is within chromosomes, even though LD can be caused by functional interaction where combinations of alleles on different chromosomes increase fitness in potential offspring. The definition of LD is usually applied to loci on the same chromosome, using a definition in terms of haplotype frequencies (Slatkin, 2008).

LD is given by \( r^2 \) or \( D' \), both measures of non-random association. The first is preferred when the goal is to predict co-occurring SNPs, while the latter is commonly used to predict recombination patterns. \( r^2 \) is the chosen metric in this thesis when LD is involved, where a value of 0 implies statistical independence, and 1 means perfect LD and co-occurring SNPs. In order to use \( r^2 \) to find variants in LD with a chosen SNP, a threshold must be defined, as low values indicate uncertainty of whether or not a SNP can be predicted from another. A limit of \( r^2 \geq 0.8 \) seems to be the consensus threshold in the research community\(^5\). Values down to 0.7 have also been observed in use, for instance the filter threshold of 0.75 used by So, Li, and Sham, (2011). See Figure 2.3 for a visualization of how increasing values of \( r^2 \) affect the distribution of physical distance between pairs of SNPs in LD.

It is important to note that the LD definition of \( r^2 \) is not the commonly used coefficient of determination from statistics, but rather a biological association coefficient calculated from allele and haplotype frequencies. Unfortunately, the linkage disequilibrium term can be misleading. With linkage as a part of its name, one can easily assume the phenomenon to be that of genetic linkage, the tendency of alleles located close together on a chromosome to be inherited together. Recombination does not happen by mixing each single base in the maternal and paternal DNA randomly, but rather by combining larger continuous segments on the chromosomes from each parent. Both linkage and LD are highly influenced by recombination patterns, but LD is a statistical property, which by definition disregard information of location and distance to other alleles on the chromosome.

It is important to note that LD may occur even if two loci are not genetically linked. Reversely, linked loci are not guaranteed to be in linkage disequilibrium, although they frequently are. The relationship between physical distance and LD can be empirically observed (Figure 2.3), and the amount of variants in LD typically drop dramatically as distance between them increase.

A set of SNP genotypes on the same chromosome that tend to occur

\(^5\)https://www.researchgate.net/post/What_is_an_acceptable_r2_threshold_for_a_proxy_SNP
CHAPTER 2. BACKGROUND

Figure 2.3: Visualization of linkage disequilibrium (LD) between variants, as a function of their physical distance, using different thresholds of $r^2$. The data plotted is LD variants computed for all significant GWAS Catalog SNPs found in the HyperBrowser as explained in Section 2.6.3, and each line represents a different threshold of $r^2$.

together in different individuals, i.e. a set of variants in LD, constitute a haplotype, also referred to as an LD block. LD thus makes it possible to account for most variation within a population by only a few haplotypes in particular regions, and a tag SNP serve as representative for all variants within the same haplotype. Choosing tag SNPs for genotyping leads to a substantial reduction in the amount of genetic variants that needs to be examined. From the HapMap Project data, it was estimated that the vast majority of SNPs with MAF >5% could be reduced to ~550,000 haplotypes for European and Asian populations, and tag SNPs for these LD blocks would account for >80% of the SNPs present in the genotyped individual (Frazer et al., 2009).

Due to differences in evolutionary forces around the world, different populations do not necessarily share haplotype structure, or even harbor the same common variants. This can be seen in Figure 2.4, where co-occurrence of SNPs within and between populations are shown. As a consequence, two variants that are in LD in one population, might not be in a different population. This is an important consideration in studies where genetic variants are associated with disease, the subject of the next couple of sections. Another property worth noting is that LD blocks within a population might vary in size due to differences in recombination events and recombination hotspots (The International HapMap Consortium, 2005; Yu et al., 2005)

2.1.5 Association of genetic variants with disease

Some diseases are easy to characterize in terms of genetics. Common for these are that they are rare, highly heritable, and variation in a single gene is the necessary and sufficient cause for their susceptibility (The International HapMap Consortium, 2005). Attempts to identify genetic contributors
2.1. UNVEILING THE CAUSES OF COMPLEX DISEASE

Both HapMap and 1000G resulted in improved technical platforms and haplotype and genotype information that have facilitated discovery of LD patterns (Buchanan et al., 2012). As a result, researchers could cost-efficiently genotype the genomes of individuals, leading to the advent and prevalence of Genome-Wide Association Studies (GWAS). SNPs occur on the population-wide level, with a certain percentage of the population harboring it, and can be used to associate genetic variation with diseases and phenotypic traits. In GWAS, at minimum hundreds of thousands of tag SNPs are assayed to identify associations with a certain disease or trait, narrowing down which loci might influence it. The DNA of a case and a control group is compared, where the case group harbor the trait or disease the researchers wish to examine, and the control group are a similar subset of the population, only without the trait. If a particular genotype occurs with much greater frequency in the diseased group, it is said to be associated with the disease, and denoted as a risk allele. Millions of genotyping tests are performed, and results must be adjusted to control for false positives. Consequently, the standard significance threshold of associated SNPs is set to $5 \times 10^{-8}$ (Gibson, 2012).

The region for which an associated SNP is located is hypothesized to influence the risk of disease, but as the tag SNP represents other variants in LD, it cannot necessarily be identified as causal from the GWAS alone. One or multiple variants in LD might be the true contributors of risk for the disease in question (Tak and Farnham, 2015).

In the context of GWAS, the effect size of a reported SNP indicates to what
CHAPTER 2. BACKGROUND

extent the variant influences the disease or trait under investigation. The effect size is the magnitude of the difference in MAF between the case and control group (Sullivan and Feinn, 2012). The p-value of a reported SNP, on the other hand, has no information on effect size, but instead indicate how certain we can be that an effect exists. P-values express significance of the reported association, and are computed from a null hypothesis of no effect. A consequence of SNPs with small effect sizes, is that a large sample size is required to get a statistically significant association.

2.1.6 The statistical and structural problem of GWAS

An important assumption that lies behind the study design of GWAS is the common variants–common disease (CV–CD) theory (Gibson, 2012). It hypothesizes that combinations of a few common risk variants, as found by GWAS, together cause the disease with which they are associated. The theory postulates that no variant will be sufficient to cause the disorder by itself, as the effect of a single SNP alone is too small to confer any significant degree of risk.

Despite the large number of GWAS that have been conducted and the resulting SNPs associated, a number of diseases and traits are still poorly characterized in terms of their underlying genetic etiology, i.e. origin or cause. Heritability is an estimate of the influence of genetic variation on a particular trait or disease. Thousands of SNPs have been significantly associated with different traits, while most of the heritability remains to be explained. Initially, risk variants found by GWAS were subject to unrealistic assumptions of effect sizes. Today, the understanding is that most alleles are associated with a relative risk of <1.2. These low effect sizes mean hundreds of thousands of individuals is needed to identify a sufficient amount of SNPs to explain >20% of genetic variance for a trait. Genetic variance is defined as differences in genotypes that contribute to phenotypic variance (Gibson, 2012).

Another issue that has been raised is that the majority of GWAS hits are located in regions without any known biological function. McClellan and King, (2010) speculate that these neutral SNPs persist by chance in the absence of selective pressure, and will not influence disease susceptibility. Others believe the high signals of associated SNPs might reflect other causal variants, either rare or common, that are in LD with the associated SNP, or alternatively that biological function is present, but undiscovered (Klein et al., 2010).

In the wake of this discussion, the CV–CD have been supplemented by other models that may explain disease susceptibility. One model is the theory that large-effect rare variants, typically with allele frequencies <1%, account for most of the genetic variance in complex disease. Other loci and environmental factors might be influential forces for disease severity and susceptibility. This theory does not require high penetrance, i.e. the
portion of individuals with a risk variant who also have the disease, and most unaffected individuals are expected to carry one or more risk alleles. Another model builds upon CV–CD, where the genetic loci detected by GWAS are believed to contribute to some of the disease susceptibility. This theory is called the infinitesimal model, where common genetic variants are hypothesized as the major cause of disease susceptibility. Here, the loci detected by GWAS are postulated to account for some of the heritability, although of small effect size. However, these SNPs are only the tip of the iceberg of the common variants in diseased individuals, where the remainder of risk is attributable to a combination of variants yet to be discovered. Some believe common variants can be imputed by computing LD variants from the risk loci found in GWAS (Gibson, 2012). Depending on the disease in question, a combination of characteristics from different suggested models, including those mentioned above, could contribute to diseases susceptibility.

2.2 Computational approaches for analyzing GWAS

A collection of GWAS data sets can have tens to hundreds, or even millions of features, i.e. SNPs, where each feature represents its own dimension. A popular computational approach for interpreting large, multidimensional data sets in different fields, including biology, is machine learning. Machine learning refers to a range of algorithms that try to interpret and extract important patterns and trends in data. It can be divided in two separate branches, depending on the kind of data available: supervised and unsupervised learning.

Supervised learning refers to algorithms that use a set of labelled data to train and test an algorithm, in order to create a model that predicts patterns and generalizes well to different data sets within the same domain. Each data sample in both training and test sets constitutes a pair of input and output values. The output value is a classification or expected value, given the input. This means that supervised learning depends on data where the result for each sample in at least a subset of the data available is known and labelled beforehand. The output, or label, is used to infer a relationship between the observed features of the data sample and a desired result. The goal of supervised learning is for the algorithms to correctly label unseen data (Christopher D. Manning, 2008). Samples of patient DNA with a particular disease could for instance be used to train a supervised algorithm to identify new DNA samples as candidates for a diseased group or not. Here, the DNA of the individuals would be the input data, and the disease they harbor the labelled output.

In GWAS, the trait or disease of association is known, but frequently the function and importance of the different risk variants are not. A primary goal of these studies is to gain insight into the underlying etiology of disease. In this setting, an area of interest would be to understand how and
to what extent diseases and traits are related. For these types of questions, unsupervised learning methods appear more relevant, as their goal is to find hidden structure within the data (Christopher D. Manning, 2008).

An unsupervised learning method called clustering has become increasingly popular for analysis of genome-wide data sets. In clustering, also referred to as cluster analysis or data segmentation, a collection of objects is grouped into subsets or clusters, in such a way that the resulting clusters contain objects related to each other to a greater degree than to objects assigned to other clusters. Clustering of genome-wide data sets provides a means of finding underlying genetic architecture. For instance, one could use clustering to discover which diseases share genetic components identified from GWAS.

Cluster analysis in itself is not an algorithm, but rather a general task to be solved, with the goal of partitioning the given data into groups, or clusters, of high intra-cluster and low inter-cluster similarity. In the following sections, definitions for cluster analysis is presented, along with two clustering paradigms: flat and hierarchical.

### 2.2.1 Cluster analysis definitions

In cluster analysis, we have a collection of \( N \) objects, \( X = (X_1, X_2, \ldots, X_N) \). Each object \( X_i \) consists of \( p \) features \( x_{ij}, j = 1, 2, \ldots, p \). These variables can also be referred to as attributes or variables. For a simple illustration of a two-dimensional feature space, see Figure 2.5. We denote the number of clusters by \( k \).

The concept of relations between data is the foundation for how clusters are organized. As a clustering algorithm has no way of knowing which objects belong together, it must find the best partition of clusters within the data collection based on the characteristics of its elements. An objective way of doing this is to supply the algorithm with a distance measure. The distance measure provides a means of telling which pairs of objects most likely belong together, and different distance functions can result in different clusterings. An illustration of how two common distance measures, Euclidean and cosine, affect which clusters are found, is given in Figure 2.5. The decision of distance metrics should be influenced by the kind of problem the clustering is solving, in order to get meaningful results (Christopher D. Manning, 2008).

Conceptually, the data representation used for clustering could be anything, as long as we are able to define an objective measure that quantify relations of interest between objects. However, most clustering implementations and toolkits will typically ask for an observation matrix of size \( N \times p \), with vectors of the same \( p \) features. Another option, rather than using feature vectors for each sample, is to calculate a distance matrix, denoted by \( Z \), and feed it directly to the clustering algorithm. The distance matrix will
2.2. COMPUTATIONAL APPROACHES FOR ANALYZING GWAS

Figure 2.5: Effect of two different distance measures in defining the closest data points in a continuous feature space. The resulting two clusters are given by coloring the points in orange or blue. The points in both plots are given by \( X_1 = [1, 3] \), \( X_2 = [1.5, 1] \), \( X_3 = [4, 1.5] \). \( X_2 \) is closest to the other node with the same color, in the orange cluster with euclidean distance, and the blue cluster with cosine distance.

be of size \( N \times N \), and for any two objects \( X_i \) and \( X_j \), their distance is inserted into \( Z[i,j] \). The distance matrix can be represented as a triangular matrix if the distance measure is constructed in a way that preserves symmetry, i.e. \( Z[i,j] = Z[j,i] \). With the distance matrix approach, the clustering algorithm does not need to know how the data was originally represented. This feature is also often supported in clustering toolkits.

2.2.2 Flat clustering

One way to do cluster analysis, is to create a flat set of clusters for the data set, without any internal structure, where each object is assigned to at least one cluster. This is called flat clustering, and can be seen as an optimization problem where we want all clusters to have the highest internal similarity and lowest external similarity as possible. The naive solution to this problem is to use a deterministic algorithm to enumerate all possible partitions of clusters within the data, and then choose the best one. The naive approach will result in exponentially many possible partitions, and does not scale well with increasing sample size. Instead, heuristic algorithms exist that approximate the solution by iteratively improving on an initial partition until some stopping criterion is met. Flat clustering is seen as conceptually simple and easy to implement, as it tries to directly partition the data using an objective measure. Many of the best known algorithms for flat clustering are fast, as their time complexity is linear in the number of objects (Christopher D. Manning, 2008).

While being fast, flat clustering has its drawbacks. The heuristic nature of the algorithms means the final clustering is non-deterministic, and the
results may change between runs. In addition, they are susceptible to local optima, and therefore not guaranteed to find the globally best partition of the clusters. We also need to define the number of clusters, $k$, beforehand, which will heavily influence the resulting partition. If the optimal solution would be at a higher or lower number of $k$, the algorithm would enforce the pre-specified number of clusters. Lastly, with no structure other than a partition of the data set, it is difficult to infer any information of relations between the clustered objects.

### 2.2.3 Hierarchical clustering

Hierarchical clustering is a different way cluster analysis can be done. The hierarchical clustering technique provides the intra-relational information that flat clusters lack, and is not dependent on a fixed $k$ chosen beforehand. The result of this method is hierarchically nested clusters, often visualized as a binary tree structure known as a dendrogram. For examples of dendrograms produced using different clustering settings, see Figure 2.6. The leaves are the data samples, or singleton clusters. In the dendrograms, merges are shown as horizontal lines, and the $y$-axis represents the distance between the merged clusters. The branches increase in length with the distance.

There are two approaches for creating hierarchical clusters: Divisive and agglomerative. Divisive clustering is a top-down approach, and starts with all the data objects being members of the same cluster, upon which
2.2. COMPUTATIONAL APPROACHES FOR ANALYZING GWAS

Figure 2.7: A simple visualization of different linkage criteria for hierarchical agglomerative clustering. Single linkage (a) is the minimum distance between any two members of the clusters, complete linkage (b) is the maximum distance between any two members of the clusters, centroid linkage (c) is the distance between the centroids of the objects of the clusters, and average linkage (d) is the average pairwise distance between two clusters.

2.2.4 Linkage criterion

For hierarchical agglomerative clustering (HAC), an objective way of finding the next clusters to merge or divide is needed. The parameter to decide on for this is called the linkage criterion, which is based on the chosen measure of distance for the clustering. The linkage criterion define the strategy for how the distance measure should be used to find the most similar clusters. HAC algorithms will commonly choose to merge the two clusters of the lowest linkage value. Figure 2.6 shows how different linkage criteria can affect the final clustering. Appendix A provide further
technical details related to linkage criteria definitions. The most widely used linkage criteria are the following, visualized in Figure 2.7 (Christopher D. Manning, 2008):

- Single-linkage: Merge the clusters where the minimum distance between any two members is smallest
- Complete-linkage: Merge the clusters where the maximum distance between any two members is smallest
- Centroid-linkage: Merge the clusters where the distance between their centroids is smallest
- Average-linkage: Merge the clusters where the average pairwise distance is smallest

2.3 Review of methods for comparison of GWAS

Clustering has become an increasingly popular method for explorative analysis within the field of bioinformatics. Sequence-level data sets can for instance be clustered to illustrate or point out genomic areas of interest, where information of shared genetic components is found. These discoveries can be valuable in a preliminary phase where further analyses of the data are to be decided on. Discussions concerning the use of clustering on sequence-level genomic data has however been limited.

We have only found a modest amount of articles where clustering of disease-associated or genome-wide data is the main focus. Literature concerning analysis of GWAS have been extensively reviewed, for instance related to cancer, gene enrichment and overlap with epigenetic data sets (Dunham et al., 2014; The Roadmap Epigenomics Consortium et al., 2015; Cowper-Saliali et al., 2012).

Of the articles read, four were chosen for more in-depth study, as they present methods that in some form can be used to compare complex common diseases with each other on the genome-wide level. The methods and results of the four articles are reviewed in the following sections. The first article is methodological in nature, and presents a standardized clustering approach for genome-wide data sets, implemented as a tool called “ClusTrack”. The second article, in an initial exploration of the underlying genetic architecture of 39 common diseases, clusters disease-associated data based on their shared genetic loci. The third article presents a regression-based approach for identification of genetic correlations between data sets of GWAS summary statistics. The fourth has a purely biological focus, and compares GWAS findings at the genetic and epigenetic level.
2.3. REVIEW OF METHODS FOR COMPARISON OF GWAS

2.3.1 Clustering using vectors of predefined bins

In their article “ClusTrack: Feature Extraction and Similarity Measures for Clustering of Genome-Wide Data Sets”, Rydbeck et al., (2015) present a framework and online tool for clustering genome-wide data sets. Approaches for feature extraction and representation of genome-wide data for use in cluster analysis are given, as is measures of similarity for the different representations. The methods presented are defined for several types of genome-wide data sets, not just SNPs, but here we will only cover the aspects of their methodology related to data of common variants.

Rydbeck et al., (2015) discuss two different ways of representing genome-wide data sets for use in cluster analysis. The first approach is a vector definition where the features are single base pairs, for which set theory is used to compute distance between pairs of vectors. Given two data sets of genomic points, $A$ and $B$, distance is computed with either (2.1) or (2.2), as defined below. The resulting distance matrix is passed to the R library `hclust` with the specified linkage criterion for the actual cluster analysis. Comparisons based on features of base pairs only indicate shared reported risk variants, and the authors note that clustering with this approach will not reflect biological relationships between tracks.

$$1 - \frac{A \cup B}{A \cap B} \quad (2.1)$$

$$\frac{1}{(A \cup B)/(A \cap B)} \quad (2.2)$$

In the second approach for data representation, a definition of bins along the chromosome is used to accommodate biological correlations between neighboring base pairs. A bin represents a smaller genomic region of base pairs, and two different bin definitions are given. In the first, a bin is defined as a number of contiguous base pairs. In the second, a bin is defined as a set of dependent positions given by reference tracks. The resulting data vectors contain features of aggregated measures of hits within each pre-specified bin. The aggregate measure given for bins of point tracks is the proportion of points falling within each bin. After vectors with features of bin aggregates have been created for all data sets, the `hclust` package is used to cluster the data with one of the similarity measures available in the clustering library. The specified linkage criteria is similarly passed as argument to the package. Rydbeck et al., (2015) conclude that the majority of data samples the tool is tested with, form meaningful subclusters, indicating that the methods capture a biological notion of similarity.

For both clustering approaches, where features are either defined as base pairs or bin aggregates, the following linkage criteria are available for the clustering: Single, complete, average, ward, centroid and median.
2.3.2 Clustering diseases based on shared genetic loci

In their article “Genetic and epigenetic fine mapping of causal autoimmune disease variants”, Farh et al., (2015) cluster the shared genetic loci of 39 diseases and traits, revealing groups of phenotypes with related clinical features. The clustering is done as an initial exploration of the underlying genetic architecture of the diseases, to discover shared genetic loci.

To make sure the GWAS chosen for analysis are well-powered, they only include studies that have at least 6 hits on the genome-wide significant level of \( p \leq 5 \times 10^{-8} \). From this set, index SNPs with significance \( p \leq 10^{-6} \) is kept for analysis. Another filtering is performed on the basis that multiple index SNPs map to the same genetic locus, defined as within 500kb of each other. Only the most significant SNP within such a locus is kept for downstream analysis.

For each pair of diseases, their respective lists of tag SNPs is compared to find instances of common genetic loci, defined as tag SNPs within 500kb of each other. More formally, given two diseases or traits \( X_1 \) and \( X_2 \), their overlapping genetic loci, denoted \( N(X_1, X_2) \), is estimated as the count of index SNPs where \( |x_{i1} - x_{j2}| \leq 500kb \), \( i \) and \( j \) being coordinates of the respective index SNPs. As the supplementary material is brief on how the shared genetic loci is used further to compare diseases, we thought this value was used directly to compute a measure of correlation between a pair of diseases.

Despite several attempts over time, we never got in touch with the researcher responsible for the correlation matrix presented in the article.
(Figure 2.8), for an explanation of how the computations were done. We did however communicate with one of the other authors behind the study. He was not involved in the details of the calculations of the figure, but believed the approach was different than our initial suggestion. To compare two diseases \( X_1 \) and \( X_2 \), rather than computing \( N(X_1, X_2) \) and using it directly to find a measure of correlation, overlap vectors for all \( X_i, i = 1 \ldots N \) are computed. In such a vector, denoted \( V_i \), the features correspond to \( N(X_i, X_j), j = 1 \ldots N \). With this representation for the 39 diseases, the two diseases would be defined as follows:

\[
X_1 = [N(X_1, X_1), N(X_1, X_2), N(X_1, X_3), \ldots, N(X_1, X_{39})] \\
X_2 = [N(X_2, X_1), N(X_2, X_2), N(X_2, X_3), \ldots, N(X_2, X_{39})]
\]

Here, the second disease in each overlap computation is used as a running index. All pairs of vectors are then used to compute a correlation coefficient, using for instance Spearman or Pearson\(^6\). In other words, two traits are considered similar, or correlated, if they are similar in their overlap to other traits. Most likely, in the comparison of two such vectors, \( V_i \) and \( V_j \), the \( i \)th and \( j \)th feature is removed before computing the correlation coefficient, in order to avoid that genetic loci overlap for a track with itself is used in the comparison.

