Universiteit
Leiden
The Netherlands

Descriptive analysis and inference of Higher-order Linkage Disequilibrium

Karasmani, E.

Citation
Karasmani, E. (2016). *Descriptive analysis and inference of Higher-order Linkage Disequilibrium*.

Version:	Not Applicable (or Unknown)
License:	License to inclusion and publication of a Bachelor or Master thesis in the Leiden University Student Repository
Downloaded from:	https://hdl.handle.net/1887/3597157

Note: To cite this publication please use the final published version (if applicable).

Mathematical Institute

Master Thesis

Statistical Science for the Life and Behavioural Sciences

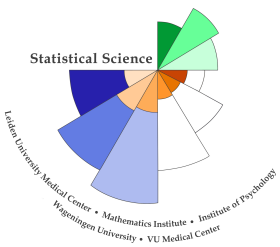
Descriptive analysis and inference of Higher-order Linkage Disequilibrium

Author:
Eleni Karasmani

First Supervisor:
Dr. Stefan Boehringer
Dept of Medical Statistics and
Bioinformatics, LUMC

Second Supervisor:
Prof.dr. A.W. van der Vaart
Mathematical Institute, Leiden University

January 2016



Abstract

Genetic studies in human populations have unveiled a vast number of polymorphisms. This variability is shaped by evolutionary forces such as drift, selection, bottlenecks and more. The approach applied in this thesis is based on haplotype method meaning that we consider block of SNPs of the human genome. This method has an advantage over single SNP analysis. The aim of the thesis is to characterise the genetic history based on the correlation structure of polymorphisms. This correlation structure is characterised in terms of joint cumulants which generalise linkage disequilibrium (LD) to more than two loci. The advantage of a multilocus LD (high order LD) measurement over a pairwise LD measurement is that the latter is not adequate to detect simultaneous allele associations among multiple markers. Characteristic patterns of this standardised higher-order LD are summarised in a catalogue for all possible genetic histories in an exponentially growing population. Heatmap approach was applied to visualise and interpret the behaviour of the LD patterns under different population scenarios. Tests were developed to determine whether given data is more likely to stem from a particular catalogue entry than others. These tests were developed using Monte Carlo techniques. Simulations were complemented by an analysis of part of the public HapMap project.

Acknowledgements

I would like to express my sincere and deepest gratitude to my supervisor Dr. Stefan Boehringer for his encouragement, positive attitude and advice during the progression of this thesis. His suggestions, feedback and discussion were instrumental for the completion of this thesis. Furthermore, I would like to acknowledge Professor Aad van der Vaart for his really helpful and critical comments. I am also grateful to the Professors from the Statistical Science for the Life and Behavioral Sciences master track, for the great programme and knowledge that they have offered to us. Moreover, I would like thank all the people of the of department of Medical Statistics of LUMC for their great support, nice meetings and work discussions. Finally, I am really grateful to my family for their support and encouragement during this endeavour.

Contents

Contents iii

1 Introduction 1
- 1.1 Aim of this thesis . . . 2

2 Biological concepts 4
- 2.1 DNA . . . 4
- 2.2 Alleles and haplotype blocks . . . 5
- 2.3 Homologous recombination . . . 7
- 2.4 Neutral selection . . . 8

3 Haplotype patterns resulting from genetic events 10
- 3.1 Catalogue . . . 11
- 3.2 Example for three loci . . . 12
- 3.3 Algorithm to construct the catalogue . . . 14
- 3.4 Structure of catalogue . . . 16