### 2.3.3 Using regression and LD to find genetic correlation

In their article “An atlas of genetic correlations across human diseases and traits”, Bulik-Sullivan et al., (2015) present a regression-based approach for computation of pairwise genetic correlation between diseases and traits. The article takes into account the statistical problem of reported risk variants in GWAS, as discussed in Section 2.1.6, and use the infinitesimal model as basis for their methods.

Bulik-Sullivan et al., (2015) use regression on GWAS summary statistics to incorporate information of low-effect SNPs. As summary statistics are used, all SNPs of some association are considered, not just the ones that reach genome-wide significance. Each SNP in the data sets used is reported with minor and major alleles, effect size and p-value. From this information, a \( z \)-value that indicates the effect size and direction of the effect allele is computed. A positive \( z \)-value indicates increased risk of the associated trait, and a negative \( z \)-value that the allele has protective properties and decreased risk.

For all reported SNPs, an LD score is computed. The LD score is defined as the sum of \( r^2 \) for all SNPs in LD with the given variant. For each associated SNP, the \( z \)-values reported in two studies is regressed onto the LD Score, and the resulting slope can be used to find the genetic covariance between

\(^6\)The chosen correlation coefficient behind Figure 2.8 was not disclosed in the correspondence concerning its creation.
traits. This covariance, normalized by SNP heritabilities, results in a score for genetic correlation. The method assumes that the GWAS effect size estimate for a given SNP incorporates the effects of other SNPs in LD with it. The results of the computed genetic correlation for the different diseases and traits, can be seen in Figure 2.9.

### 2.3.4 Genetic investigation of five chronic inflammatory diseases

With the goal of investigating relationships between five chronic inflammatory diseases, Ellinghaus et al., (2016) performs a cross-disease GWAS, with genotype data from more than 86,000 individuals of European descent in their article “Analysis of five chronic inflammatory diseases identifies 27 new associations and highlights disease-specific patterns at shared loci”. Cross-disease genetic studies give researchers the opportunity to uncover discrete disease pathways and explore details of what seems to be shared and distinct genetic etiologies (Ellinghaus et al., 2016). In their study, 52,262 cases and 34,213 controls are genotyped for the immune-driven diseases ankylosing spondylitis, Crohn’s disease, psoriasis, primary sclerosing cholangitis and ulcerative choliitis. This is the currently largest available genetic data sets available for these diseases. Using novel cross-phenotype analytical methodologies on the genotype information, the study identifies new risk variants for three of the conditions, in addition to shared risk-associated SNPs between diseases.

Through assessment and analysis of their data, Ellinghaus et al., (2016) finds that many SNPs were shared between diseases. This finding supports
the theory that presence of shared etiological pathways is the basis for clinical co-occurrence.

2.4 The Genomic HyperBrowser

The Genomic HyperBrowser, HyperBrowser for short, is an open source, web-based software system for comparative genomic analysis. It is the result of an ongoing collaboration between the research group for biomedical informatics at the department of informatics, University of Oslo, and the Radium Hospital in Oslo. As stated on the HyperBrowser front page\(^7\), it provides “statistical methodology and computing power to handle a variety of biological inquiries on genomic data sets”. With a range of already implemented biological tools and availability of genomic data in the form of annotated genomic tracks, the HyperBrowser can be used directly in analyses of data sets, or as a basis for developing new tools (Sandve, Gundersen, Rydbeck, et al., 2010). This flexibility takes into account the variability of technical skill the HyperBrowser users possess. While biologists without programming experience can test their hypotheses by using the graphical user interface and already implemented tools, those who wish to further develop functionality can add analyses and tools to the code base. A specialized version of the HyperBrowser, called GSuite Tools focus on functionality for analyzing collections of genome-wide data sets.

The Genomic HyperBrowser builds on the Galaxy software framework, meaning that many of the features provided in the public Galaxy Server are available (Sandve, Gundersen, Johansen, et al., 2013). Galaxy features such as history elements and pages provide support for reproducible research, where data sets, test runs and functionality easily can be shared with collaborators and external audiences (Afgan et al., 2016). The core concepts for HyperBrowser development, are the technical entities tools and statistics.

2.4.1 Tools

The main flow of control in the HyperBrowser is based on the class GalaxyInterface, which links the web functionality to the HyperBrowser core. By following certain code patterns, described in templates in the HyperBrowser documentation and code, this implementation supports fast prototyping and creation of new tools.

A HyperBrowser tool typically takes as input some form of genomic data. The web interface is defined in the tool code, where input boxes of different formats can be added and customized. These input boxes provide options for the user to choose amongst, typically parameters for the underlying tool.

\(^7\)https://hyperbrowser.uio.no, visited 13.06.16
class ExampleStat(MagicStatFactory):
    pass

class ExampleStatUnsplittable(Statistic):
    def __createChildren__(self):
        self._addChild(RawDataStat(self._region, self._track, 
                                   TrackFormatReq(allowOverlaps=True)))
        self._addChild(RawDataStat(self._region, self._track, 
                                   TrackFormatReq(allowOverlaps=True)))

    def __compute__(self):
        track1 = self._children[0].getResult()
        track2 = self._children[1].getResult()

        intermediateResult = <computation between track1 and track2>
        return intermediateResult

class ExampleStatSplittable(StatisticSplittable):
    def __combineResults__(self):
        accumulator = <structure>
        for childResult in self._childResults:
            <add childResult property to accumulator>

        return accumulator

Figure 2.10: Code for an example implementation of a statistic, defined for computational analysis on two tracks. The Unsplittable class defines data representations and computations for local analysis, and the Splittable class defines how the intermediary local results should be combined for a global result.

analysis. The actual biological analysis runs from the execute function in the tool class, which is called when the user clicks “execute” in the web interface.

A developer can choose to define all computations from execute within the tool code, or use one or more HyperBrowser statistics, defined in the next section, as a part of the execution. HyperBrowser best practices and documentation suggest that modular statistics is the preferred code structure.

2.4.2 Statistics

A statistic is a HyperBrowser module that defines any mathematical or statistical operation on one or several genomic data sets referred to as tracks. It can be defined as a question, i.e. a hypothesis, or be a computation. The structure of a statistic is based on the functional primitives map and reduce\(^8\). In this functional scheme, the map primitive

\(^8\)Not to be confused with the machine learning algorithm MapReduce, developed by
applies a function to all elements of a list, and the results for each element-wise computation is returned in a new list. The reduce primitive is further used to compute a global value across the list of results, which is returned.

When a statistic is run on one or multiple tracks, the data is first partitioned into what is referred to as bins. The bins are longer continuous regions of predefined size, and are the same across all tracks. The largest bins possible are those reflecting actual chromosome sizes, which are variable in length for the different chromosome numbers. In the map part of a statistic, a function is applied to all track bins, and a list of results is returned. In reduce, the list of local results is combined in a global result for the given analysis.

A statistic is specified in a separate file, an example of which can be seen in Figure 2.10. In this example, two tracks are passed to the statistic for analysis. A statistic file will usually contain two or three classes, as listed below. For these classes, the map part is defined in the StatUnsplittable class, while the steps of reduce is defined in the StatSplittable class:

1. A main public statistics class, ending with Stat
2. The class of main computation in each bin, ending with StatUnsplittable
3. An optional class for combining results across bins, ending with StatSplittable

The class StatUnsplittable require the functions _createChildren and _compute. The first function define the data available in the statistic, using the tracks the analysis is run on. The other function defines the computation done for each bin of the input data.

The StatSplittable class is optional, but in order for the analysis to output a global result for the entire track, it must be defined. This definition can be explicit, through the given class, or implicit, where intermediate statistics objects that are defined as children in the statistic, have a defined splittable class. The function _combineResults has access to all local results, and its return value is the global value of the full analysis.

An example of how the execute function of a tool is set up to call a statistic with two tracks in a GSuite is given in Figure 2.11. AnalysisSpec takes in the Statistic object to use, while GlobalBinSource fetches bins representing all chromosomes in the given reference genome specified in the GSuite. A double loop iterates through all tracks in the GSuite, and the function doAnalysis calls the specified statistic on all pairs of tracks.

2.5 Representing genomic data

Before computational analyses can be conducted, we need to define how the genomic data should be stored. Several different formats for genomic

CHAPTER 2. BACKGROUND

```python
@staticmethod
def execute(choices, galaxyFn=None, username=' '):
    analysisSpec = AnalysisSpec(ExampleStat)
gSuite = getGSuiteFromGalaxyTN(choices.gSuite)
analysisBins = GlobalBinSource(gSuite.genome)

    tracks = gSuite.allTracks():
        for gSuiteTrack1 in tracks:
            for gSuiteTrack2 in tracks:
                track1 = Track(gSuiteTrack1.trackName)
                track2 = Track(gSuiteTrack2.trackName)
                result = doAnalysis(analysisSpec, analysisBins, [track1, track2])
                print result.getResult()
```

Figure 2.11: How the execute function of a tool can be set up to call a Statistic on all pairs of tracks in a GSuite. Here, the ExampleStat statistic is chosen for analysis on the tracks. A double loop is used to iterate over the tracks in the given GSuite, so that the statistic can be called for all pairs of tracks.

tracks exist, with different required columns and properties. One reason for this is the fact that different properties are required for analysis of the data in specific domains. Choice of technologies or experimental methods may also produce differences in representational formats, as the availability of information of detailed genomic events will influence the resulting data.

2.5.1 Genomic track data

In their article “Identifying elemental genomic track types and representing them uniformly”, Gundersen et al., (2011) discuss a general track representation, which they refer to as a genomic track, or GTrack. In this definition, a reference genome, such as hg19 or hg38, is abstracted as a line based coordinate system. Elements of genetic or functional importance is represented as a series of data units positioned on such a line. The track elements must always contain their position on the reference genome, but can also include a value associated with it and other features that explains its function, such as connections to other track elements located in a different position (Gundersen et al., 2011).

GTrack is the standard format used in the Genomic HyperBrowser, but other common formats are also supported, such as BED and FASTA (Sandve, Gundersen, Johansen, et al., 2013). A GSuite is a collection of tracks, ideal for computational analyses where several tracks are needed. The primary tracks of a GSuite, i.e. the original data sets, does not need to be of the same format.
2.5. REPRESENTING GENOMIC DATA

<table>
<thead>
<tr>
<th>Track type</th>
<th>genome</th>
<th>seqid</th>
<th>start</th>
<th>end</th>
<th>value</th>
<th>strand</th>
<th>id</th>
<th>edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>?</td>
<td>!</td>
<td>✓</td>
<td>x</td>
<td></td>
<td>?</td>
<td>?</td>
<td>x</td>
</tr>
<tr>
<td>LP</td>
<td>?</td>
<td>!</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>?</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>VP</td>
<td>?</td>
<td>!</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>?</td>
<td>?</td>
<td>x</td>
</tr>
</tbody>
</table>

Table 2.1: Three specifications for GTracks, namely points (P) track, linked points (LP) track and valued points (VP) track. ✓ means the column is mandatory, ? optional, × not allowed, and ! that the property must be present, either as column or in bounding region specification.

2.5.2 GTrack representation of SNPs

A set of SNPs can be represented as a point track, which contains track elements that refer to single discrete positions. If the different points also have a defined value, such as the p-value of a significantly associated SNP, the track elements can be represented as valued points in a valued points track (VP track) (Gundersen et al., 2011). The VP track specification is given in Table 2.1, but the track can have additional columns that describe features of each track element.

Each track has a header that is used when a track is preprocessed within the HyperBrowser. The preprocessing creates a binary, indexed version of each track, allowing much faster access to the track data within the HyperBrowser. The header properties enable the HyperBrowser to uniformly store tracks that differ in terms of columns and indexation. The preprocessing checks that the entire track agrees with the header properties and is correctly defined in terms of the track type specification. For a VP track, an example header is shown below. The track is set to be 1-indexed, i.e. the reference genome coordinates start at 1. In addition, the values are defined in the column “pvalue_mlog”.

```plaintext
#track type: valued points
#value column: pvalue_mlog
#1-indexed: true
```

2.5.3 GTrack representation of linkage disequilibrium

The LD structure between variants can be visualized as an undirected or bidirectional graph, see Figure 2.12. This structure can further be represented as a linked points track (LP track), which support functions for graph processing (Gundersen et al., 2011). The LP track specification can be seen in Table 2.1. As with VP tracks, extra columns can be added to an LP track that exceeds the minimum column requirements. A header example for an LP track is shown below. In the example, the direction of edges is specified to be undirected, and weights must be defined for all edges.
In an LP track, each point must be uniquely identified by an id, which can further be used to describe edges between points. For SNPs, this id could be their unique rsIDs. The edges from a given point is a ;-separated list of the ids the point is connected to, with weights given by $r^2$. An rsID can only appear once as identifier for a track element, but in the edges-column where it represents which SNPs are in LD, the rsID can occur multiple times. If we assume the given rsIDs are located on chromosome 1, the graph illustration of Figure 2.12 would translate into the following tab-separated LP track rows:

<table>
<thead>
<tr>
<th>seqid</th>
<th>start</th>
<th>id</th>
<th>edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr1</td>
<td>1000</td>
<td>rs1</td>
<td>rs2=0.83</td>
</tr>
<tr>
<td>chr1</td>
<td>13240</td>
<td>rs2</td>
<td>rs1=0.83;rs3=0.76;rs4=0.97</td>
</tr>
<tr>
<td>chr1</td>
<td>40318</td>
<td>rs3</td>
<td>rs2=0.76</td>
</tr>
<tr>
<td>chr1</td>
<td>79123</td>
<td>rs4</td>
<td>rs2=0.97</td>
</tr>
</tbody>
</table>

### 2.6 Collecting data for diseases and traits

Data of diseases and traits must be collected and uploaded to the HyperBrowser before it can be used for analysis. This section describe sources of data for GWAS and LD, and how to create and upload collections of data as GSuites in the HyperBrowser.

#### 2.6.1 Sources of GWAS data

The National Human Genome Research Institute (NHGRI) and the European Bioinformatics Institute (EMBL-EBI) provide a curated, publicly available catalog of GWAS, called the NHGRI-EBI GWAS Catalog, or the GWAS Catalog for short. It contains all published GWAS assaying at least 100,000 SNPs, and all SNP-trait associations with p-values $< 1.0 \times 10^{-5}$ (Hindorff...
et al., 2009), slightly higher than the standard significance threshold of $5 \times 10^{-8}$. As of July 2016, the GWAS Catalog is built on 2510 studies, with 21321 identified SNPs and 24065 SNP-trait associations.\footnote{https://www.ebi.ac.uk/gwas/search, with latest data release July 24th 2016}

GWAS data can be directly downloaded from the GWAS Catalog. The attributes stored for each SNP can be found in Appendix B.2, where Tables B.2 and B.3 show the information stored for all studies and SNPs, respectively. Since multiple studies might contribute to associations of the same trait, a particular SNP can be reported more than once if a data set with all SNPs related to a disease is downloaded. Similarly, tag SNPs are chosen by the research institutions or the genotype platform providers, and might not be consistent across all studies. This means some of the risk alleles reported for a trait in the GWAS Catalog might be in LD with each other, and refer to the same haplotype block.

Some research consortia have published GWAS summary statistics, data sets that contain imputed and genotyped data resulting from one or several GWAS. The summary statistics are not restricted to any thresholds of significance, and can contain information of millions of SNPs. Summary statistics usually contain rows of SNPs, identified by their rsIDs. P-values and effect size values are often included, as is information of the effect alleles. Different consortia often use different columns and column headers in their summary statistics, however, meaning the data should be converted to a standard format before comparisons between studies are conducted. LD Hub is a centralized database for summary-level GWAS results,\footnote{https://ldsc.broadinstitute.org} and provides a list of resources where GWAS summary statistic downloads can be found (Zheng et al., 2016).

### 2.6.2 Sources of LD data

To incorporate information of actual LD with GWAS data sets, data of LD between variants is needed. There seems to be no precomputed data sets with LD scores between variants directly available for download from any public databases, with the exception of the outdated LD data from HapMap phase 3.

The Ensembl Project\footnote{https://www.ensembl.org} provide, amongst other things, a range of public genomic databases, available online. Their variation database can be used to compute LD between variants, based on 1000G phase 3 genotype data. Detailed instructions for installation and local setup of the API, along with the scripts used for LD computation, can be found in Appendices E.1 and E.2, respectively.
2.6.3 Creating GSuites within the HyperBrowser

The HyperBrowser directly supports GSuite creation from many of the best known genomic resources, such as the GWAS Catalog. Under the header “HyperBrowser GSuite Tools” the option to “Create a GSuite of genomic tracks” is found. The submenu of this option gives the user a multitude of ways to create a GSuite. To use external data resources, choose “Create a remote GSuite from a public repository”. The NHGRI-EBI GWAS Catalog is available amongst the listed remote repositories. For the purpose of finding SNPs related to specific diseases and GWAS, the metadata attributes from Tables B.2 and B.3 can be used in a more refined search. One or multiple disease tracks can be chosen for GSuite generation.

If the HyperBrowser is used to extract disease data sets from the GWAS Catalog, the resulting GSuite will consist of VP tracks, one for each chosen disease. Table B.4 in the appendix shows the column headers for a GWAS Catalog track, and example data for a chosen SNP track element. The value of each track element is set to the column “pvalue_mlog”, defined as $-\log(pvalue)$.

As mentioned in Section 2.6.1, multiple GWAS are conducted independently, and many of the same SNPs might be reported and stored for the same trait. This will be reflected in the GSuite tracks. If a track contains multiple rows of the same variant, or alternating variants with different rsIDs mapping to the same start coordinate, the HyperBrowser will collapse these rows into one representation of the SNP, unless otherwise specified. If several columns, such as “snps” or “pvalue_mlog”, have different values, they will be collapsed into a |-separated string or set to nan (not a number).

2.6.4 Software for a standardized summary statistic format

As mentioned in Section 2.6.1, the columns of GWAS summary statistics might vary greatly across data sets. The script munge_sumstats.py, provided in the LD score software by Bulik-Sullivan et al., (2015)\(^{12}\), parses through summary statistics and converts them to a standardized sumstat format, specified in Table 2.2. In addition to providing a uniform column representation for the summary statistics, the script applies filters to remove structural variants, SNPs of low sample size, poorly imputed SNPs, strand ambiguous SNPs and SNPs whose alleles does not match the 1000G alleles (Bulik-Sullivan et al., 2015).

The script outputs effect sizes of the individual SNPs in the form of z-values that also indicate direction of effect. A positive value indicate that an increase in risk of disease susceptibility is associated with the effect allele, and a negative value a decrease. P-values does not indicate effect size, but can still be useful for analysis of summary statistics. If the conversion from

\(^{12}\)https://github.com/bulik/ldsc
## 2.6. COLLECTING DATA FOR DISEASES AND TRAITS

<table>
<thead>
<tr>
<th>Column header</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNP</td>
<td>SNP identifier (e.g. rsID)</td>
</tr>
<tr>
<td>N</td>
<td>Sample size (which may vary from SNP to SNP)</td>
</tr>
<tr>
<td>Z</td>
<td>Z-score. Sign with respect to A1</td>
</tr>
<tr>
<td>A1</td>
<td>First allele (effect allele)</td>
</tr>
<tr>
<td>A2</td>
<td>Second allele (other allele)</td>
</tr>
</tbody>
</table>

Table 2.2: Required headers in the sumstat file format

p to z-values done at the end in `munge_sumstats.py` is dropped, and “P” added as column header, the original p-values from the summary files are kept instead of the z-values. The standardized summary statistic files can be collected in a folder, zipped and uploaded to the HyperBrowser as a GSuite of primary tracks with the sumstat format.
Chapter 3

Methods

This section addresses choices regarding data representation and distance measures that should be considered when different GWAS data sets are subject to cluster analysis. Two different methods for GWAS data representation and comparison are presented. The first method discusses different binary representations, defined through a mathematical taxonomy of binary properties. The second method explores data representations of continuous vectors. Measures of similarity and correlation are described for the different representations.

3.1 Clustering

For the cluster analysis of diseases, as represented by GWAS data sets, we will use a hierarchical agglomerative clustering (HAC) algorithm. As described in Section 2.2.3, to compare data with this technique, the algorithm needs a way to objectively assess relationships between data. With such an assessment, we can quantify how “similar” two disease-associated data sets are.

With HAC, the common way of objectively comparing data is to use a distance measure. Distance and similarity quantify the same relation to a great extent, except in opposite directions. Therefore, although distance is the chosen focus for clustering, the term similarity is used interchangeably when comparative measures are discussed. For a standardized interpretation of clustering dendrograms, all similarity measures or correlation coefficients used to compare diseases are converted into measures of distance before clustering is performed.
3.2 Binary data representations

There are many different ways to represent GWAS data sets using a binary definition, depending on what biological properties we want to incorporate. This section starts with the definition of a mathematical taxonomy that can be used to express binary data representations. It is followed by different methods for representing the GWAS data using the taxonomy expressions, where properties related to tag SNPs and LD are incorporated in the binary definitions. After the different methods have been presented, we give an overview of the properties of each representation and how they relate to each other. Finally, measures of distance and similarity that can be used with the binary specification are described.

3.2.1 Binary Taxonomic Units

A binary vector is defined as an array of features that are either 0 or 1. In the binary vector context, a feature that is present is set to 1, and the rest is set to 0. When binary vectors are compared, a scheme of Binary Taxonomic Units (BTUs) can be used, which defines the four possible combinations of binary features:

- Positive match: Values of \( i \) and \( j \) are both 1
- Mismatch: Values of \( i \) and \( j \) are \((1, 0)\)
- Mismatch: Values of \( i \) and \( j \) are \((0, 1)\)
- Negative match: Values of \( i \) and \( j \) are both 0

For a pair of binary vectors subject to comparison, the BTU summaries \( a, b, c \) and \( d \) can be defined, which express the number of mismatches and matches between the vectors, as shown in Table 3.1 (Choi, Cha, and Tappert, 2010). Using these definitions, \( a + d \) gives us the total number of exact matches, negative and positive, between the samples, while \( b + c \) is the total number of mismatches. The total number of data in the two vectors is given by \( n \).

<table>
<thead>
<tr>
<th>( i ) (Presence)</th>
<th>( j ) (Presence)</th>
<th>( i ) (Absence)</th>
<th>( j ) (Absence)</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Presence)</td>
<td>( a )</td>
<td>( b )</td>
<td>( a + b )</td>
<td></td>
</tr>
<tr>
<td>0 (Absence)</td>
<td>( c )</td>
<td>( d )</td>
<td>( d + c )</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>( a + c )</td>
<td>( b + d )</td>
<td>( n = a + b + c + d )</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Expression of BTU summaries for the binary features \( i \) and \( j \).
CHAPTER 3. METHODS

3.2.2 Definitions of binary features

Although the BTU summaries of Table 3.1 are defined for binary vectors, it is possible to define $a$, $b$, $c$ and $d$ differently, to accommodate biologically meaningful representations. This way of representing GWAS data sets facilitates experimentation of data representation, while measures of similarity and distance that can be defined in terms of the BTU summaries remain unchanged.

Feature representation: single base-pair position

Conceptually, a GWAS data set can be represented as a binary vector with ~3.3 billion features, where each genomic position is its own feature. With this approach, a position is assumed to represent a single SNP, despite the fact that many SNPs of different genotypes can map to the same position. All reported SNPs in the data set represent hits in the vector, i.e. the features are set to 1. See Figure 3.1 for a visualization of two tracks with SNPs shown as dots along the genome. The corresponding binary vectors, with features of single base positions, are displayed at the top and bottom.

This binary vector definition where features are single base-pair positions can easily be converted into BTU summaries. Matches and mismatches are defined as they were in Table 3.1. A positive match between two SNPs is highlighted in red in the Figure 3.1. This representation of features as the single bases of the genome is conceptually one of the simplest ways to define matches and mismatches in terms of the data available, but contains no extensions where biological information is considered.

Feature representation: bin

Data for diseases downloaded from the GWAS Catalog can contain several SNPs that are in LD with each other, as described in Section 2.6.1. In a track where variants are correlated, several tag SNPs present in the data will represent the same LD blocks. When deciding on a biologically
meaningful feature representation, LD between variants is an important aspect to consider, as these variants by definition represent the same signal of association. In a comparison, tracks with many variants in LD might artificially get higher scores of similarity, as overlap between features that in effect represent the same biological entity are counted more than once. Another consideration of the LD structure between reported tag SNPs, is that two SNPs on different tracks could be in LD and thus represent the same biological feature, despite differences of their chromosomal positions.

We can approximate feature representations of LD blocks or haplotypes, and at the same time remove intra-correlated SNPs in LD, by using an approach where contiguous genomic positions are reduced to one genomic feature. One way to incorporate a feature representation of contiguous genomic positions is to use a binary vector definition where the features are represented by bins. A bin size approximating the length of an LD block is determined, and the genomic track partitioned into bins. If a bin contains a tag SNP, it will count as a hit. In comparison with other tracks, the matches now account for entire genomic regions. As with the single base binary vector, matches and mismatches are defined as in Table 3.1. See Figure 3.2 for a visualization of the feature representation with bins. Several positive matches of (1, 1) are highlighted in red.

The binning approach is rather naive in its approximation of a haplotype. With the predefined bin size, each chromosome is split up accordingly without any consideration of the actual distribution of SNPs along the genome. Two SNPs in close proximity might by chance end up in different bins as border cases. An example of this can be seen at in Figure 3.3a, where two SNPs fall into separate bins and are not matched, although other pairs of the same distribution in other bins are.

**Feature representation: block definition of genetic locus**

The border cases of SNPs in different bins mentioned in the previous section is shown in Figure 3.3a, where the vertical line represents the border between two bins. Instead of partitioning the chromosomes directly into bins of the same size, the reported SNPs themselves can define the
CHAPTER 3. METHODS

Figure 3.3: Impact on positive matches given different approximations of a haplotype block. The same two points have no overlap with a definition of bins, full overlap with a block definition, and partial overlap with a Gaussian definition.

borders of an approximated LD block, as seen in Figure 3.3b. The feature representation is now defined as a physical window of subsequent genomic positions surrounding a tag SNP, which we refer to as a genetic locus. The features of this representation is not necessarily uniformly distributed, as with the bins, as the genetic loci are dependent on the distribution of the reported SNPs.

To remove intra-correlation of SNPs in LD on the same track, a preliminary filtering of SNPs that map to the same genetic locus is conducted. If multiple SNPs on the same track are located within the predefined window of continuous base pairs, the most significant SNP is kept as representative for the region.

With the pre-specified block definition of genetic loci, we have moved away from a binary vector representation, as different features are not guaranteed to represent the exact same genomic positions. The definitions of \( a, b, c \) and \( d \) that applied to the single base and bin feature representations can thus no longer be used. Instead, a positive match between tracks is defined as two SNPs mapping to the same locus, shown in red in Figure 3.3b.

The full block size, or window, is used when filtering out the tag SNPs to represent the different genetic loci. To count overlap between two genetic loci on different tracks, \( a \) is increased if \( |x - y| \leq \frac{\text{window}}{2} \), where \( x \) and \( y \) denote the positions of the compared tag SNPs. Mismatches, i.e. \( b \) and \( c \), are defined as points not overlapping with any locus in the other track. Negative matches, \( d \), are not defined for this representation, as the features are neither uniformly nor evenly distributed across the genome.

**Feature representation: Gaussian definition of genetic locus**

The main argument for defining a genetic locus is to account for LD variants of the same haplotype block. Variants in LD are observed to often occur in close proximity to each other, meaning an approximation where the same importance is given to a match where two SNPs are at the borders of each other’s genetic loci as that of two SNPs in close proximity, might not
3.2. BINARY DATA REPRESENTATIONS

Figure 3.4: Visualization of matching using LD expansions and haplotype blocks. The black dots are tag SNPs representing the haplotype blocks, where red curves show which variants it is in LD with. Matches between haplotypes are found when SNPs on different tracks map to the same position, shown with dashed arrows. Matches between haplotypes are counted only once, shown with a red dashed arrow.

To give overlap of distant alleles mapping to the same genetic locus less importance than alleles located close together, we can use a Gaussian function for computing genetic overlap, weighting down positive matches that have less probability of reflecting SNPs in LD. The Gaussian definition will give SNPs within the same locus a positive score as in the function given in (3.1), where $\sigma = \frac{\text{window}}{4}$. Here, $x$ and $y$ denote the positions of the tag SNPs on different tracks for which we want to compute a match score. If the positive match score is less than 1, i.e. the SNPs have some distance between them, although they map to the same locus, the mismatch scores $b$ and $c$ are also updated with $\frac{1-a^2}{2}$.

$$a = e^{-\frac{(x-y)^2}{2\sigma^2}}$$ (3.1)

$\sigma$ is chosen to be a quarter of the pre-defined haplotype size, as the Gaussian function with these settings agrees with the empirically observed decay of LD between variants for increasing physical distance (The 1000 Genomes Project Consortium, 2015). The rest of the implementation and specifications are identical to that of the block definition. Figure 3.3c visualizes the Gaussian loci definition. While the full positive match of the block definition to the left of the figure is highlighted in red, the Gaussian matching is only partial, and shown in gray instead.

Feature representation: haplotype block

In the methods where definitions of genetic loci or bins are used, haplotype structure and LD information are approximated by a predefined window that represent the given feature. An alternative is to use actual information of LD between variants. One way to do this is to expand all reported SNPs in each track with their LD variants, in effect representing the SNPs with
CHAPTER 3. METHODS

Figure 3.5: A bipartite graph representation of variants from two tracks. Each track is represented by a disjoint set of nodes, where the SNPs are nodes. The edges represent LD between them, and are weighted by the $r^2$ between the connected variants.

their true haplotypes. These haplotypes are further matched with each other using positional overlap.

To create haplotypes for each original SNP, all variants with $r^2$ above a predefined threshold are found and added to the track. Each SNP will be aware of the haplotype they belong to, and if SNPs within the same track are in LD, they will be mapped to the same haplotype feature. The expansion of SNPs to haplotypes is visualized in Figure 3.4, where black dots are the tag SNPs chosen to represent the full haplotype, and gray dots are other tag SNPs or expanded variants in LD with the haplotype SNP. The red curves show which SNPs are in LD with each other.

After all the original SNPs have been expanded, the features are given by haplotypes. A positive match is found between two haplotypes if at least two variants in the respective LD blocks are located at the same physical position. A visualization of this haplotype overlap is shown by dashed arrows between variants within haplotypes of different tracks in Figure 3.4. However, if two haplotypes have several overlapping variants between them, only one positive match is counted, illustrated with a red dashed arrow in the figure. The BTU summaries for $b$ and $c$ are given by the number of unmatched haplotypes in the different tracks.

By counting overlap between haplotypes once, intra-correlation of LD between features in the same track is removed. In that respect, the LD block matching can be seen as a different kind of filtering than the distance based filtering used for the definitions of genetic loci. As with the approximated genetic locus definitions, negative matches, $d$, are not defined for matching of haplotypes. This is because we only have knowledge of the haplotypes that stem from the tag SNPs in the compared GWAS data sets, and have no possibility of finding the haplotype structure for the rest of the genome.

Feature representation: LD matching score

With information of actual LD between variants, it is possible to discard information concerning positions altogether, as the correlation given by $r^2$ is not approximated or dependent on any physical properties of the tracks.
3.2. BINARY DATA REPRESENTATIONS

at hand. As shown in Figure 2.12, a set of SNPs can be represented as a graph, where variants in LD are connected with weighted edges of \( r^2 \). Consequently, it is possible to use algorithms of graph matching to find a matching score between two tracks.

A bipartite graph is a graph whose nodes can be divided into two disjoint sets \( U \) and \( V \), where every edge connects a node in \( U \) to a node in \( V \). No edges exist between nodes within the same set, and edges can be weighted. With this graph representation, a bipartite matching can be conducted, where the goal is to find the best overall score, given the weighted edges. A common example to illustrate the purpose of bipartite matching, is that of a group of boys and girls, and how interested each boy or girl is in the individuals of the other group. A score of interest is given for all pairs of boys and girls as edges, and the bipartite matching in this context aims to match each boy and girl, in such a way that the overall happiness of the group is maximized. A pair might be matched without them having the highest preference for each other, if that means another pair will get a match that overall yields a higher level of happiness for the two pairs.

The bipartite graph representation can be adapted for pairs of tracks, where the variants of each track form a set of nodes. If a pair of variants in different sets is in LD with each other, an edge weighted by the corresponding \( r^2 \) will connect their nodes. See Figure 3.5 for an illustration of a bipartite graph representation of two tracks. Variants with no LD can be excluded from the graph, as they have no possibility of contributing to a matching score.

For the bipartite graph of variants in LD, we want to find a match that maximize the overall sum of \( r^2 \). The BTU summary for \( a \) is given by the sum of \( r^2 \) weights for the best bipartite matching. Mismatch summaries, \( b \) and \( c \), are defined as SNPs without edges in the corresponding tracks.

The bipartite matching incorporate a partial filter of SNPs in LD within the same track. If a node, or SNP, is connected to multiple SNPs at the other track, only one of the connections is used in the overall score. At the same time, however, if both tracks have multiple SNPs in LD both within and between tracks, the optimal score will favor multiple matches within the same haplotype. Examples of both these cases of matching can be seen in Figure 3.6. Here, the same LD structure as in the haplotype representation of Figure 3.4 is shown, only with edges of correlation between the different tracks. To the right, the three variants in LD are matched only once. To the left, the five variants in LD are represented with two matches. Weights of \( r^2 = 0.8 \) for each of these matched pairs would result in an overall match score of 1.6 for the given haplotype, which would be favored over any single-edge match which at most would give a score of 1.

At first glance, the only partial filtering of the bipartite matching seems like a disadvantage of this method. However, LD between variants is not a transitive property when the correlation is less than 1. That is, if the variants rs1 and rs2 are in LD with \( r^2 = 0.8 \), and the variants rs2 and
CHAPTER 3. METHODS

Figure 3.6: Visualization of bipartite matching between SNPs of different tracks. Lines, either solid red or dashed gray, show which variants are in linkage disequilibrium. The optimal match found is shown in red.

<table>
<thead>
<tr>
<th>Property</th>
<th>Single base</th>
<th>Bins</th>
<th>Genetic loci</th>
<th>Haplotype</th>
<th>Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative matches</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>SNP distribution</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Filter</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Approximate LD</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Actual LD</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Table 3.2: Properties of the different binary feature representations. √ means presence of the property, and × absence.

rs3 are in LD with $r^2 = 0.9$, rs1 and rs3 are not guaranteed to be in LD with each other. Here, the best biological approximation could be a match between rs1 and rs2, and between rs3 and a completely different SNP it is in LD with, like that produced by the bipartite matching technique. Another possible beneficial property of the bipartite matching is that different SNPs located at the same genomic position, i.e. alternating variants, will be represented separately, and not collapsed into the same feature.

3.2.3 Properties of the different binary representations

The different feature representations presented in Section 3.2.2 have some combinatorial properties that can be used to describe in overview how they relate to each other. These combinatorics do not necessarily uniquely define all aspects of the different representations, but serve as a summary of the feature representations, and a starting point for discussion.

In Table 3.2, each column is a feature representation, except the leftmost, which contain different properties that characterize the methods. “Single base” refers to the binary vector representation where each feature is a single genomic position. “Bins” refers to binary vector representation where each feature is a bin. “Genetic loci” refers to the feature representations of a genetic locus surrounding a tag SNP, both the block and Gaussian definitions. “Haplotype” refers to the LD expansions of all tag SNPs, and subsequent matching of expanded LD blocks. Lastly, “Graph” refers to the technique of bipartite matching scores.

The first property of the table is “Negative matches”, which refers to whether or not $d$, the BTU summary for negative matches, is well defined. The feature representations with this property are the ones that are defined...
3.2. BINARY DATA REPRESENTATIONS

as vectors, i.e. single bases and bins. The property “SNP distribution” refers to whether or not information of the actual distribution of SNPs within a track is used in the feature definition. All feature representations inhabit this property, except the bins, which are defined in advance. The predefined window of a bin is uniformly distributed across the genome, and does not use any information of the actual SNPs in the given data set. “Filter” refers to whether any filtering is performed with respect to intra-correlation and SNPs in LD within a track, and is done in bins by reducing all hits within a bin to 1, in genetic loci by keeping only the most significant SNP within a genetic locus, and in the haplotype matching by defining the features as an expanded set of SNPs in LD. “Approximated LD” is set for bins and genetic loci, which both use a predefined threshold to account for LD between closely located SNPs. The haplotype and bipartite matching methods use actual information of LD, as computed between the variants.

3.2.4 Similarity measures

When BTU summaries are defined for a specific binary feature representation, the information of positive matches, negative matches and mismatches can be used to compute different similarity or distance measures directly. In their article “A Survey of Binary Similarity and Distance Measures”, Choi, Cha, and Tappert, (2010) present 76 well known binary similarity and distance measures using $a$, $b$, $c$ and $d$. Some of the similarity measures presented are listed below:

\[
S_{\text{Jaccard}} = \frac{a}{a + b + c} \quad (3.2)
\]

\[
S_{\text{Simpson}} = \frac{a}{\min(a + b, a + c)} \quad (3.3)
\]

\[
S_{\text{Cosine}} = \frac{a}{\sqrt{(a + b) \sqrt{(a + c)}}} \quad (3.4)
\]

\[
S_{\text{Otsuka}} = \frac{a}{\sqrt{(a + b)(a + c)}} \quad (3.5)
\]

\[
S_{\text{Sorgenfrei}} = \frac{a^2}{(a + b)(a + c)} \quad (3.6)
\]

\[
S_{\text{Kulczynski}} = \frac{\frac{1}{2}(2a + b + c)}{(a + b)(a + c)} \quad (3.7)
\]

\[
S_{\text{Forbes}} = \frac{na}{(a + b)(a + c)} \quad (3.8)
\]
CHAPTER 3. METHODS

\[ S_{\text{McConnaughey}} = \frac{a^2 - bc}{(a + b)(a + c)} \]  \hspace{1cm} (3.9)

\[ S_{\text{Pearson}} = \frac{ad - bc}{\sqrt{(a + b)(a + c)(d + c)(d + b)}} \]  \hspace{1cm} (3.10)

The formulas (3.2)–(3.7) above output a value between 0 and 1, where 1 means full similarity in terms of the given definition, and 0 means no similarity at all. The Jaccard similarity coefficient (3.2) is a widely used measure of similarity, and is defined as the intersection of two sets, divided by the union. The formulas of Simpson (3.3), Otsuka (3.5), Sorgenfrei (3.6) and Kulczynski (3.7) are variations over the same properties as used for the Jaccard measure. In an experimental analysis of the measures, in terms of performance on a randomly drawn set of binary data, Choi, Cha, and Tappert, (2010) found these measures to be related.

Forbes, given in formula 3.8 is not directly a measure of distance or similarity, but a ratio of observed overlap compared to the expected overlap. A value of 10 means it overlaps 10 times more than what is expected given independence, while a value of 0.01 means it overlaps 10 times less than expected.

The McConnaughey (3.9) and Pearson (3.10) similarity measures output a value between -1 and 1. These definitions describe how the data subject to comparison are linearly dependent. McConnaughey measures to what extent positive matches occur compared to mismatches. Here, a comparison with only positive matches will result in a value of 1, a comparison with only mismatches will result in a value of -1, and an even distribution of positive matches and mismatches will result in 0. Pearson measures to what extent matches, both positive and negative, occur compared to mismatches. If \( a + d = b + c \), the correlation is 0, if there are only matches, the correlation is 1, and if there are only mismatches, correlation is -1.

Note that Forbes and Pearson are defined for BTU summaries that include negative matches. Consequently, many of the feature representations cannot be compared with these coefficients.

### 3.2.5 Standardized measures of distance

For clustering, a standardized measure of distance is desired, where 0 means no distance, or full similarity, and 1 means full distance, or no similarity. As the formulas (3.2)–(3.7) output a similarity value between 0 and 1, they are standardized by subtracting them from 1. The correlation measures of McConnaughey (3.9) and Pearson (3.10) are converted into distances by computing \( 1 - \frac{1 + \text{corr}}{2} \). This conversion turns negative
correlation into high distance, and positive correlation into low distance. A neutral correlation of 0 is converted into a distance of 0.5.

The Forbes measure (3.8) stand out from the other measures, in that it indicates the proportion of observed matches relative to what is expected, and does not output a value between 0 and 1. As we are interested in overlap that happens more often than expected, the following formula is used to convert it into a measure of distance:

\[ \frac{1}{\max(1, \log(S_{Forbes}) + 1)} \]  

(3.11)

Here, if Forbes overlap less than expected, i.e. Forbes < 1, the distance is set to 1. For a behavior where the distance moves slower towards 0, than for instance if we were to divide by \( \max(1, S_{Forbes}) \), we use a logarithmic scale for Forbes before division.

### 3.3 Continuous data representations

Rather than using a binary data representation for GWAS, it is possible to create vectors of continuous values that describe specific attributes of interest for the disease-associated data set. The following section describes two different ways for representing the data as such continuous vectors. Measures of correlation and distance that can be used with the continuous vector representations are also discussed.

#### 3.3.1 Definitions of continuous features

We suggest two different ways of representing tracks as continuous vectors. The first make use of the values associated with SNPs, for instance p-values or log transformed p-values. The other is the vector representation described in Section 2.3.2, where the features are genetic overlap between a given track to other tracks.

**Vectors of effect at overlapping positions**

We can represent a track as a vector where the features are continuous values that describe a specific attribute of the SNPs. GWAS data sets contain variants associated with the disease under investigation, meaning we have no guarantee that the SNPs reported are the same for different diseases. Given two sets of SNPs, one from each GWAS, we are left with two main options for vector representation. We can create a vector where all SNPs in the union of the set are represented as their own feature, or restrict the features to only represent SNPs present in both tracks. An issue with the first approach is that for SNPs only present in one of the diseases,
CHAPTER 3. METHODS

the corresponding feature of the other track must be set to a specific value. We call this issue “the missing value problem”, as we do not know whether or not the absence of the given SNP in the other track is the equivalent of it having no effect, or if the effect simply has not been studied or reported. Therefore, no neutral value can be inserted for the missing values, without the possibility of introducing noise.

To avoid the missing value problem, the vectors are constructed from the intersection of SNPs in the given data sets. This means that the features present in the two constructed vectors may vary between different pairs of GWAS data.

This method is applicable to many different kinds of values related to SNPs. It is important to note, however, that different types of values might describe entirely different properties of the associated SNPs. Therefore, when deciding on the values to represent the data by, it is important to be aware of which relationships one wish to analyze.

Vectors of overlap with other data sets

Rather than using the actual values of the GWAS data sets, each feature can be a value of computed overlap to other data sets, much like the overlap defined in the section of genetic loci of block size (Section 3.2.2). In this approach, the same filtering of each data set is done, and only the most significant SNPs within a genetic locus are kept for analysis. Genetic overlap between tracks is defined as two tag SNPs on different data sets, represented by their positions \( x \) and \( y \), where \( \lvert x - y \rvert \leq \frac{\text{window}}{2} \).

For all pairs of diseases, the total genetic loci overlap is found, and a matrix of all pairwise counts constructed. Using this matrix, a disease is now represented as a continuous vector, where its features are the genetic overlap to other diseases in the analysis. For all pairwise comparisons between vectors, we remove the features where the diseases overlap with themselves.

3.3.2 Correlation coefficients

With a defined vector representation for pairwise comparisons between GWAS data sets, a correlation coefficient can be applied to each pair of vectors. The goal of the coefficient is to assess whether or not a functional relationship exists between the features of the vectors. These correlation coefficients, applied to continuous data, are different from that of the binary application seen previously (Section 3.2.4), as a continuous data representation can exhibit more complex relationships with other data sets than a linear one. Correlation in this context refers to whether or not the features of two vectors co-vary, and the value of association, i.e. the coefficient, represents the strength and direction of the functional relationship between the tracks (Mukaka, 2012). Two different
correlation coefficients is used to measure association of the GWAS vector representations, the Pearson product-moment correlation coefficient and Spearman’s rank correlation coefficient.

\[
C_{\text{Pearson}} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}}
\]  

(3.12)

The Pearson product-moment correlation coefficient, Pearson for short, assesses whether or not there is a linear correlation between two data points. Its formula is given in (3.12), where \(\bar{x}\) and \(\bar{y}\) represent the sample mean of the respective vectors. Spearman’s rank correlation coefficient, Spearman for short, describes the degree of similarity between two rankings. Instead of assuming a linear relation, it measures how well the relationship can be described with any monotonic function. Spearman is defined as the Pearson correlation coefficient, but on rank transformation of the values in each vector.

Values of positive correlation indicate that there exists a positive relationship between the two objects, where an increase or decrease in one object tends to co-occur with a similar movement in the other. Negative correlation, on the other hand, also indicates a relationship, but the influence is in opposite directions, where values of one object increase as the other decrease. Values of 0 mean there is no correlation between the data.

One consideration that must be taken into account, is the resulting vector sizes of the data sets. As can be seen from equation 3.12, the coefficient is undefined for samples with zero standard deviation, and the sample must exceed two SNPs to give any information of an actual relationship. In addition, Pearson is sensitive to outliers and generally assumes a normal distribution of the data to work as expected. Spearman does not suffer from the same issues, as it converts all data to ranks before assessing whether or not a monotonic relationship exist (Mukaka, 2012).

### 3.3.3 Standardized measures of distance

As with the binary measures that output values between -1 and 1, we standardize Pearson and Spearman to a distance measure by \(1 - \frac{1+\text{corr}}{2}\).
Chapter 4

Implementation

The different methods for GWAS representation and comparison presented in the previous chapter are implemented as a tool suite for cluster analysis of diseases and traits. This chapter addresses implementation choices, such as frameworks, programming language and data representations. The different tools are described in terms of the implemented code.

4.1 Languages and frameworks

4.1.1 The Genomic HyperBrowser

The suite of tools for disease comparison is made available through the Genomic HyperBrowser. The HyperBrowser is developed with most of the code patterns and HyperBrowser functionality in the Python programming language. Consequently, Python is the chosen language for development in this thesis.

Python is popular programming language within the field of bioinformatics and scientific computing, and is open source, freely available, and distributable. It is a high-level, general-purpose programming language, and allows fast development of flexible and portable code. Python is an interpreted programming language, with the drawback that it is slower than compiled and lower-level languages.

4.1.2 Python libraries

Many initiatives exist that extend the main Python functionality distributed with the basic installation. Examples of extensions include Python packages developed for the purpose of solving problems within the domains of mathematics, science and engineering. The core libraries used

\(^1\)https://www.python.org/about/
in this thesis are SciPy\textsuperscript{2}, Numpy\textsuperscript{3}, Matplotlib\textsuperscript{4} and Seaborn\textsuperscript{5}. All packages are part of or built on other libraries from the SciPy stack, a collection of Python software for scientific computing.

The methods presented for GWAS feature representations and comparisons form the basis for the cluster analysis. SciPy and Numpy are used in this thesis to implement the methods, where the clustering functions provided in SciPy, and array transformations and computations provided by NumPy are of particular interest. Matplotlib is a 2D graphics package for publication-quality image generation (Hunter, 2007). Seaborn is a Python library based on Matplotlib. It provides out-of-the-box plotting functions for drawing visually pleasing statistical graphics, and is used for most of the resulting plots of the clustering tools.

### 4.1.3 Perl

Perl is a general-purpose programming language used amongst other things for web development and network programming. The Ensembl APIs provide most of their features through a Perl interface. Therefore, some Perl scripts which use the Ensembl APIs have been developed.

### 4.2 LD computation

#### 4.2.1 Scripts for finding variants in LD

Two Perl script for LD computation with the Ensembl API has been created, along with some Python scripts for automation of the LD computation process. The LD computation can be done for either hg19 or hg38, depending on the script used. All the scripts are available on GitHub, see Appendix C.2.

The scripts for LD computation takes as input a list of tag SNPs given by their rsIDs. For each tag SNP, the script finds variants in LD within a window of 1Mb, i.e. 500kb in each direction from the given SNP. The population chosen for LD calculation is CEU. All pairs of variants in LD is written to file in the following tab-separated format:

\[
\begin{array}{ccccccc}
\text{chrnum} & \text{pos}_{\text{ldSNP}} & \text{ldSNP} & \text{pos}_{\text{tagSNP}} & \text{tagSNP} & r^2 \\
\end{array}
\]

\textsuperscript{2}http://docs.scipy.org/doc/scipy/reference/
\textsuperscript{3}http://docs.scipy.org/doc/numpy/
\textsuperscript{4}http://matplotlib.org/contents.html
\textsuperscript{5}https://stanford.edu/~mwaskom/software/seaborn/index.html
CHAPTER 4. IMPLEMENTATION

4.2.2 A master file of variants in LD

The scripts using the Ensembl API for LD calculation are time consuming and must be run locally on a computer external to the HyperBrowser. Because of this, a master file containing LD information for all significant SNPs extracted from the GWAS Catalog via the HyperBrowser was created. The purpose of this master file is to provide lookup functionality for LD information of SNPs in the GWAS Catalog.

The list of all significant SNPs were passed to the LD computation script which use the hg38 reference genome. Further, all variants in LD with the input SNPs were computed, \( r^2 > 0 \). The result was a file with all LD pairs, where at least one of the variants in the LD pair is present in the GWAS Catalog.

Note that this master file is large, with ~13 million LD pairs. No thresholding was done in the LD computation, as the Ensembl API call for finding variants in LD calculate \( r^2 \) on the fly between all variants in the available window, and filtering out low \( r^2 \) values will not improve computation time. Any thresholding can be done faster afterwards by simple text parsing of the resulting LD file, where a lower bounds of \( r^2 \) can be set. See Appendix B.3 for download of the data set.

4.3 Tools for data modification and creation

Several tools have been created for this thesis, with the purpose of modifying or creating new data from a given GSuite of GWAS data. The tools and implementation choices are described in the following sections.

4.3.1 Genomic liftover

If coordinates are important in an analysis of different tracks, the same reference genome should be used for all track elements. Some genomic resources for annotation data only use the hg19 reference genome, and to compare tracks generated from these sources with data mapped to hg38, tracks might need to be lifted over to the other reference genome. The tool “GSuite liftover” is implemented for this purpose.

The tool use static data files from dbSNP, stored on the HyperBrowser server, which can be used to map SNP track elements, given by their rsIDs, to the correct reference genome position. The specific dbSNP files used for the liftover are described in Appendix B.1. The 1-indexed header variable of each track must be set to False, as the dbSNP coordinates start at 0.

The input of the tool is a GSuite and the reference genome the tracks should be mapped to. The tool then parses through the GSuite tracks, and given that they have a column header “snp”, which contains rsIDs for all track
4.3. TOOLS FOR DATA MODIFICATION AND CREATION

elements, it will remap each track element to the correct position on the chosen reference genome. The rest of the track element attributes remain untouched.

4.3.2 From sumstats to GTracks

Collections of files with the sumstat format given in Section 2.6.4, can be uploaded as a GSuite, but need conversion into a standardized track format before they can be used in the HyperBrowser analyses. The tool “Convert GSuite sumstat tracks” is implemented for this purpose.

The tool requires that the tracks of the GSuite given as input has a column headed by either “P” or “Z”, which contains information of association signal or effect size, respectively. Additionally, it requires a column headed by “snp”, containing rsIDs that identify the SNP in each row. The sumstat track of the GSuite will then be converted into a valued point track, where the value is the p- or z-values.

The user is asked to specify which reference genome the new track is annotated with. The static files of dbSNP positions, used in the genome liftover tool, is also used in the sumstat conversion to find the appropriate chromosome and coordinate for each SNP, based on their rsIDs.

The tool allows for a lower threshold to be set, which can be used to filter out higher values. The main object of this option is to allow filtering of p-values. If defined, only values below the given significance threshold will be present in the new GSuite. An additional option to log-transform the values, which could be advantageous for tracks with p-values in some analyses, is also available.

4.3.3 LD track generation

The tool “LD Track generation” can be used to create a linked points track, the format of which is described in Section 2.5.3, from the master file described in Section 4.2.2. The master file of LD expansions has been uploaded to the HyperBrowser server as a static file.

The tool takes as input a GSuite of tracks with significant SNPs, as found in the GWAS Catalog, and a lower threshold for $r^2$. For the LD track generation, the original tracks need a column headed by “snps”, which contain the SNP rsIDs for all track elements. A list of all the unique rsIDs for the entire GSuite is extracted within the tool, and passed to a function that finds all corresponding variants in LD in the master LD track. This information is further used to create a linked point track where each row correspond to a SNP in the LD graph, either a tag SNP or an expanded LD SNP. The edges of each track element show which variants it is in LD with.

An additional Python script for LD track generation is described in Appendix C.2. The script takes as input a file of LD expansion information
formatted as the master LD track of Section 4.2.2, and a threshold for \( r^2 \). With this information the LD file is converted to a linked point track. This script for linked point creation can be used if a different LD data set than the one stored in the HyperBrowser is needed.

### 4.3.4 Expanding a GSuite with LD

The tool “LD GSuite generator” is not used in the methods of this thesis, but provide an additional way to incorporate information of LD between variants in a GSuite. The tool is similar to the tool for LD track generation, in that it takes as input a GSuite, whose tracks have a column headed by “snps”, which contain SNP rsIDs. It also makes use of the static master file of LD expansions stored on the server, and has the same options for a lower threshold of \( r^2 \).

Rather than creating a general LD track, however, the tool expands all the primary tracks in the input GSuite, to also include points that are in LD with the original track elements. The resulting expanded GSuite contains point tracks only, as the expanded variants are not associated with any p-values, as associated GWAS SNPs are.

### 4.4 Tools for empirical exploration

Three tools have been created with the purpose of empirical exploration of track properties in a given GSuite.

#### 4.4.1 Empirical exploration of point track properties

The tool “Empirical properties of point tracks” uses HyperBrowser statistics already implemented, such as PointGapsStat and NearestPointDistsStat to find the smallest distances for all points in a GSuite, within and between tracks, respectively. These distances are further visualized in different plots.

#### 4.4.2 Empirical exploration of LD tracks

The tool “Empirical properties of LD tracks” takes in a linked point track of LD information, for instance created by the “LD Track generation” tool. It makes use of the functions in the LDExpansions class, and for all pairs in LD, stores information of the physical distance between them. This information is further used to plot distributions of pairs of variants in LD, as a function of the distance between them, given a threshold of \( r^2 \).
4.5. Binary feature representations

This section describes how the different methods for binary feature representations, presented in Section 3.2, have been implemented in the HyperBrowser.

4.5.1 Single base positions

For the single base features, the statistic DistanceMetricsFoundationStat is used to compute $a$, $b$, $c$ and $d$ for pairs of tracks. The two tracks are represented by lists of unique SNP positions, for each chromosome the local analysis is computed for. Thus, for each chromosome, we can calculate the single base matches and mismatches with NumPy set operations. In the following code, $\text{snps1}$ and $\text{snps2}$ represent different lists of chromosome positions:

```python
intersectSNPs = np.intersect1d(snps1, snps2)
a = intersectSNPs.size
b = snps1.size - intersectSNPs.size
c = snps2.size - intersectSNPs.size
d = chromosomeLength - a - b - c
```

4.5.2 Bins

The implementation of binary features of bins first create a uniform track representation of vectors, where each feature is a bin. It makes use of the already implemented HyperBrowser statistic PointCountPerMicroBinV2Stat, which generate a feature vector for the entire track. Each feature is given by the number of points within the corresponding bin. With this approach, each track is uniformly represented, i.e. have the same length and order of features. Consequently, for all pairwise comparisons of vectors, we can use NumPy masks and vector operations to calculate $a$, $b$, $c$ and $d$ across the entire genome. In the following code, $\text{bins1}$ and $\text{bins2}$ represent the different tracks, given by their feature vectors:

```python
nonzero_intersect = nonzero((bins1 != 0) & (bins2 != 0))
nonzero1 = nonzero(bins1)
nonzero2 = nonzero(bins2)
a = len(bins1[nonzero_intersect])
```
4.5.3 Genetic loci

For the representation of features as genetic loci, a preliminary filtering is conducted on each track. We extract two vectors for the tag SNPs in a track, one with positions, the other with p-values, and can traverse the SNPs in increasing order of positions. Consecutive SNPs within the predefined size for the genetic locus, are filtered to only keep the most significant SNP. As mentioned in Section 2.6.3, the values of GWAS Catalog tracks are defined as $-\log(pval)$. Therefore, the track element with the highest value is the one kept for analysis.

\[
\begin{align*}
b &= \text{len}(\text{bins1}[:\text{nonzero_snps1}]) - a \\
c &= \text{len}(\text{bins2}[:\text{nonzero_snps2}]) - a \\
d &= \text{len}(\text{bins1}) - a - b - c
\end{align*}
\]

The increasing order of positions can also be used when we compute overlap of genetic loci between two tracks. Rather than a double for loop, an index can be used for traversal of the two tracks, which is updated to point at the lowest unseen positions in each track. This results in a worst case time complexity of $O(N + M)$ for computing the overlapping loci, where N and M are the two track lengths. The SNP traversal is shown in the simplified example below, where `snp1` and `snp2` are vectors of chromosome positions, one from each track.

```python
while i < len(track1) and j < len(track2):
    if abs(snp1 - snp2) <= window/2:
        a += 1
        i += 1
        j += 1
    elif snp1 < snp2:
        b += 1
        i += 1
    elif snp1 > snp2:
        c += 1
        j += 1
```

For the Gaussian definition of genetic loci, all that is changed is the way $a$, $b$ and $c$ is updated in the event of two SNPs mapping to the same genetic locus. Instead of increasing $a$ with 1, the Gaussian function described in Section 3.2.2 is used:
4.5. BINARY FEATURE REPRESENTATIONS

\[
\begin{align*}
a &= \exp\left(- \frac{((snp2 - snp1) \times 2)}{2 \times (\text{window} / 4 \times 2)}\right) \\
b &= \frac{(1 - a)}{2} \\
c &= \frac{(1 - a)}{2}
\end{align*}
\]

The block and Gaussian block definitions are implemented with the statistics DistanceBlockFoundationStat and DistanceFuzzyFoundationStat, respectively.

### 4.5.4 Haplotype blocks

To expand all tag SNPs into haplotypes, an additional linked point track of LD structure for the track elements is used. This LD structure is accessed with the use of GraphStat objects. The original point tracks that are to be compared, need a column of “snps” with rsIDs for all track elements.

Before comparing all pairs of tracks, each track is expanded to include variants in LD. The haplotypes are implemented by representing each track as a dictionary that map positions to rsIDs. Here, all tag SNPs from the original track, as well as the LD expanded variants are stored with their positions as keys, and an rsID as value. The rsID each position map to, denote the haplotype the given SNP is part of, and variants in LD will map to the same rsID. The statistic ExpandWithLDVariantStat is used for this dictionary representation.

In the statistic ExpandTrackAndMatchStat, matches between the haplotype dictionaries are computed. Here, \(a\) is the unique count of pairs of haplotype rsIDs found for SNPs at the same positions. Mismatches, \(b\) and \(c\), are computed as unmatched haplotypes in the respective tracks, given by the count of unique haplotype rsIDs in the track dictionary, minus the matched haplotypes in \(a\).

### 4.5.5 Bipartite matching scores

The problem of finding a matching of a bipartite graph that maximize the overall weight of edges between the matched nodes, can be seen as an optimization problem. This optimization problem, often referred to as a linear assignment problem (LAP) with extensions of overall maximization or minimization of weights, is solved by many different algorithms. Greedy implementations are the fastest, but are based on heuristics, and thus susceptible to local optima. The optimal algorithms are guaranteed to find the highest matching score, but have a worst case time complexity of \(O(N^3)\), where \(N\) denote the total number of nodes. Both a greedy and optimal variant have been implemented to solve the graph matching problem.

The bipartite graph between two tracks is implemented as a cost matrix, where the rows and columns represent the tag SNPs in each track.
CHAPTER 4. IMPLEMENTATION

<table>
<thead>
<tr>
<th></th>
<th>rs1</th>
<th>rs2</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs3</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>rs4</td>
<td>0.8</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.1: Minimal example of cost matrix, which will produce different results for greedy and optimal implementation for a bipartite matching score respectively. If two tag SNPs, or nodes, in different tracks are in LD with each other, the value of the given row and column is set to the corresponding \( r^2 \) for the two variants. The full \( N \times M \) cost matrix, where \( N \) and \( M \) are the SNP count of the respective tracks, represent all LD connections between the two tracks.

The tool for bipartite matching have not made use of the HyperBrowser statistics for the optimization problem. Instead, the LD graph structure is implemented as a dictionary where pairs of SNPs in LD, or edges, are stored as keys with their \( r^2 \) as value. The functionality for this graph representation is implemented in the class LDExpansions. Each track is represented by a list of all its rsIDs, and for all pairs of tracks, a cost matrix is generated using the LD graph. The cost matrix generation, and other functionality for bipartite matching, is located in the class BipartiteMatching.

The greedy algorithm process the edges in decreasing order of \( r^2 \). Each processed \( r^2 \) is added to the overall matching score of the bipartite track graph. For each added score, the entire column and row of the given match is zeroed out, so that the two matched SNPs will not be matched with any other nodes. When there are no edges left to process, the algorithm terminates. Table 4.1 show a minimal example of a cost matrix, where the greedy algorithm would give a suboptimal matching score. In the cost matrix, the nodes of rs1 and rs3 will be matched with each other, as they have the highest \( r^2 \). Overall, however, a better score would be computed if rs1 was matched with rs4, and rs2 with rs3. Pseudo code for the greedy implementation follows:

```python
while <edges left to process>:
    maxEdge = max(costMatrix)
    <set maxEdge.row in costMatrix to zero>
    <set maxEdge.col in costMatrix to zero>
    a += maxEdge.val
```

Many software packages out there, implemented in a variety of languages, claim that they have implemented an optimal solver for the optimization problem of finding the overall maximal weight. However, the Python packages tested, such as munkres, did not find the optimal solution for all pairwise comparisons of tracks, and was at times outperformed by the greedy implementation.

Therefore, an optimal algorithm was implemented from scratch in the
4.6. CONTINUOUS VECTOR REPRESENTATIONS

HyperBrowser in the class JonkerVolgenant. The optimal algorithm is the LAPJV, a shortest augmenting path algorithm for dense and sparse linear assignment problems (Jonker and Volgenant, 1987). The algorithm is extended to solve the minimum cost flow problem, i.e. minimize weights. Its worst time complexity is $O(n^3)$, but as Jonker and Volgenant, (1987) show in their article, this implementation outperform many similar solutions, in terms of actual computational time.

The LAPJV algorithm solves a minimization problem, where the cost matrices are square. Therefore, the implementation has an initial step where the cost matrices are padded with zeros to be square, and inverted to fit a minimization problem. The actual implementation is the exact same as the pascal code proposed by Jonker and Volgenant, (1987), but translated to Python. The non-square cost-matrixes are handled as suggested in an existing Java implementation.

For both the greedy and optimal implementation, $a$ is set to the overall matching score between two tracks. $b$ and $c$ is set to number of nodes in the respective tracks without any edges.

4.5.6 Pairwise comparison

The similarity measures proposed in Section 3.2.4 is implemented for all the different binary representations, with the exception of Forbes and Pearson. As Forbes and Pearson make use of $d$, either directly or through the overall feature count $n$, they require a well defined binary feature vector representation, more specifically a representation where negative matches are computed. The single base and bin features are the only representations where $d$ is defined, and comparison with Pearson and Forbes are restricted to these feature representations.

For each pairwise comparison of tracks, the BTU summaries $a$, $b$, $c$ and $d$ are computed and used with a similarity measure to find the distance between the tracks. For the similarity measures that output a value between 0 and 1, i.e. all except Pearson and McConnaughey, they are converted to distance as described in Section 3.2.5. Pearson and McConnaughey, which produce values between -1 and 1, are stored as is. Section 4.7 describes how these values are further used in the cluster analysis.

4.6 Continuous vector representations

This section describes how the different ways to represent tracks as continuous vectors, presented in Section 3.3, have been implemented in the HyperBrowser.

6https://github.com/dberm22/Bipartite-Solver
4.6.1 Vectors of values at overlapping positions

For a vector representation of values at overlapping positions, each feature corresponds to the value of a SNP present in the track. The features only represent SNPs that are present in both tracks that are to be compared. The statistic \texttt{OverlappingValsListStat} takes as input two tracks and return two vectors with values for the same feature representation.

In this vector creation, multiple reported SNPs at the same position, with different p-values, present an issue. A statistic can be set up to get track data for unique positions only, but in the case where the values of the track elements are numbers, SNPs with the same position will get a value of \texttt{nan}. Therefore, all reported SNPs at the same position must be represented separately. As each feature of the resulting vectors only contain information of one SNP, in the case where a SNP is reported multiple times, it is represented as the highest value of the overlapping SNPs.

4.6.2 Vectors of overlap with other tracks

For a vector representation where the values are overlap with other tracks, the features are the number of overlapping loci between the given track and all other tracks. First, a preliminary filtering of all tracks is done, like in the binary feature representation of block genetic loci (Section 4.5.3). Then, overlap between all pairs of tracks are computed to generate a track-by-track overlap matrix. The statistic \texttt{GeneticLociOverlapStat} can be used for all pairs of tracks, and return the number of overlapping genetic loci between the two tracks.

A disease-associated data set is represented as a vector where the features are the genetic overlap count to all other tracks, as given in the track-by-track overlap matrix. Before pairwise comparisons between the vectors are conducted, we remove the features where the tracks have values of overlap with themselves.

4.6.3 Pairwise comparison

For each pairwise comparison of tracks, Pearson or Spearman is applied to the pair of continuous vectors that represent the given tracks. The SciPy function \texttt{stats.pearsonr} can be used for Pearson, and \texttt{stats.spearmanr} for Spearman. Both these functions takes as input two data sets which are vectors of the same length. They output a correlation coefficient and a p-value which test against a null hypothesis of uncorrelated data. The datasets should be larger than 500 for the p-values to be reasonable, according to the package documentation. The correlation functions output values between -1 and 1, which are stored as is. Section 4.7 describes how these values are further used in the cluster analysis.
4.7 Disease-by-disease comparison

All tools for comparison of diseases and traits are implemented with a GSuite as input. This GSuite contains all the diseases and traits one wish to compare, in the form of VP tracks. Each tool computes a value for all pairwise combinations within the GSuite, which is stored in a triangular distance matrix. This matrix is further used to cluster the diseases and produce a dendrogram which show the relationships between the tracks. The global results for all pairwise comparisons are computed for the genome-wide level, and local track analyses are done on bins the size of chromosomes.

4.7.1 Triangular matrix creation

To improve run time of the tools, comparison is done once for each unique pair of tracks, and a triangular matrix created where each feature is the result of a pairwise comparison. In Sections 4.5.6 and 4.6.3, the different possible results from a pairwise comparison, given a specific data representation and comparative measure, are presented. The comparisons that use Jaccard, Simpson, Cosine, Otsuka, Sorgenfrei, Kulczynski and Forbes to compute a measure of similarity, are stored as distances in the triangular matrix. The comparisons that use McConnaughey, Pearson or Spearman, are stored as correlation coefficients between -1 and 1.

4.7.2 Heatmap

In all tools, a heatmap is shown for each disease-by-disease cluster analysis, containing all pairwise comparisons in terms of similarity or correlation. The triangular distance matrices is therefore converted to similarity by subtracting all values from 1 before they are plotted.

The SciPy function `spatial.distance.squareform` is used to make all the triangular matrices square, and the diagonal, where the tracks are compared with themselves, is set to 1. The matrices that contain any negative values is displayed with a heatmap that range from -1 to 1, where negative values are displayed in blue, positive in red, and the neutral value of 0 is shown in white. The matrices with purely positive values is represented with a heatmap that range from 0 to 1, where positive values are red, and 0 is white. The value of 1 in the heatmap indicate full positive correlation or similarity.

The order of the GSuite tracks decide the order of diseases and traits in the rows and columns of the heatmap.
4.7.3 Clustering

The hierarchical clustering functions of SciPy is used for clustering, in particular `scipy.cluster.hierarchy`, where the functions `linkage` and `dendrogram` is located.

The `linkage` function takes in either a set of observations in the form of vectors or a precomputed distance matrix, and output the result of a hierarchical clustering in the form of a linkage matrix. The caveat of this flexibility is that the function seems to misinterpret redundant, i.e. square, similarity matrixes as observation vectors\(^7\).

The linkage criteria available for the linkage function are single, complete, average (UPGMA), weighted (WPGMA), centroid, median and ward. Appendix A describe technical details concerning these criteria, and the first four are implemented in the linkage package as described in the appendix. The centroid, median and ward criteria are not appropriate in our setting, however, as the two first only work with observation data, and the ward metric is limited by an assumption of initial distances proportional to squared euclidean distance. Therefore, only single, complete, average and weighted will be available for the clustering done in the tools.

For the clustering with `linkage`, a triangular matrix of distances is needed. Consequently, the triangular matrixes with values between -1 and 1 is converted to a distance matrix by computing \(1 - \frac{1 + \text{corr}}{2}\) for each value. The `dendrogram` function is further used to plot the hierarchical clustering the linkage function outputs.

4.7.4 Tool overview

The code for all tools implemented in this thesis, along with scripts for LD computations, can be found at the GitHub repository described in Appendix C.1. All the code, except the LAPJV code, which have been translated from the original pascal implementation, have been developed for the purpose of this thesis. Some of the tools and statistics make use of other HyperBrowser functionality, for instance the general interface to run analyses from tools, and statistics developed by others. These are not included in the GitHub repository.

To a great extent, each feature representation have been implemented as its own tool. The reason for this choice is that it makes a clear separation of different ways to represent and cluster data, and each tool is limited to provide options for the given representation. Consequently, the tool code does not need to account for a multitude of combinatorics for different

---
\(^7\)This issue has been reported more than once: https://github.com/scipy/scipy/issues/5508
4.7. DISEASE-BY-DISEASE COMPARISON

<table>
<thead>
<tr>
<th>Feature representation</th>
<th>Toolname</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single base</td>
<td>Binary vector clustering</td>
<td>Binary ((a, b, c, d))</td>
</tr>
<tr>
<td>Bins</td>
<td>Binary vector clustering</td>
<td>Binary ((a, b, c, d))</td>
</tr>
<tr>
<td>Genetic locus (block)</td>
<td>Genetic loci clustering</td>
<td>Binary ((a, b, c))</td>
</tr>
<tr>
<td>Genetic locus (Gaussian)</td>
<td>Genetic loci clustering</td>
<td>Binary ((a, b, c))</td>
</tr>
<tr>
<td>Haplotype block</td>
<td>Haplotype block clustering</td>
<td>Binary ((a, b, c))</td>
</tr>
<tr>
<td>LD matching score</td>
<td>Bipartite matching of point tracks</td>
<td>Binary ((a, b, c))</td>
</tr>
<tr>
<td>Overlapping values</td>
<td>Correlation of track values</td>
<td>Pearson / Spearman</td>
</tr>
<tr>
<td>Overlap with other tracks</td>
<td>Correlation of overlap with other tracks</td>
<td>Pearson / Spearman</td>
</tr>
</tbody>
</table>

Table 4.2: The different feature representations and the tools they are available from. The column to the right show which measures for comparison are available for the given representation.

parameters, with the result of a clean implementation that hopefully is easy for other developers to familiarize themselves with.

Table 4.2 shows an overview of the tools for disease comparisons, implemented in the HyperBrowser. It indicates which feature representations and comparative measures are found in the different tools. Here, the representations of bins and single bases have been implemented in the same tool, as they both reflect binary vector representations of the data. The two different definitions of genetic loci are also available from the same tool, as the only difference between them is the way the Gaussian definition weighs down overlap between distant SNPs. The rest of the representations are implemented in their own tools.
Chapter 5

Results

In thesis we have presented a range of different methods for feature representation and comparison of diseases and traits, as represented by associated SNPs from GWAS. The methods have been implemented in a tool suite for clustering of disease-associated data sets. The tools are described in the following sections in terms of user interface and purpose. We also show examples of utilization through a use case of seven diseases and traits, and the results are presented.

5.1 A suite of tools for comparison of diseases

The tools can be found at https://hyperbrowser.uio.no/clustering/. The user interface of the implemented tools are listed below:

- **Clustering of binary representations**
  - Binary vector clustering
  - Genetic loci clustering
  - Haplotype block clustering
  - Bipartite matching of point tracks

- **Clustering of continuous vectors**
  - Correlation of track values
  - Correlation of overlap with other tracks

- **Empirical exploration tools**
  - Empirical properties of point tracks
  - Empirical properties of LD tracks
  - Print rsIDs of tracks in GSuite

- **Create and modify data**
5.1. A SUITE OF TOOLS FOR COMPARISON OF DISEASES

- GSuite liftover
- Convert GSuite sumstat tracks
- LD track generator
- Expand GSuite with LD

5.1.1 Main purpose

The main purpose of the tool suite is to enable researchers to cluster disease-associated genomic tracks. In this thesis, GWAS SNPs represent the features of genomic tracks, but any source of genome-wide SNP data can be used for the analyses.

All tools for comparison of diseases accept a GSuite of valued point tracks as input, containing tracks of associated SNPs for the diseases one wish to compare with each other. A specific choice of similarity measure and underlying data representation will produce a heatmap of all pairwise similarities for the set of diseases, a dendrogram resulting from the clustering of distances, and a rank matrix with increasing order of distance for all pairwise comparisons.

The tools also have the option to output information of all available comparative measures. If this option is set, a heatmap, dendrogram and rank matrix is produced for all the measures available for the given representation, listed below one another. When the option of all measures is used, researchers can review how the different measures behave in terms of comparison for the given feature representation, and how the resulting output is affected.

The raw values that were used to generate the heatmap, dendrogram and rank matrix in a cluster analysis, are included in the output.

5.1.2 Tools for clustering of binary representations

The tools under the menu option Clustering of binary representations use binary feature representations and similarity measures to compare diseases and traits with each other. All the tools for binary representations are based on a pleiotropic model for relationships between pairs of tracks. Pleiotropy refers to the extent of which two diseases are affected by many of the same variants.

The tools for clustering of binary representations are defined for tracks of significantly associated SNPs, as found by GWAS. The different feature representations incorporate information of LD in various ways, with the goal of a biologically meaningful comparison of diseases and traits.

Each similarity measure will define a relationship between the tracks, which describe different properties of overlap between the features in the
CHAPTER 5. RESULTS

given data representation. Seven similarity measures are defined for all the binary representations, with two additional measures for the “Binary vector clustering” tool, as its feature representations are defined in terms of negative matches. Many measures are available, so that researchers can decide for themselves which properties they want to compare. If other measures are sought after at a later point, it is trivial to update the list of comparative measures for all tools.

5.1.3 Tools for clustering of continuous vectors

The tools under the menu option Clustering of continuous vectors use continuous vector representations and correlation coefficients to compare diseases and traits with each other. The two ways of representing tracks as continuous vectors differ greatly from one another, and are implemented in separate tools. The correlation coefficients Pearson and Spearman are used on all pairs of continuous vectors to assess whether or not the disease-associated data sets functionally related.

The first tool, “Correlation of track values”, forms two vectors for all pairwise comparisons of valued point tracks, where the features of each vector are the values in the respective tracks, for positions that are present in both tracks. It will further assess the relationships between the overlapping values of all pairs of tracks. Interpretation of results from this tool is heavily influenced on what the values of the VP track represent. For instance, if the values of the vectors are log-transformed p-values that indicate how likely it is that a given SNP is associated with a disease, the correlation will reflect an assessment of whether or not two diseases are related with respect to the significance levels of their SNPs. Diseases and traits represented by GWAS summary statistics are data sets that work well for this tool.

The second tool, “Correlation of overlap with other tracks”, uses a vector representation for the diseases and traits where the features are the number of overlapping genetic loci to other tracks. The relationships assessed in this tool is not whether or not two diseases are similar in terms of their reported SNP associations, but to what extent they are similar in their overlap with other diseases. The tool is made for data sets of significant SNPs.

5.1.4 Tools for empirical exploration

The tools that have been made for empirical exploration are meant to supplement the already existing HyperBrowser tool “Summary statistics per track in a GSuite”. It can be used to get information and plots for smallest distances for all points in a GSuite, either between or within tracks. It also allows for empirical exploration of LD tracks created from the “LD track generator” tool.
The purpose of the empirical tools is to investigate properties of point tracks within a GSuite. These properties can aid the decision of which tool for disease comparison is appropriate. The LD information can also be used to decide on which threshold to use in the tools that try to approximate LD.

The tool that print rsIDs of tracks in a GSuite can be used to print out all significant SNPs. This list of rsIDs can further be used in scripts outside the HyperBrowser, for instance for generating files of variants in LD with the Ensembl API.

5.1.5 Tools for data modification and creation

The tools that have been made for the purpose of data modification and creation provide utilities to generate data with LD information, and to uniformly represent GSuites with diseases for comparison.

The “GSuite liftover” tool maps all tracks of a GSuite to the chosen reference genome. For the clustering tools that use information of physical positions to compare track elements, it is important that the tracks are uniformly represented with respect to reference genome. The “Convert GSuite sumstat tracks” tool converts the primary tracks of a GSuite with sumstat files, into VP tracks, which might be needed when GWAS summary statistics are analyzed. The “LD Track generator” tool creates a linked point track of LD information for an entire GSuite, which can be used in the binary tools for haplotype and bipartite matching.

The last tool, “Expand GSuite with LD”, expands the point tracks of a GSuite with variants in LD with the original points. The expanded GSuite can be used in analyses with other tracks to find functional enrichment that overlap with the sequence-level data of the original GSuite.

5.2 Use case

We demonstrate parts of the methodology and software with a use case of seven diseases and traits, as represented by GWAS data. The articles reviewed in Section 2.3 had some overlap with respect to the diseases and traits they investigated, of which the following seven diseases and traits were chosen as subject for comparison:

- Crohn’s disease
- Ulcerative colitis
- Rheumatoid arthritis
- HDL
- LDL
- Triglycerides
CHAPTER 5. RESULTS

Figure 5.1: Two plots from the tools for empirical exploration, visualizing different properties of the tracks. To the left, count of SNPs per track is shown. To the right, the distribution of variants in LD for the GSuite is plotted as a function of distance, where $r^2 \geq 0.8$. The diseases and traits were abbreviated to align the figures: CD (Crohn’s disease), UC (Ulcerative colitis), RA (Rheumatoid arthritis), FG (Fasting glucose-related traits), HDL (HDL cholesterol), LDL (LDL cholesterol), TG (Triglycerides).

- Fasting glucose

Of these diseases and traits, Ellinghaus et al., (2016) study Crohn’s disease and ulcerative colitis, while Farh et al., (2015) and Bulik-Sullivan et al., (2015) have included all of them in their analyses.

We will start off with a description of how the data was collected and converted to GSuites. Next, some comparisons with different tools are conducted. Finally, output and results from the clustering is presented. Further details of data sets, HyperBrowser histories and workflow for this use case can be found in Appendix D.1.

5.2.1 GSuite creation

Three different collections of data, i.e. three different GSuites, were created with the seven diseases and traits. The first GSuite was created from the GWAS Catalog via the HyperBrowser, thus only with tracks of significant SNPs. As the GWAS Catalog contain SNPs from all populations, a crude manual filtering was conducted to remove SNPs of studies done for non-European ancestries. The other two GSuites were created from GWAS summary statistics, the first with p-values, the other with z-values.

5.2.2 Exploration of data properties

The different tools for exploration of track properties were run with the data set of significant SNPs from the GWAS Catalog, as were the HyperBrowser tool “Summary statistics per track in a GSuite”. Two of the resulting plots can be seen in Figure 5.1.
5.2. USE CASE

With the LD graph track created for the significant SNPs, one can for instance observe that ~1250 pairs of variants in LD, \( r^2 \geq 0.8 \) are parted by a distance of 200-300kb. Although these LD pairs only account for ~5% of all the pairs in LD given in the track, it is not a negligible amount. The interactive bar plots of smallest distances within or between tracks for all points in a GSuite give us an overview of the distribution of significant SNPs. For instance, it is indicated that approximately 260 SNPs lie within 500kb of another SNP on the same track.

5.2.3 Comparison of significant SNPs

The tools that use binary feature representations, as well as the tool with a vector representation of genetic overlap to other tracks were used to analyse the GSuite of significant SNPs. For the approximation of an LD block in the feature representations of bins and genetic loci, a threshold of 500kb was used. The threshold of 500kb was chosen partly because it is the locus size used by Farh et al., (2015), but also because the empirical exploration shows that this range will capture most signals from variants in LD.

In the tool executions, all available measures are chosen for the given representations. This leaves us with a range of different comparisons, all of which can be seen in the HyperBrowser histories linked to in Appendix D.1. The results of three different binary feature representations, all with different comparative measures, are shown in the Figures 5.2, 5.3 and 5.4.

The resulting cluster dendrograms for almost all representations and distance measures, show the same partition, although distances vary slightly. The exception is the comparison of continuous vectors, where the features are genetic overlap to other tracks. As mentioned in Section 5.1.3, this tool assesses the extent to which the different tracks overlap similarly to other diseases, and stand out from the comparisons of binary representations.

5.2.4 Comparison of GWAS summary statistics

The summary statistic GSuites are both compared with the tool “Correlation of track values”, and the resulting plots can be found in Figures 5.5 and 5.6. Their values represent different properties of GWAS SNPs, and the resulting comparisons thus reflect different relationships between the disease-associated data sets. The correlation of log-transformed p-values indicate whether or not two diseases have a similar distribution of significance for the SNPs present in both. The correlation of z-values, which in this scenario represent effect size and direction of the given alleles, indicate whether or not two diseases are affected by the same SNPs with effects of same or opposite direction.

64
CHAPTER 5. RESULTS

(a) Heatmap of pairwise similarity  
(b) Clustering dendrogram of distances

Figure 5.2: Output from cluster analysis where binary feature representations of haplotypes were used with the cosine similarity measure

(a) Heatmap of pairwise similarity  
(b) Clustering dendrogram of distances

Figure 5.3: Output from cluster analysis where binary feature representations of bipartite matching scores were used with the McConnaughey similarity measure

(a) Heatmap of pairwise similarity  
(b) Clustering dendrogram of distances

Figure 5.4: Output from cluster analysis where binary feature representations of bins of 500kb were used with the Forbes similarity measure
5.2. USE CASE

(a) Heatmap of pairwise correlation  
(b) Clustering dendrogram of distances

Figure 5.5: Output from cluster analysis where a continuous data representation of log-transformed p-values was used with the Pearson correlation measure.

(a) Heatmap of pairwise correlation  
(b) Clustering dendrogram of distances

Figure 5.6: Output from cluster analysis where a continuous data representation of z-values was used with the Pearson correlation measure.
Pearson and Spearman produce quite similar plots for both types of values, with Pearson as the chosen measure for the displayed figures. The heatmap and clustering of z-values deviate from the clustering of p-value correlation coefficients and binary similarity measures.
Chapter 6

Discussion

6.1 Data representation and resources

6.1.1 Data formats

We have chosen GSuites to represent collections of data subject for analysis in our tools. Comparisons are performed for all pairwise combinations of tracks within a GSuite. Diseases and traits, as given by GWAS data, are formatted as valued points tracks. The values for the tracks in our examples are for the most part log transformed p-values indicating degree of certainty that a SNP is significantly associated with the disease. Information of LD between variants are given as linked point tracks.

In bioinformatics, many different formats and standards are used for data representation, an example of which can be seen for all the different column headers of GWAS summary statistics published by different consortia. The range of different ways to format data pose a well known challenge of standardization in biomedical analyses, where for instance different choices for reference genomes and indexation of tracks to be compared might affect whether or not two tracks have similar positional overlap. The software of Bulik-Sullivan et al., (2015), munge_sumstats.py have been used to process GWAS summary statistics into a standardized sumstat file format. Some tools have also been proposed and developed through the course of this thesis that provide the users with options to create GSuites and LD tracks for given reference genomes and variants. These tools often require that the input data is SNPs, with certain specific column headers, requirements that can limit the usability of the tools on other data sets than the ones used in this thesis.

6.1.2 Software for LD computation

For two of the proposed tools for feature representation and comparison of GWAS data sets, actual information of LD between variants has been
incorporated. These HyperBrowser tools rely on precomputed linked point tracks that represent the LD structure within and between the associated SNPs of different diseases.

A multitude of tools exist that output information of LD in some form. Most of these, however, for instance the R library `ldblock`, require already processed LD data like that generated for HapMap phase 3. The HapMap phase 3 LD data set was released in 2009, and is seen as outdated compared to the genotype information found in the 1000 Genomes project. The problem of outdated data also applies to the software SNAP, which can be used to get LD variants from lists of rsIDs. The online interface is fast and easy to use, and the result is a file of the input SNPs expanded with LD variants, but the newest phased genotype data they use comes from the 1000 Genomes pilot 1, also referred to as “the low coverage pilot”, released in 2009.

LDlink, calculates LD between SNPs with data from 1000G phase 3, and has functionality close to what is needed. Unfortunately, it is only available through an online interface, and support for large batches of SNPs is poor. We decided that the two best candidates for finding variants in LD, were PLINK and the Ensembl Variation API. As the first option in its last update on May 15th, 2014, informed that “PLINK 1.9 is now available for beta-testing”, and had its last original release on October 10th 2009, the Ensembl API was chosen for LD computations, as the databases are well known and widely used, with support for developers and available documentation.

As it turned out, the Ensembl Perl API was quite difficult to set up and use. The functions for LD computations in themselves had many bugs which were reported and fixed through communication with the Ensembl development team. Moreover, the scripts had to be run on a local computer, and depended on a steady internet connection. In addition, the API calls raised a multitude of connection errors from time to time, supposedly because the script had run for a longer contiguous period of time, exceeding a couple of hours. These errors meant the runs had to be manually supervised, where SNPs with computed LD variants were removed from the folders the script was used on. Run time of the scripts were also long, where information of ten SNPs took between 5 to 50 minutes. The master file of variants in LD were computed over several weeks. Appendix E.3 go into further detail on bugs and deficiencies of the Ensembl API.

The PLINK software, although suspected of instability, as a result of its beta status, might have been a better and faster choice. The beta PLINK seems

---

2[https://www.broadinstitute.org/mpg/snap/ldsearch.php](https://www.broadinstitute.org/mpg/snap/ldsearch.php)
3[https://analysisistools.nci.nih.gov/LDlink/](https://analysisistools.nci.nih.gov/LDlink/)
4[https://pngu.mgh.harvard.edu/~purcell/plink/](https://pngu.mgh.harvard.edu/~purcell/plink/)
5[https://www.ensembl.org/info/docs/api/variation/variation_tutorial.html](https://www.ensembl.org/info/docs/api/variation/variation_tutorial.html)
6[https://www.cog-genomics.org/plink2](https://www.cog-genomics.org/plink2)
to be used to a great deal, and the latest release of June 7th 2016 is reported as stable on the home page.

At the time LD computation was performed, we did not find the functionality we sought after in the Ensembl REST API. However, later on, documentation for API calls with the exact same purpose and results as the Perl API were discovered. A Python script with basic functionality for LD variant computation have been tested and show great promise, further details of which can be found in Appendix C.2. The latest Ensembl release, version 85, was announced on July 19th 2016, supposedly containing improvements to both Perl and REST API for LD computation.

6.1.3 Potential data confounders

Data from the GWAS Catalog

For the use case, the GWAS Catalog was used to create a GSuite of significant SNPs for the given diseases. As the GWAS Catalog is manually curated, it cannot be expected to contain the latest data, a limitation that must be considered when data are collected for analysis. One example of this is the disease Primary Sclerosing Cholangitis (PSC), for which only two reported SNPs are found in the GWAS Catalog. In the studies of Liu, Hov, et al., (2013) and Ellinghaus et al., (2016), an additional 15 SNPs have been significantly associated with the disease.

Another potential problem is related to a slight discrepancy between the number of variants in the GWAS catalog integrated within the HyperBrowser and the numbers found in the GWAS catalog available online. The reasons for this has not yet been addressed by the HyperBrowser developers.

When the GWAS Catalog is used as source for GWAS data, the user must in most cases manually validate and filter the data to get a well-powered set of disease-associated SNPs. The SNP entries contain links to publications and a brief description of population, which in the use case in Section 5.2 was used for a crude filtering of SNPs found in populations of non-European descent. When data from the GWAS Catalog is extracted directly as a GSuite, filtering and data validation might become less accessible, as the separate tracks cannot be viewed directly without first downloading each track as a history element. If GWAS Catalog GSuites are extracted and used directly without filtering, the biological feature representations and resulting comparisons might be confounded by haplotype and SNP differences in populations.

21,321 reported SNPs in the GWAS Catalog as of July 24th 2016, and 18,660 SNPs in the HyperBrowser
6.2 METHODS RELATED TO CLUSTER ANALYSIS

Data from dbSNP

Data from dbSNP was used in the tool for genomic liftover. The latest builds were chosen, but the possibility of errors within the data are present. For instance, the dbSNP data used in the tool was at one point observed not to contain all SNPs. This was solved by another upload of SNP data for all chromosomes and reference genomes. Another observation has been that some of the BED files downloaded from dbSNP had discrepancies in the data format, violating the BED format requirement of not having the same start and end position. For our purpose, where a consistent mapping between rsID and positions for different reference genomes was the goal, errors in format does not necessarily pose a problem.

Data from the Ensembl Variation API

Data for LD between variants was computed with a script for the Ensembl API. The master LD file that was generated with the script for LD computations was subject to manual evaluation, and failing files of SNPs were re-run. However, as this was a manual process, the possibility that some error was introduced is present.

Bugs in the API itself could also be a source of erroneous or missing data. SNPs on alternating loci caused errors when used with the API, and were excluded from the master LD file. Another, perhaps more serious possibility of faulty data is some reported discrepancies with respect to the actual LD computation. In July, a user reported that the variation API found LD between variants that were not correlated. Furthermore, the same user discovered wrong values of $r^2$ between certain variants. These bugs have now been fixed, but whether or not they were present when the master file of LD information was generated, is unknown.

Another limitation of the LD computation is that it only contains information of LD between variants for the CEU population, i.e. the population of Utah Residents (CEPH) with Northern and Western Ancestry. It has been argued that the majority of studies in the GWAS Catalog tend to be from European-descent populations (Hindorff et al., 2009), which the CEU population cover to some extent. However, for GWAS conducted on other populations, the master file of LD may not represent the haplotype structure in an optimal manner.

6.2 Methods related to cluster analysis

This section discusses the methods implemented in the different tools. To illustrate some differences in how the tools work, in terms of the chosen feature representations and similarity measures, a GSuite of 40 diseases
and traits have been extracted from the GWAS Catalog and used in cluster analyses. See Appendix D.2 for sources and illustrations.

6.2.1 Feature representations

The binary feature representations presented in this thesis incorporate information of LD in different ways, an overview of which is given in Section 3.2.3. Of these, the representations of bins and genetic loci are closely related, and their similarity matrices show the same patterns of overlap, where features of bins produce the highest signals of similarity, and Gaussian loci the lowest. The approach with bins can be argued to be more naive than the block and Gaussian definitions of genetic loci, as it does not use information of the actual reported SNPs in its feature definitions. The two approximations for a genetic locus are similar, but the Gaussian version is more conservative in its estimations of similarity, as it weighs down the overlap between distant alleles. Figure D.3 in the appendix shows the similarity matrix for a feature representation of bins of 500kb. If the size of the bins or genetic loci is set to 1, the feature representations are equal to the single base definition. The approach with features of single bases does not include any information of biological properties, and is most appropriately used to indicate extent to which reported disease-associated SNPs overlap between the data sets.

For the approximations of LD, the haplotype block size can be defined with many different thresholds. In the CEU population, the average block size is reported to be below 20kb (The International HapMap Consortium, 2005). However, it is important to bear in mind that haplotype block sizes vary across the genome, and not necessarily well approximated with only one predefined threshold. Haplotypes up to 1Mb have been observed (Yu et al., 2005), a finding that is supported by the physical distances between the computed LD variants for all GWAS Catalog SNPs, visualized in Figure 6.1. By setting a block size of 1Mb, most SNPs in LD will be counted as overlapping, but the chance of introducing noise for the SNPs within LD-blocks that span a smaller region, is high.

The binary feature representations for haplotypes and bipartite matching, the similarity matrices of which are shown in Figures D.4 and D.5, are similar in that they both use actual information of LD. For the example analyses where these approaches have been used, the results from the pairwise comparisons are far more conservative than for the representations that approximate LD. Under the assumption that the LD data is without error and representative of the population under investigation in the GWAS data sets, this might indicate that the approaches for approximation of LD introduce a lot of false positives. It is important to note that the GSuite used to generate the example heatmaps of Appendix D.2 have not been controlled for appearances of non-European populations. Consequently, the less conservative models that approximate LD, in particular, might get higher pair-
6.2. METHODS RELATED TO CLUSTER ANALYSIS

Figure 6.1: Distribution of variants in LD for $r^2 \geq x$, where $x \in (0.8, 0.9, 1.0)$, with respect to the physical distance between them. The data used to generate the plot is the LD linked point track computed for all significant variants in the GWAS Catalog.

wise similarity between tracks because of noise from SNPs found in separate populations.

The feature representations of vectors with continuous values do not directly represent or approximate LD blocks, as in the binary representations. Rather, the values of the vectors describe properties of the two diseases that are to be compared. The vector representation of values associated with SNPs that are present in both diseases are appropriate for GWAS summary statistics, possibly with the ability to model relationships between diseases that have no associated SNPs of genome-wide significance.

The vector representation where the features are overlapping genetic loci between the represented disease and all other disease-associated data sets, model relations of genetic overlap to other diseases. This method for cluster analysis is more appropriate for data sets of significantly associated SNPs.

6.2.2 Similarity and correlation

For the binary feature representations, different similarity measures have been made available, and for the continuous feature vectors, two correlation coefficients are available. We do not conclude on what measures are best in this thesis, as they serve different purposes and can in their own way describe properties of the chosen representation.

When deciding on a comparative measure, however, some properties of the feature representations should be considered. In the binary representations, the feature space is extremely sparse, i.e. the features present in the different disease representations are few compared to the possible features of the full genome. With sparse feature representations, the chance that features overlap is also smaller. In general, small data sets can result in higher variance for the pairwise comparisons, and sample
sizes can influence which pairs of disease-associated data are the most similar. For interpretation of results, small samples of significant SNPs can be sensitive to noise, for instance when no filtering of the data set is performed.

Figures D.7, D.8 and D.9 show the similarity matrices of Jaccard, Simpson and McConnaughey, respectively, with the same feature representations of Gaussian genetic loci of 500kb. The GWAS data used in these figures have very variable SNP frequencies, ranging from 1 SNP in peanut allergy to 228 SNPs in type 2 diabetes. While Jaccard in general does not find any high levels of similarity between pairs of diseases, Simpson indicate full correlation between peanut allergy and a range of other diseases, which in effect is only caused by the one SNP in the peanut allergy track overlapping to any SNP in the other tracks. The heatmap showing the pairwise similarity with McConnaughey is similar to that of Simpson, but the meaning is different. The peanut allergy track which overlap with other diseases, will be given a similarity value close to 0, if the match found is greatly outweighed by number of mismatches.

Another property of the binary representations, is that they can be said to be asymmetric, where the positive matches are given more significance than the negative. For most of the representations, the negative matches are undefined. For the representations where negative matches are defined, $d$ greatly outnumber the other BTU summaries, an aspect that should be considered if the Forbes or Pearson measures are used.

For the correlation coefficients used on the continuous representations, different considerations are to be made, for instance the scale for which the values are given. If Pearson is used on p-values without any log-transformation, a pair of SNPs with values $(0.5, 0.5)$ will be found more similar than a pair of SNPs with $(0.5, 0.9)$, but the differences between these pairs of significance values are quite negligible. The difference between $(0.00001, 0.00001)$ and $(0.00001, 0.1)$ will be found to be small, while these in a p-value setting have great differences. For Pearson, differences in the meaning of the value, should be consistent with the numerical differences of the given scale. Spearman is a ranking coefficient, and does not have the same requirements for a linear scale.

As with the similarity measures, correlation coefficients used on smaller sample sizes can be subject to high variance, where noise and outliers might influence the resulting assessment to a greater degree than if large sample sizes are used. For vectors without normal distribution which might have outliers or extreme values, Spearman is the recommended correlation coefficient (Mukaka, 2012).

### 6.2.3 Distance conversion

For the feature representations, measures of similarity or correlation are used. The conversions of distance done for values between -1 and 1
could have been defined differently, for instance by taking the absolute value. For the purpose of comparing diseases with the data sets presented in this thesis, however, a negative correlation translates better into a large distance, and a positive correlation into a small distance. Different values that are used for correlation might describe entirely different relationships, and for some purposes, negative correlation might indicate similar diseases.

The Forbes measure, used as distance, does not have the same properties as the other similarity measures. For instance, a track compared with itself will probably not result in a value of 0, but for the cluster analysis, how tracks compare with themselves are without importance.

6.2.4 Clustering

For the clustering of the different binary representations, the feature representations that produce higher signals of similarity between diseases, result in more separated clusters. Figure D.1 in the appendix show a clustering dendrogram for the bin representation. However, as the binary comparisons for the most part produce values of high distance between pairs of diseases, there is little distinct separation of diseases in different clusters.

For the smaller use case presented in Section 5.2, the diseases and traits chosen are better characterized with respect to significant SNPs than many other diseases in the GWAS Catalog. The resulting clusters of the use case is notably similar across both feature representations and similarity metrics. The example of 40 diseases and traits given in Appendix D.2 does not output similarly well-defined clusters. Here, the biological signals are weak for most representations of significant SNPs, and the similarity measures can influence the final clustering to a greater extent, as small nuances produce different clusters.

6.3 Design principles

Three design principles have aided the course of development, and played a part in choices made for the online tool suite and publication of data sets and code. These principles are transparency, reproducibility and usability.

6.3.1 Reproducibility

A recently published survey in *Nature*, “1,500 scientists lift the lid on reproducibility”, reveals that more than 70% of researchers have attempted, but failed, to reproduce the experiments and results of another scientist. In addition, 50% of the researchers in the survey had failed to reproduce their own experiments (Baker, 2016).
Inability to reproduce the results of others have also been an issue during the work of this thesis. While at first glance, the disease-by-disease correlation matrix of (Farh et al., 2015) seemed straightforward to reproduce, the opposite was the case when the supplementary material was put under scrutiny. Even more so, it proved impossible to get in touch with the scientist that had performed the analysis and created the illustration, and his colleagues interpreted his methods differently than we expected, leading to a very different method for comparison of diseases.

When others are able to replicate experiments and test out methods and technologies, erroneous assumptions and flaws can be discovered, which might result in self-correction and adjustment into promising research directions. It can also be argued that reproducible methods and experiments leads to more trustworthy results. As TOP put it (Nosek et al., 2015):

> Reproducibility increases confidence in results and also allows scholars to learn more about what results do and do not mean

The software developed for this thesis have been implemented as a public tool suite in the Genomic HyperBrowser, meaning that the methods presented is available online for all to use. In the process where others are to test different data sets and use cases not thought of in the course of this thesis, methodological weaknesses might be discovered. Reproducibility in this sense might originate discussions that result in new insights and methods.

### 6.3.2 Transparency

An important part of reproducibility is transparency of methods and data, as it enables outsiders to review the steps taken in different experiments, and the exact computations performed for a given result.

The Transparency and Openness Promotion (TOP) Committee have created a range of standards to facilitate and promote openness in scientific research (Nosek et al., 2015). Two of the guidelines address transparency of data and code, of which the level 1 guidelines for data and code transparency are posted below:

- Article states whether data are available, and, if so, where to access them.
- Article states whether code is available, and, if so, where to access them.

With Galaxy histories and detailed explanation of tools and sources for data, the data sets used for the experiments in this thesis can be used by others in their own analyses. Additionally, descriptions of sources for

---

8https://osf.io/2cz65/
6.4. DIFFERENT METHODS FOR COMPARISON OF DISEASES

GWAS summary statistics and GSuite creation within the HyperBrowser is provided, so others can create their own data sets and test the methodology.

We disclose all the code developed for the tool suite on GitHub, a platform that is easy to access and navigate. The code repository can also be forked by others if they wish to improve and test the functionality further.

One setback with the available code is that the repository does not contain the rest of the HyperBrowser, meaning the statistics and web tools cannot be run locally. Also, the implementation make use of other HyperBrowser statistics, which are not included in the repository, meaning the released code is not transparent in all respects. The HyperBrowser team is however in the process of moving the entire code base to GitHub, which will alleviate this problem in the future.

6.3.3 Usability

Many tools and software for biomedical research can be quite difficult to use, either because they lack documentation, for instance with respect to information of data formats and input, or because they require local installation and have a lot of external software dependencies.

The tool suite produced in this thesis is meant to be accessible to others. Since it is available online, the tools require no installation to use. Different feature representations have been separated into different tools, so that the documentation for each tool can be limited to one way of representing the data, with different ways to compute similarity for the given representation. Detailed textual descriptions of purpose and underlying computation is provided below all tools.

The input boxes a user can set is implemented with error handling, so that the user is aided to select the correct parameters for the tool execution. In addition, text boxes with information accompany parameters that might need extra explanations. We have not conducted any extensive user tests to validate that the tool suite is in fact easy for others to use, however.

A drawback of the HyperBrowser web tools, in regard to usability, is that some introduction to the general flow of execution and data handling, where the Galaxy history is a central component, might be required before a user is confident in utilizing the tools for their desired purpose.

6.4 Different methods for comparison of diseases

6.4.1 Pleiotropy versus genetic correlation

Pleiotropy and genetic correlation are two different models that can be used to describe relationships between diseases. Pleiotropy indicate that a mapping exists from one genetic entity to many phenotypic entities (Paaby
and Rockman, 2013), in this context, from a SNP to different diseases and traits. If two diseases are affected by many of the same variants, we describe their relationship as pleiotropic, but how the shared loci influence the given traits are not known. Genetic correlation, on the other hand, try to describe genetic relationships between diseases, using information of alleles and effect sizes. With the genetic correlation model, two diseases can share many variants, but still be assessed as different. For two diseases to be correlated, they must have many shared variants whose direction of effect must be consistently aligned.

### 6.4.2 Related methods

In Section 2.3, four different articles were presented, which in their own way define methods for comparison of diseases.

The methods of ClusTrack (Rydbeck et al., 2015) are similar to the binary feature representations of single base positions and bins in this thesis. In an attempt to use their tool to cluster the same diseases and traits as in the use case presented in Section 5.2, it produced the same results for the option of direct sequence-level similarity, as the single base comparisons with Jaccard. However, the smallest bins available for features of proportional base pair coverage that worked, were that of the full chromosomes. These parameters did not produce the exact same clustering as the methods of the tool suite presented in this thesis, but managed to separate the blood lipids from the autoimmune diseases. The methodology of ClusTrack can be used to compare diseases with a pleiotropic model.

The method suggested by Farh et al., (2015), have been implemented in the tool suite with a continuous vector representation where the features are the genetic overlap to other diseases. Figure 2.8 created by Farh et al., (2015), and Figure D.6 created with the HyperBrowser tool have many similarities. The differences between the figures might be caused by differently curated data sets or the chosen correlation coefficient. Farh et al., (2015) have done some filtering of studies based on genome-wide hits and significance values, whereas no such filtering has been done for the data set of the HyperBrowser cluster analysis. Additionally, the tool implementation in the HyperBrowser have a smaller range for the overlap, only half of the genetic locus definition to avoid counting overlap more than once. The comparison of the continuous vector representation does not assess the extent of pleiotropy for pairs of diseases, but rather whether or not two diseases are related with respect to the genetic overlap to other diseases. The clustering in the article was used to reveal groups of phenotypes with related clinical features.

The analysis performed by Bulik-Sullivan et al., (2015) uses regression and GWAS summary statistics to model relations of genetic correlation between diseases and traits. Although no similar methods have been implemented in this thesis, the tool that uses a representation of vectors of continuous
values of overlapping SNPs have similar results when summary statistic data with z-values are used\(^9\). These z-values represent effect size and direction of SNPs, as computed with the same script Bulik-Sullivan et al., (2015) used. Notably, they yield similar values of correlation between Crohn’s disease and ulcerative colitis, HDL and triglycerides, and LDL and triglycerides. For the rest of pairwise comparisons, both methods show little correlation, but for low levels of reported correlation, they are often of similar direction. One pairwise comparison differ somewhat between these two methods, however. Bulik-Sullivan et al., (2015) report that HDL and fasting glucose are negatively correlated, a correlation also reported with the HyperBrowser tool, but with lower values. The regression based model generally report correlations on a scale between -1 and 1, while the HyperBrowser tool have the same pattern of correlation, but on a scale from -0.5 to 0.5.

The comparisons conducted by Ellinghaus et al., (2016) differ from the others, in that they target a specific set of diseases. They perform in-depth analyses of the relationships between the diseases, far more focused in their investigations than the other methods of clustering or correlation. It can be seen as an example of what further analysis can be performed after a superficial initial comparison between diseases and traits have been conducted.

There are many ways to compare diseases and traits, with different considerations for decisions of representation and comparative measures. Concluding on one correct way to compare diseases is not the aim of this thesis, as different methods are suitable for different purposes. The regression based software of Bulik-Sullivan et al., (2015) is available online\(^{10}\), with detailed documentation and user support. The method of Farh et al., (2015) have not been as accessible, as it has only been briefly described in the supplementary material, while ClusTrack seems error-prone and with little transparency regarding methods for data representation and comparison.

### 6.5 Developed software

#### 6.5.1 Source code

The tools for comparison of diseases and traits have for the most part been implemented with the use of statistics for representing the tracks with the different binary and continuous features. A focus have been to write functions of few lines, and define recurring computations and code as their own modular functions for reuse. The source code have been commented, with the purpose of explaining the tasks solved in the different files, so

\(^9\)Note that genetic correlation in the article of Bulik-Sullivan et al., (2015) is shown with blue for positive values, red for negative.

\(^{10}\)http://ldsc.broadinstitute.org
that developers can quickly familiarize themselves with the implemented code. The tool implementations follow the general prototyping templates for HyperBrowser web tools.

The feature representations are to a great extent separated in different tools. This has resulted in a more limited code base for the separate tools, and options does not rely on testing of predicates with logical operators. However, these tools share a lot of functionality, as the steps after a distance matrix have been computed are the same across all representations, where clustering are performed and plots outputted. To avoid a lot of recurring code, two helper classes, CommonClusteringFunctions and CommonCorrelationFunctions have been used for shared functionality in the binary and continuous representation tools, respectively. This structure might be surprising to a new developer, and not necessarily an optimal choice with regard to further development.

The tool code for data modification and creation are based on source code of other tools in the HyperBrowser which modify GSuites. These tools have solved issues of data formats that have been encountered throughout this thesis, but the implementations are not up to the general standard of the HyperBrowser code base, as very specific column headers must be present, and the code generally carry signs of it being the result of copy and paste.

As the code have been integrated in the HyperBrowser, other developers can reuse the statistics and methods for representation. Other helper classes might also contribute to other projects. The class of the JonkerVolgenant bipartite matching can be for instance be used in other applications where an optimal matching algorithm is needed. The class of Matplotlib can serve as examples for others for an alternative way of plotting graphs, to the R plotting functions more commonly used in the HyperBrowser code.

6.5.2 Code integration in the HyperBrowser

Development of tools within the framework of a mature code base such as the HyperBrowser can be a great advantage. The HyperBrowser is implemented in an object oriented manner, with tools and statistics as the main components that need to be altered or added when new functionality is wanted. It has established code patterns and syntax, for instance making it possible to prototype tools and get a working web interface very quickly. Developers that are familiar with the code and workflow can create an online tool for almost any purpose, and make it publicly available within hours. In addition, the code base is well developed, with modules and analyses for a wide range of mathematical and biological computations, meaning a lot of functionality can be reused, rather than developed from scratch.

The HyperBrowser provide functionality where genomic data sets can be created, inspected and formatted correctly without the need to download and process the data on a personal computer, and tracks must be subject
to a preliminary preprocessing before they can be used in statistics. In this preprocessing, the HyperBrowser checks whether or not the tracks are consistently defined, or contain errors with respect to data format or column specifications. The automated preprocessing tests can detect problems that are likely to cause problems in analyses, and which a manual validation process might have overlooked.

A simultaneous strength and weakness of the HyperBrowser implementation is that the more advanced functionality have been abstracted into core functions and methods that facilitate fast development with modularized code. When familiar with these code patterns, new and advanced functionality can be developed quickly and with few code lines. The true behavior of functions and code might not be obvious, however, when complex underlying computations are abstracted into a simple interface that works out of the box, especially if documentation is sparse. To the inexperienced HyperBrowser developer, mistaken use of the interface might lead to unexpected errors and a stagnant development process.

Many of the challenges met during development have been related to lack of documentation for code patterns and core HyperBrowser functionality. These challenges might have been avoided if the tool suite was implemented with custom functions and the HyperBrowser track parser, rather than with the use of statistics and more advanced functionality for internal track representations. We are not guaranteed, however, that an implementation with less use of advanced HyperBrowser functionality would have been less challenging. Most implementation choices, independent of framework, can in certain situations result in time-consuming debugging and unexpected errors.

For further development and use of the methods for data representation and comparison presented in this thesis, the approach with implemented statistics have resulted in code that can be reused, built on and improved by other HyperBrowser developers.

### 6.5.3 An alternative path for development

An alternative to the integrated tool suite of the HyperBrowser, could have been a stand-alone software package for comparison and clustering of GWAS data sets. It could have been made available through Docker\(^{11}\) to avoid problems related to installation of external dependencies. In terms of usability, compared to the HyperBrowser interface, many steps would have been taken in order to make the software accessible to others, such as choices regarding interface and execution. To account for issues of reproducibility, logs could be created for all runs of the tools with information of the parameters used. In contrast to the workflows and data sets that can be shared in the HyperBrowser, this approach relies on the users having control of which logs are connected with which results, and

\(^{11}\text{https://www.docker.com}\)
might be difficult to interpret for outsiders unfamiliar with the local file structure.

For a standalone application, processing and validation of the genomic data would have become a big part of the software package. Automated checks similar to the preprocessing step of tracks in the HyperBrowser would be beneficial, as well as further error handling of options and formats to help a user with tool parameters and execution. Additional scripts for formatting data sets downloaded from the GWAS Catalog would also be needed. The small tools for data modification created in the HyperBrowser for this thesis, would have been expanded to a much more extensive range of tools for data handling. For users, collections of data could be difficult to localize and use with the tool suite.

The HyperBrowser implementation provides an accessible and well explained tool suite that can be navigated and used for comparative analysis with only a minimum of technical skill required of the user. Its strengths are related to this usability, and integration with a framework that handles a multitude of other considerations related to genome-wide data analysis has enabled focus on methods for representation and comparison for the tool suite developed for this thesis. For a similar software package developed as a separate program, the focus would instead have been on infrastructure, execution and data handling.

6.6 Weaknesses in implementation

6.6.1 No automated tests

The implementation have throughout this thesis been tested manually by checking border cases and behavior on track data in test runs. This is a great weakness, as refactoring and new code could introduce errors that might not be detected afterwards. The lack of implemented tests have in the long run led to extra work and much time spent on manual testing of separate code parts.

Well written automated tests also serve as documentation for the code, and can help other developers to familiarize themselves with the implementation. The HyperBrowser provides integrated functionality and code patterns for testing. Although the test framework is not extensively documented as of today, a priority should have been to learn the test code patterns, as it would have greatly improved the tool suite with respect to trustworthy results.

6.6.2 Use of static data files

Two types of static data files are used in the tools for this thesis. The first are dbSNP information of SNPs, to facilitate mapping between rsIDs
and their positions. Two folders are created, one for the hg19 reference genome, the other for hg38. If none of the chromosome files of the chosen genome contains the SNP that are to be mapped in the tools that need this information, the tools will drop the SNP.

The other static data file is the master LD file of computed variants. This file is based on a list of all significant variants in GWAS Catalog, extracted via the HyperBrowser. Therefore, the master LD file does not have information of all expanded SNPs in LD, just those of high disease-associated significance. A weakness of the tools that rely on the LD data file, is that they assume that the SNPs to be expanded in the tools, are the same as those used to make the LD master file.

In general, use of static files pose a problem, as they can contain errors, and are not automatically updated to the latest version.

6.6.3 Time and space complexity

Memory usage and time complexity have not been the focus in this thesis. Although some measures have been taken to reduce run time of the tools, for instance creating triangular distance matrices and using the LAPJV algorithm for bipartite matching, time complexity has not been a main consideration during development.

For most analyses, there will be several nested loops, but the number of diseases subject to the pairwise comparisons are likely to be limited, as results for comparisons of more than 50 diseases can be difficult to visualize and interpret with the available cluster analysis plots. The tools for binary representations are made with the purpose of analyzing disease-associated data sets with SNPs of genome-wide significance, resulting in sample sizes of maximally a few hundred variants. As a result, parts of the implementation with bad time complexity will not necessarily lead to a time consuming tool execution.

The Galaxy history in the HyperBrowser allows a user to start a job that will run in the HyperBrowser, and does not need to think of it again before the analysis has completed. As a result, long runs does not demand resources or time from the user.

Since tool executions run on the HyperBrowser server, space complexity might be an issue. Jobs are slowed down when the load on the server is high, including analyses run by other users. The tools for comparisons of diseases are for the most part not demanding with respect to memory. The exception is the bipartite matching tool, which read in and store all information of the linked point track file provided. Similarly, the tools which use reference genomes read all SNPs of the chosen genome build into memory to map rsIDs to positions.
Chapter 7

Conclusion and future work

7.1 Conclusion

In this thesis, a comprehensive suite of tools for cluster analysis of GWAS data sets is presented. The focus has been on the role of feature representations and similarity measures, where the aim has been to find methods for a biologically meaningful cluster analysis of sets of disease-associated SNPs.

The methods provided can be used to investigate pleiotropic relationships between GWAS data sets of significant SNPs, or to discover relationships of functional dependence, such as genetic correlation, between diseases represented by GWAS summary statistics. To our knowledge, there does not exist any other as easily accessible and thoroughly explained suite of tools for different ways to compare sets of disease-associated data. With the options available in the tools for cluster analysis, a user can compare GWAS data sets with a chosen feature representation and measure, or get results for all available measures listed below each other. The latter option is mostly useful for those who wish to visualize the effect of different comparative measures, but can also provide further insight into the properties of the clustered data sets.

The tool suite work well with GWAS data of significant SNPs, as well as summary statistics, depending on the tool used. The software have been made part of the Genomic HyperBrowser, of which the online user interface and Galaxy elements facilitate reproducibility, transparency and usability in terms of utilizing the tools for research.

7.2 Future work

The web tools developed for this thesis have been tested with small collections of disease-associated data. The results have been promising, for instance in the use case where similar relations between diseases were
7.2. FUTURE WORK

found, as in the article by Bulik-Sullivan et al., (2015). A thorough analysis of results has however not been conducted. To validate that the tools can be used to find clusters that reflect true biological relationships, further experiments with updated and properly filtered data sets can be conducted, and the output analyzed and compared to the results of others.

Another aspect of validation of methods for feature representation and comparison, relates to the data used for experiments with the tools. We have used biological data of disease-associated SNPs, and the true relationships between these data sets are unknown. Synthetic data sets could be constructed that simulate properties of disease-associated data, where relations between tracks are known. The tools could then be run on these synthetic sets, in order to objectively assess whether or not the cluster analysis detects the constructed relationships between the data sets.

The master file of LD used in this thesis might be erroneous, and for future use of the LD tools, a new computation should be conducted. The Ensembl REST API shows great promise in this respect, and might be included in HyperBrowser tools directly for LD computation, replacing the local scripts that make use of the Perl API. Throughout the work with this thesis, we have been approached by other scientists that have sought information of LD, given a set of SNPs. An accessible tool for LD computation might be a great asset for the rest of the research group. A tool developed for this purpose is the “LD GSuite generator”, which expands a GSuite of point tracks to include all variants in LD with the original tag SNPs. However, steps should be taken to update the tool to not use a precomputed LD file.

A main focus of this thesis has been to incorporate information of linkage disequilibrium for reported risk variants, so that the data representations would account for the entire haplotypes that constitute risk factors for a given disease. The different methods for representing GWAS data presented in this thesis, can be extended to integrate information of epigenomic tracks or pathways associated with genomic loci. Clustering on this extended level might provide further insight into functional relationships between diseases and traits.
Appendices
Appendix A

Linkage criteria

The following section describes some further technical details concerning the definitions of linkage criteria for hierarchical agglomerative clustering. The information can be used to decide on which criteria is the more appropriate in a given clustering setting.

Given two clusters \( A \) and \( B \), the equations A.1, A.2 and A.3 show the mathematical formula of single, complete and average linkage, respectively.

\[
d_{\text{single}}(A, B) = \min \{ \text{dist}(a, b) : a \in A, b \in B \} \quad (A.1)
\]

\[
d_{\text{complete}}(A, B) = \max \{ \text{dist}(a, b) : a \in A, b \in B \} \quad (A.2)
\]

\[
d_{\text{average}} = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} \text{dist}(a, b) \quad (A.3)
\]

The single linkage criterion only looks at the closest two members of each cluster and choose to merge the ones with the smallest distance. Single linkage work locally, and can produce long straggling clusters, as can be seen to the right in figure 2.6. This chaining of objects can violate the property of high intra-cluster similarity. In the middle of the same figure, a complete linkage clustering is shown, which demonstrate the opposite extreme. Complete linkage focus on minimizing the distance between the members furthest away from each other in each cluster, and creates compact clusters with the smallest diameter possible. The downside of the complete linkage criterion is that it is sensitive to outliers, which means it can violate the property of high inter-cluster dissimilarity (Hastie, Tibshirani, and Friedman, 2009). Both single and complete linkage only use one member of each cluster when deciding on clusters to merge, in effect disregarding the overall structure of the clusters in question. With

\( A \) and \( B \) can also be single objects, i.e. singleton clusters
The average linkage criterion is given by A.3, an equation that translates to, in each clustering step, updating the distance from a joined cluster \( A \cup B \) and all other clusters \( X \) by proportionally averaging distances between \( A \) and \( X \), and \( B \) and \( X \), see equation A.4. This algorithm is also called UPGMA, Unweighted Pair Group Method with Arithmetic Mean. A weighted variant, WPGMA, uses the arithmetic mean rather than the cluster sizes to update distances between a new cluster \( A \cup B \) and \( X \), shown in equation A.5.

\[
\text{dist}_{(A \cup B), X} = \frac{|A| \cdot \text{dist}(a, x) + |B| \cdot \text{dist}(b, x)}{|A| + |B|} \quad (A.4)
\]

\[
\text{dist}_{(A \cup B), X} = \frac{\text{dist}(a, x) + \text{dist}(b, x)}{2} \quad (A.5)
\]

The group average linkage criterion uses all objects to evaluate cluster similarity, and represents a compromise between the more extreme single and complete linkage criteria. An argument for average linkage and against complete and single linkage, is that it seems to have a statistical consistency property the other two lack. Assume a data set with \( p \) attributes, \( N \) elements, and each cluster \( k \) a random sample from some population joint density. The data set is a random sample of \( K \) such cluster densities. For two clusters \( G \) and \( H \), when \( N \to \infty \), the single linkage measure approaches zero and complete linkage approaches infinity, in both cases independent of the joint cluster densities of \( G \) and \( H \). Only the group average will reach the average distance measure of the two clusters, given their densities. This statistical inconsistency makes it unclear what single and complete linkage model in terms of population distribution (Hastie, Tibshirani, and Friedman, 2009).

The centroid linkage criterion use all original data points of a cluster to create a centroid, whereupon the closest centroids are merged. Due to its dependency in the original data, it is not easily applied to other data representations than real-valued vectors, for instance a distance matrix. Centroid linkage have an addition problem of inversions, where similarity increase during clustering. Consequently, it is not monotonic, and as a result often not used in research, as the dendrograms can be harder to interpret. Single and complete linkage does not need to know how the distances were computed, and can work directly on distance matrixes, as can UPGMA and WPGMA, the two average linkage criteria (Christopher D. Manning, 2008).
Appendix B

Data sources and formats

This chapter contains links to and descriptions of sources for data sets used in the thesis. Additionally, different tables of properties related to the data sets are presented.

B.1 NCBI: dbSNP

The National Center for Biotechnology Information (NCBI) provide access to biomedical and genomic information, including the dbSNP database, which contain SNP-related data. To map rsIDs with SNPs to different reference genomes, two sets of BED files were downloaded from the SNP resources of NCBI1. The data used are the BED files of SNPs found for the genome builds given in Table B.1.

<table>
<thead>
<tr>
<th>Reference genome</th>
<th>Genome build</th>
</tr>
</thead>
<tbody>
<tr>
<td>hg19</td>
<td>human_9606_b147_GRCH37p13</td>
</tr>
<tr>
<td>hg38</td>
<td>human_9606_b147_GRCh38p2</td>
</tr>
</tbody>
</table>

Table B.1: Genome build for the data from dbSNP used to map rsIDs to positions

B.2 NHGRI-EBI GWAS Catalog

The GWAS Catalog has been used to create disease-associated data sets for use in this thesis. Tables B.2 and B.3 describe the properties related to the SNPs stored in the GWAS Catalog, as given in the GWAS Catalog documentation2. The first table show the information stored for the different studies, and the second table show the information stored for each SNP in a given study. Additionally, Table B.4 shows an example of

---

2https://www.ebi.ac.uk/gwas/docs/methods
how a SNP from the GWAS catalog is represented as a track element. In the example, the table properties are the same as the columns in the valued point tracks generated for GWAS Catalog data sets via the HyperBrowser.

### B.3 LD tracks

The Perl script for LD computation with the Ensembl API, presented in Appendix C.2, was used to create a master file of LD information with the following format:

| chrnum | pos_ldSNP | ldSNP | pos_tagSNP | tagSNP | r2 |

In addition, a master linked point track for the master LD file has been created, with no thresholding of $r^2$. This track can be downloaded directly and further uploaded to the HyperBrowser. Table B.5 shows the locations where these data sets can be downloaded. We provide data of the original SNPs used for the computation, the master file of LD information, as computed with the Ensembl API, and the corresponding linked point track.
### Table B.3: Attributes related to a SNP in the GWAS Catalog

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORTED GENE</td>
<td>Gene(s) reported by the author; “Intergenic” is used to denote a reported intergenic location (or lack of gene if it appeared that gene information was sought); ?NR? is used to denote that no gene location information was reported.</td>
</tr>
<tr>
<td>STRONGEST SNP-RISK ALLELE</td>
<td>SNP(s) most strongly associated with trait + risk/effect allele (? for unknown risk allele). May also refer to a haplotype.</td>
</tr>
<tr>
<td>SNPS</td>
<td>Strongest SNP; If a haplotype is reported above, this field may include more than one rs number (multiple SNPs comprising the haplotype). Multiple SNPs may also be included if proxy SNPs are reported.</td>
</tr>
<tr>
<td>RISK ALLELE FREQUENCY</td>
<td>Reported risk/effect allele frequency associated with strongest SNP in controls (if not available among all controls, among the control group with the largest sample size). If the associated locus is a haplotype the haplotype frequency will be extracted.</td>
</tr>
<tr>
<td>P-VALUE</td>
<td>Reported p-value for strongest SNP risk allele. Note that p-values are rounded to 1 significant digit (for example, a published p-value of 4.8 x 10^-7 is rounded to 5 x 10^-7).</td>
</tr>
<tr>
<td>PVALUE_MLOG</td>
<td>-log(p-value)</td>
</tr>
<tr>
<td>P-VALUE (TEXT)</td>
<td>Information describing context of p-value (e.g. females, smokers).</td>
</tr>
<tr>
<td>OR or BETA</td>
<td>Reported odds ratio or beta-coefficient associated with strongest SNP risk allele. Note that if an OR &lt;1 is reported this is inverted, along with the reported allele, so that all ORs included in the Catalog are &gt;1. Appropriate unit and increase/decrease are included for beta coefficients.</td>
</tr>
<tr>
<td>95% CI (TEXT)</td>
<td>Reported 95% confidence interval associated with strongest SNP risk allele, along with unit in the case of beta-coefficients. If 95% CIs are not published, we estimate these using the standard error, where available.</td>
</tr>
</tbody>
</table>
### B.3. LD TRACKS

<table>
<thead>
<tr>
<th>Property</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>date_added_to_catalog</td>
<td>25-nov-08</td>
</tr>
<tr>
<td>pubmedid</td>
<td>17554300</td>
</tr>
<tr>
<td>first_author</td>
<td>WTCCC</td>
</tr>
<tr>
<td>date</td>
<td>07-juni-07</td>
</tr>
<tr>
<td>journal</td>
<td>Nature</td>
</tr>
<tr>
<td>link</td>
<td><a href="http://europepmc.org/abstract/MED/17554300">http://europepmc.org/abstract/MED/17554300</a></td>
</tr>
<tr>
<td>study</td>
<td>Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls.</td>
</tr>
<tr>
<td>initial_sample_description</td>
<td>1,748 European ancestry cases, 2,938 European ancestry controls</td>
</tr>
<tr>
<td>replication_sample_description</td>
<td>(see Parkes 2007)</td>
</tr>
<tr>
<td>region</td>
<td>5q33.1</td>
</tr>
<tr>
<td>seqid</td>
<td>5</td>
</tr>
<tr>
<td>start</td>
<td>150860514</td>
</tr>
<tr>
<td>reported_gene(s)</td>
<td>IRGM</td>
</tr>
<tr>
<td>mapped_gene</td>
<td>IRGM - ZNF300</td>
</tr>
<tr>
<td>upstream_gene_id</td>
<td>345611</td>
</tr>
<tr>
<td>downstream_gene_id</td>
<td>91975</td>
</tr>
<tr>
<td>snp_gene_ids</td>
<td>.</td>
</tr>
<tr>
<td>upstream_gene_distance</td>
<td>11845</td>
</tr>
<tr>
<td>downstream_gene_distance</td>
<td>33869</td>
</tr>
<tr>
<td>strongest.snp.risk.allele</td>
<td>rs1000113-T</td>
</tr>
<tr>
<td>snps</td>
<td>rs1000113</td>
</tr>
<tr>
<td>merged</td>
<td>0</td>
</tr>
<tr>
<td>snp_id_current</td>
<td>1000113</td>
</tr>
<tr>
<td>context</td>
<td>intergenic</td>
</tr>
<tr>
<td>intergenic</td>
<td>1</td>
</tr>
<tr>
<td>risk.allele.frequency</td>
<td>0.07</td>
</tr>
<tr>
<td>p.value</td>
<td>3.00E-07</td>
</tr>
<tr>
<td>pvalue_mlog</td>
<td>6.522878745280337</td>
</tr>
<tr>
<td>p_value_(text)</td>
<td>.</td>
</tr>
<tr>
<td>or_or_beta</td>
<td>1.54</td>
</tr>
<tr>
<td>95%<em>ci</em>(text)</td>
<td>[1.31-1.82]</td>
</tr>
<tr>
<td>platform_[snps_passing_qc]</td>
<td>Affymetrix[469,557]</td>
</tr>
<tr>
<td>cnv</td>
<td>N</td>
</tr>
</tbody>
</table>

Table B.4: Example of a SNP from the GWAS Catalog, represented as a row in a valued point track
### APPENDIX B. DATA SOURCES AND FORMATS

<table>
<thead>
<tr>
<th>Data set</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original significant SNPs retrieved via HyperBrowser</td>
<td><a href="http://folk.uio.no/johannhh/all_significant.txt">http://folk.uio.no/johannhh/all_significant.txt</a></td>
</tr>
<tr>
<td>SNPs expanded with LD variants, as computed with the Ensembl Variation API</td>
<td><a href="http://folk.uio.no/johannhh/all_significant_expanded.txt">http://folk.uio.no/johannhh/all_significant_expanded.txt</a></td>
</tr>
<tr>
<td>Master linked point track, ( r^2 &gt; 0 )</td>
<td><a href="http://folk.uio.no/johannhh/linked_point_track_0.gtrack">http://folk.uio.no/johannhh/linked_point_track_0.gtrack</a></td>
</tr>
</tbody>
</table>

Table B.5: Overview of files used for and resulting from the LD computations. The first is all significant variants fetched from the GWAS Catalog through the HyperBrowser. The second entry is the master file of variants in LD, as computed with the Ensembl Variation API from the list of significant SNPs. The third entry is the master LD file represented as a linked point track.

Note that use of the linked point track might be restricted by the track size and HyperBrowser processing load. Some thresholding of the original LD data is recommended to reduce data size, and remove noise of variants in low LD.
Appendix C

Source code

C.1 GitHub repository

All the code developed for this thesis have been made available at GitHub:

- https://johhorn.github.io/gwas-clustering/

Note that the source code of this repository only contains code developed specifically for this thesis. The code that have been implemented as part the HyperBrowser installation will not be possible to run directly, as it depends on the rest of the HyperBrowser framework.

C.2 LD scripts

As described in section 2.5.3, LD between variants can be represented as a graph with the linked point track format. The GitHub repository linked to in Section C.1 contains a folder named “LD scripts”, which contains the programs used for the computation of LD.

Of these scripts, the majority of Python code is used for automatization of LD computation, where the Perl scripts were called on folders with several smaller files of SNPs. This was done because the Ensembl Variation API raised errors at regular intervals, and to reduce the amount of SNPs that had to be re-run with the LD computation. The Python script createLinkedPointTrack.py can be used to create linked point tracks with a predefined $r^2$ threshold from any text files with the following format:

<table>
<thead>
<tr>
<th>chrnum</th>
<th>pos_ldSNP</th>
<th>ldSNP</th>
<th>pos_tagSNP</th>
<th>tagSNP</th>
<th>r2</th>
</tr>
</thead>
</table>

The Perl scripts hg19expandSNPs.pl and hg38expandSNPs.pl can be used to compute LD variants from a list of rsIDs. The first maps the variants to the hg19 reference genome, the latter maps the variants to hg38. Appendix E.3.2 contains some suggestions of improvement for faster computation of LD.
C.2. LD SCRIPTS

The Python script `ensemblRestTest.py` contains test code which makes use of the Ensembl REST API.
Appendix D

Analyses and experimental data sets

D.1 Use case of seven diseases and traits

To demonstrate how the different tools work, a use case for cluster analysis of seven diseases and traits was conducted, as described in section 5.2. The full galaxy histories of all steps conducted for the different parts of the analysis, is provided in Table D.1. These histories can be viewed and imported, where the results can be further inspected, and new analyses conducted with the data.

D.1.1 Data from the GWAS Catalog

The GSuite of tracks with significant SNPs was created using GWAS Catalog data. The full extraction and GSuite creation was done via the HyperBrowser, as described in section 2.6.3. Further, a crude manual filtering was done on each track, to remove SNPs resulting from studies where the subjects were of non-European descent. SNPs from trans-ethnic studies were not subject to any filtering, as long as cases and controls of European descent were part of the study. Table D.2 show information of the SNP count of the different tracks before and after filtering.

D.1.2 Summary statistics

The summary statistics were downloaded locally to a computer from the resources given in Table D.3. These files were further used with the munge_sumstats.py script described in 2.6.4 to generate filtered sumstat files. However, the files of Crohn’s disease and ulcerative colitis contained some extreme numeric values in the column of signed statistics, which overflowed Python’s numerical limits on the local computer that ran the
D.2. Example data for 40 diseases and traits

GWAS data for 40 diseases and traits, many of which were used in the clustering method of (Farh et al., 2015) was created with the purpose of explorative analysis and testing of the different feature representations. The galaxy history for these analyses can be found at the following location:
APPENDIX D. ANALYSES AND EXPERIMENTAL DATA SETS

<table>
<thead>
<tr>
<th>Disease/trait</th>
<th>Original SNP count</th>
<th>Filtered SNP count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crohn’s disease</td>
<td>194</td>
<td>162</td>
</tr>
<tr>
<td>Ulcerative colitis</td>
<td>127</td>
<td>114</td>
</tr>
<tr>
<td>Rheumatoid arthritis</td>
<td>183</td>
<td>151</td>
</tr>
<tr>
<td>Fasting glucose-related traits</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>HDL cholesterol</td>
<td>152</td>
<td>115</td>
</tr>
<tr>
<td>LDL cholesterol</td>
<td>117</td>
<td>95</td>
</tr>
<tr>
<td>Triglycerides</td>
<td>103</td>
<td>85</td>
</tr>
</tbody>
</table>

Table D.2: Original and filtered SNP counts for use case data

- [hyperbrowser.uio.no/clustering/u/johannhh/h/several-diseases-and-traits](https://hyperbrowser.uio.no/clustering/u/johannhh/h/several-diseases-and-traits)

The GSuite of disease-associated tracks were created in the HyperBrowser as explained in Section 2.6.3 The track of Primary Sclerosing Cholangitis was increased with 15 additional SNPs, as reported by Ellinghaus et al., (2016) and Liu, Hov, et al., (2013). No filtering or further modification has been done for the rest of the tracks, meaning differences in populations for the studies where SNPs are reported, might introduce noise. The biological implications of these comparisons should therefore not be given too much importance. The purpose of the examples are first and foremost to illustrate how feature representations and similarity measures might influence the resulting similarity. These examples are only given for significant SNPs.

The heatmaps in the Figures D.3, D.4 and D.5 show the effect of different binary feature representations, all with the cosine similarity measure. The feature representations are bins of 500kb, haplotypes and bipartite matching scores, respectively. The dendrogram in figure D.1 show the resulting clustering from the feature representation of bins of 500kb.

The heatmap of Figure D.6 and resulting dendrogram in D.2 show the cluster analysis where the data is represented with features of genetic overlap to all other diseases, as described in Section 3.3.

The heatmaps in the Figures D.7, D.8 and D.9 show the effect of different similarity measures, more specifically Jaccard, Simpson and McConnaughey, for the same feature representation of Gaussian genetic loci of 500kb.
<table>
<thead>
<tr>
<th>Consortium</th>
<th>Disease/trait</th>
<th>Link</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadinstitute</td>
<td>Rheumatoid arthritis</td>
<td><a href="https://data.broadinstitute.org/alkesgroup/SUMMARYSTATS/PUBLIC/RA/RA_GWASmeta_European_v2.txt">https://data.broadinstitute.org/alkesgroup/SUMMARYSTATS/PUBLIC/RA/RA_GWASmeta_European_v2.txt</a></td>
<td>(Okada et al., 2014)</td>
</tr>
<tr>
<td>Broadinstitute</td>
<td>HDL</td>
<td><a href="http://www.broadinstitute.org/mpg/pubs/lipids2010/">http://www.broadinstitute.org/mpg/pubs/lipids2010/</a></td>
<td>(Teslovich et al., 2010)</td>
</tr>
<tr>
<td>Broadinstitute</td>
<td>LDL</td>
<td><a href="http://www.broadinstitute.org/mpg/pubs/lipids2010/">http://www.broadinstitute.org/mpg/pubs/lipids2010/</a></td>
<td>(Teslovich et al., 2010)</td>
</tr>
<tr>
<td>Broadinstitute</td>
<td>Triglycerides</td>
<td><a href="http://www.broadinstitute.org/mpg/pubs/lipids2010/">http://www.broadinstitute.org/mpg/pubs/lipids2010/</a></td>
<td>(Teslovich et al., 2010)</td>
</tr>
<tr>
<td>Magic</td>
<td>Fasting glucose</td>
<td><a href="http://www.magicinvestigators.org/downloads/">http://www.magicinvestigators.org/downloads/</a></td>
<td>(Dupuis et al., 2010)</td>
</tr>
</tbody>
</table>

Table D.3: Overview of sources for GWAS summary statistics, used in the use case of seven diseases and traits.
APPENDIX D. ANALYSES AND EXPERIMENTAL DATA SETS

Figure D.1: Typical dendrogram for the binary feature representations and clusterings. In this case, bins of 500kb and the Cosine similarity measure was used.

Figure D.2: Dendrogram for comparison of overlap between tracks, Pearson correlation coefficient.
D.2. EXAMPLE DATA FOR 40 DISEASES AND TRAITS

Figure D.3: Cosine similarity measure, binary feature representations of 500kb bins.

Figure D.4: Cosine similarity measure, binary feature representations of haplotypes.
Figure D.5: Cosine similarity measure, binary feature representations of bipartite matching scores

Figure D.6: Comparison of overlap between tracks, Pearson correlation coefficient.
Figure D.7: Binary feature representation of gaussian genetic locus of 500kb, Jaccard similarity measure.

Figure D.8: Binary feature representation of gaussian genetic locus of 500kb, Simpson similarity measure.
Figure D.9: Binary feature representation of gaussian genetic locus of 500kb, McConnaughey similarity measure.
Appendix E

Using the Ensembl Variation API

This chapter describes how to use the Ensembl Variation API. It is meant as a starter help for other developers, and contains information about installation, scripts and challenges met, with the intent of aiding others in making an informed choice when deciding on software for LD computation.

There are different steps for installation, depending on which API is needed and the computer it is downloaded to. This project has used the variation API, which deals with various forms and functions of variants, including SNPs and linkage disequilibrium. Note that the variation API, as of May 2016, to some degree has outdated or no documentation, with many of its functions marked as “at risk”. Through the thesis project, a lot of bugs was discovered and fixed. Such instability should be taken into account before deciding on whether or not the API is suitable for the given needs.

The REST API provided by Ensembl can be easier to set up and use, but have fewer available options than the Perl API. It does not provide LD variant calling, for instance. The Ensembl dev team is available through email and fix reported bugs quickly, in addition to detailed guidance and debug assistance, which can make up for the issues one might encounter during installation and usage.

The Ensembl web page contains most of the information needed to install and run scripts with the API. The information needed is not gathered on a single page, however, but rather spread out in different guides and blog posts. Therefore, a summary of what was needed to use the scripts for LD variants on a Mac OS X 10.11 (El Capitan) is provided below. This computer have been used throughout the thesis project.

At the beginning of this thesis, the author had very little experience with Perl and bash environments. Parts of the setup that were not explicitly detailed in the guides, perhaps because they are obvious to the experienced user, were difficult to recognize and debug. To install and run scripts successfully on the author’s computer, support from the Ensembl dev team
was needed. To aid others in a similar situation, the first section details the steps needed to install the Variation API and its dependencies, in the order they were carried out. The information was gathered from the links below and through e-mail correspondence with the dev team at dev@ensembl.org. Note that with new releases of the API and different operating systems, the installation steps might differ.

The section after the installation setup briefly describes the scripts used, and how to set up a local genome assembly, rather using the default remote FTP connection. The last part describes challenges encountered when using the Variation API, as well as technical limitations of the scripts used. Hopefully, it can be of help for others experiencing the same behavior. If the walk-through is too superficial, these links are recommended for in-depth explanations:

<table>
<thead>
<tr>
<th>Content</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensembl API installation</td>
<td><a href="http://www.ensembl.org/info/docs/api/api_installation.html">http://www.ensembl.org/info/docs/api/api_installation.html</a></td>
</tr>
<tr>
<td>1000 Genomes LD data</td>
<td><a href="http://www.ensembl.info/blog/2015/06/18/1000-genomes-phase-3-frequencies-genotypes-and-ld-data/">http://www.ensembl.info/blog/2015/06/18/1000-genomes-phase-3-frequencies-genotypes-and-ld-data/</a></td>
</tr>
<tr>
<td>Variation API tutorial</td>
<td><a href="http://www.ensembl.org/info/docs/api/variation/variation_tutorial.html#ld">http://www.ensembl.org/info/docs/api/variation/variation_tutorial.html#ld</a></td>
</tr>
<tr>
<td>Variation API README</td>
<td><a href="https://raw.githubusercontent.com/Ensembl/ensembl-variation/release/84/C_code/README.txt">https://raw.githubusercontent.com/Ensembl/ensembl-variation/release/84/C_code/README.txt</a></td>
</tr>
</tbody>
</table>

Table E.1: Overview of documentation needed for setup and computations of LD using the Ensembl API

E.1 Setup and installation on OS X

In-depth information on how to check whether or not these are installed already, where to find these programs, how to install etc., can be found in the OSX prerequisites blog post

E.1.1 Prerequisites

The following programs must or should be installed, if not already on your computer:

- Xcode, available from App Store

---

APPENDIX E. USING THE ENSEMBL VARIATION API

- GCC
- Perlbrew
- Perl 5.14.4
- cpanminus
- A local MySQL server installation (optional, but recommended)

E.1.2 Core dependencies

Install the following with cpanm (cpanminus)

```
$cpanm DBI
$cpanm IPC::Run
$cpanm JSON::XS
```

E.1.3 DBD::mysql

As the ensembl blog points out, the dynamic library is not available on the default search paths of OSX. They recommend to symbolically link the library into /usr/lib, an option which requires administration rights:

```
$sudo ln -s /usr/local/mysql/lib/libmysqlclient.X.dylib
  /usr/lib/libmysqlclient.X.dylib
```

Replace the X with the version on your computer. The Ensembl blog has libmysqlclient.18.dylib, while this thesis uses libmysqlclient.20.dylib. Note that OS X 10.11 have stricter root access than what the blog post accounts for. Hopefully, the mySQL library has been updated to handle OS X root restrictions, so that such absolute references are not needed. If not, however, some tweaking might be needed, such as setting an absolute reference via the install_name_tool.

When the library is symlinked, and the MySQL server is running, DBD::mysql can be installed:

```
$cpanm DBD::mysql
```

E.1.4 Cloning the Ensembl repos

For each database in the Ensembl project, different APIs are provided. When installing the source code, one can choose to download the APIs individually, as not all are needed to communicate with a specific database. The variation API needs at least the following two repos:

https://github.com/Ensembl/ensembl-variation.git
https://github.com/Ensembl/ensembl-io.git
As the Ensembl APIs are quite complex, however, and it can be difficult for a non-experienced user to know which functions are needed before running scripts, downloading all APIs might be the easiest. The following instructions assume that an installation directory, `src` is created as below. If you wish to use a different source folder, environment paths must be set accordingly later on.

```
$ mkdir ~/src
```

### BioPerl

I would recommend others to clone the BioPerl code from GitHub. In the installation used in this thesis, however, BioPerl was fetched with wget:

```
$ cd ~/src
$ wget http://bioperl.org/DIST/BioPerl-1.6.1.tar.gz
$ tar xzvf BioPerl-1.6.1.tar.gz
```

### Ensembl API repositories

When GitHub is used to clone the Ensembl source code, updating existing repos to newly released versions is easy. The following instructions install Ensembl git tools, all Ensembl APIs, and show how all APIs simultaneously can be updated to a new branch.

```
$ cd ~/src

# Install Ensembl git tools for easy Ensembl repo management
$ git clone https://github.com/Ensembl/ensembl-git-tools.git
$ export PATH=$PWD/ensembl-git-tools/bin:$PATH

# Install all APIs:
$ git ensembl --clone api

# Update all APIs to a branch:
$ git ensembl --checkout --branch release/X api
```

With the last git command, one can easily checkout all branches to the latest release. Change the X to the desired release branch, i.e. API version. The latest version, as of May 2016, is 84. Version 85 is scheduled for release in July 2016.

### Tabix

To access the correct genotype phase data sets, tabix need to be installed, as explained in the 1000 Genomes LD data blog post.\(^2\)

\(^2\)http://www.ensembl.info/blog/2015/06/18/1000-genomes-phase-3-frequencies-genotypes-and-ld-data/
E.1.5 Compilation

In order to use the Variation API, go to the ensembl-variation/C_code folder and compile it with make calc_genotypes. After pulling changes in the variation source code, the C code should be recompiled. In these cases, it is safest to first delete the calc_genotypes and calc_genotypes.o files before running the make command.

E.1.6 Setting the environment

The following should be set in your ~/.bash_profile. Otherwise, it must be run each time a new terminal window is opened to run the variation scripts. Note that the paths are relative to the folder structure I got from downloading the modules, but might vary if a different approach was taken than described above. BioPerl have a different folder structure when downloaded with git, for instance.

E.1.7 Check installation status

Finally, you can check whether or not the Ensembl API is installed correctly by running the following:

```bash
$ perl ~/src/ensembl/misc-scripts/ping_ensembl.pl
```
If the scripts are not working, a starting point for debugging is to check whether or not the environment is set in the terminal, and if the MySQL client is running.

E.2  Scripts and local setup

The scripts used for LD computation are made available on GitHub. Appendix C.2 briefly describe the scripts created and used for the computations in the sections below.

E.2.1  API calls without FTP connection

The default setup described in the guides and above sets the api calls to fetch data from the Ensembl Variation database through FTP. However, as described in the next section, this setup might have some issues, especially with regard to the FTP connection. An alternative is therefore to download the files locally and set the script up to fetch these in stead. For the following setup, the folders ~/src/ensembl-vcf/GRCh37 and ~/src/ensembl-vcf/GRCh38 were used as location for the VCF files of the corresponding reference genome, downloaded from the locations in table E.2.

<table>
<thead>
<tr>
<th>Genome</th>
<th>Location</th>
</tr>
</thead>
</table>

Table E.2: Location of 1000G VCF files for genotype data

When the VCF files are downloaded into the chosen location, open the file ~/src/ensembl-variation/modules/Bio/EnsEMBL/Variation/DBSQL/vcf_config.json. For the two human genome assemblies, change the type to "local" and filename_template to the path where the files were downloaded. As perl do not interpret the path starting with ~/..., an environment variable or the full path must be used, as in the following example:

```json
{
    "id": "1000genomes_phase3",
    "species": "homo_sapiens",
    "assembly": "GRCh37",
    "type": "local",
    "strict_name_match": 1,
    "filename_template":
    "~/Users/Johanne/src/ensembl-vcf/GRCh37/ALL.chr###CHR###.phase3_somatic.rsID.genotypes.vcf.gz",
    "chromosomes": [
```
For different folder structures of where the VCF files are put, the important thing is to know the path to the data, and add them correctly to the JSON declaration. Also note that the current genome release is at version 82. With new releases, the ftp links for the download will change. The local setup will not download the binary files in the working directory, as the remote setup does.

### E.3 Challenges with the Ensembl APIs

This section is based on the authors experience with the Ensemble Variation API. A more experienced perl developer and user of Ensembl APIs might not have the same challenges. The discussion below is based on usage of a simple Ensembl script, as well as the resources available and needs of the bioinformatics group at UiO.

#### E.3.1 Running the scripts

Many issues appear during run time of the perl script expanding SNPs with their LD variants. A stable Internet connection during LD computation of each SNP was required in the default set up as described in the Ensembl guides. This stability might not be guaranteed neither at home nor at school. If the Internet connection timed out, or the computer went into a resting state, the process stopped and had to be rerun.

When calling LD variants for a small set of SNPs, less than 2500, these issues were manageable. To make it easier to run the perl script without detailed monitoring and modification of a master file containing all SNPs
that should be expanded, some additional python scripts were made. First, a python script dividing the file of rsIDs that should be expanded into smaller files with generic names, where each file contained 11 SNPs, except the last one that could have fewer. These files was manually put in separate folders, for instance with between 10 or 15 SNP files in each folder. A second script then ran the variation API on each such folder containing SNP files, accumulating out files in the same folder of corresponding lists of expanded SNPs with the following tab-separated column format:

| chrnum | pos_ldSNP | ldSNP | pos_tagSNP | tagSNP | r2 |

Unfortunately, this called for a certain amount of human supervision when running the scripts, to make sure files that timed out was rerun. Files successfully run could be moved out of their folder, leaving the failed files behind for a new round of variant calling. With human supervision, however, comes a greater chance of human error. Files could be deleted, misplaced or thought to be successful when they actually failed.

When the process of extracting all significant SNPs from NHGRI-EBI and expand them with variants began, however, the amount of supervision of the running scripts became very time consuming. The various error messages gives few clues as to what might be the issue, except that the connection with the API seemed to be the prime suspect. They suggested to download them locally in stead, and alter the configuration to use the local downloads in stead of going through the FTP connection, the setup of which is given in Section E.2.1. With this setup, the only remaining error encountered, is one related to the MySQL server connection. According to the Ensembl dev team, it can appear when a script has run for a long time:

```
DBD::mysql::st execute failed: Lost connection to MySQL server during query at /Users/Johanne/src/ensembl/modules//Bio/EnsEMBL/DBSQL/Ba
```

If a fully automated pipeline is made and put remotely, with local VCF files for genotype data to alleviate the internet connection problems, one could have a good way to extract data from the Ensembl Variation API. Some error handling for ensuring the database connection is up might be needed to avoid This would undoubtedly be a great asset to many research groups, as this functionality has been requested both in a range of forums as well as by researchers in the bioinformatics group at UiO during the thesis project. Unfortunately, as many of the functions are still at risk, and the frequent releases might introduce new error, the Ensembl APIs might require some expertise and resources for maintenance of the setup and scripts. Local VCF files should be updated from time to time to include
the newest genotype data. New releases should not be pulled immediately, as is advised on the Ensembl web page, to make sure post-release bug fixes have been incorporated.

E.3.2 Implementation and documentation

The Ensembl project started in 1999, and has among other things provided researchers with the Perl APIs used in this thesis, a comprehensive toolbox for accessing their genomic databases, as listed below:

- Core databases
- Compara database
- Variation database
- Regulation Database

The functionality available is complex and seems to have everything one need to extract data sets for biological analysis. The Variation API alone, which was used in the thesis project, contains of almost 200 classes, each with a wide range of methods related to biological concepts the class represent, such as population and LD. Documentation and whether or not a certain method is stable or not can be found in the Variation API documentation. The key database adaptors for used in the LD variation script were:

```
Bio::EnsEMBL::Variation::DBSQL::VariationAdaptor
Bio::EnsEMBL::Variation::DBSQL::LDFeatureContainerAdaptor
Bio::EnsEMBL::Variation::DBSQL::PopulationAdaptor
```

For a user unexperienced with the Variation API, the vast amount of methods to choose from presents a challenge. First of all, the way that some of the classes work together, is not necessarily intuitive from the documentation provided. Second of all, as the Variation API turned out to have many methods with "at risk" status, some did not do what the documentation claimed, returned nothing or crashed the script and produced an error message. For instance, to generate LD variants for all populations containing LD of interest, rather than pre-specifying CEU, it seemed as though on could use the VariationFeature method `get_all_LD_Populations()`. But due to a switch of how genotypes was used, this function crashed entirely in v83.

Technical limitations were made clear through error messages or unexpected results, rather than being explained in the documentation. For instance, using the super population EUR was not possible as it contained more than 503 samples, which exceeded the number of genotypes supported by the Ensembl LD computation script. In correspondence with the

---

3 http://www.ensembl.org/info/docs/Doxygen/variation-api/
Ensembl dev team, they considered to increase the number of allowed populations in the LD computation script in release 85. An alternative is to use `fetch_all_LD_Populations()` from the population adaptor, which return all available populations, or loop over the EUR subpopulations. However, we decided to use only CEU, as each run with a different population will add substantially to the time used.

Another technical limitation came through the use of the GRCh38, the default reference genome for the Ensembl database connector. Several human chromosomal regions exhibit sufficient variability to prevent adequate representation by a single sequence, and GRCh38 takes this into account by using alternating stand-alone sequences for selected variant regions. All variants mapping to such alternate loci caused an error in the LD script. The Ensembl dev team asked us to exclude these variants and just use the mappings to the reference sequence in stead. Therefore, LD calling is only done for the variation features that adhere to the requirement `$vf->slice->is_reference`. The newest release of the API, version 85, has corrected this error.

For the LD computations, the API has implemented an upper limit of 500kb for LD computation. Variants further away than this will not be computed. The script used for this thesis is quite slow, as it for all LD computations fetches both SNPs in the LD pair to print their information. With the genome liftover tool implemented in the HyperBrowser, this information can be found afterwards, rather than being part of the Perl script. The LD computation can then be made faster by setting the flag of not fetching the full variant objects with `get_all_ld_values(1)` as in the code below. In the following lines, rsIDs are still computed, as is $r^2$:

```perl
my $ldfc = $ldfc_adaptor->fetch_by_VariationFeature($vf, $ld_population);
my $ld_values = $ldfc->get_all_ld_values(1);
foreach my $ld_hash (@$ld_values) {
    my $r2 = $ld_hash->{r2};
    my $variation_name1 = $ld_hash->{variation_name1};
    my $variation_name2 = $ld_hash->{variation_name2};
}
```
Bibliography


Cowper-Sal·lari, R. et al. (2012). Breast cancer risk-associated SNPs modulate the affinity of chromatin for FOXA1 and alter gene expression. In: Nature Genetics 44.11, pp. 1191–1198. DOI: 10.1038/ng.2416.


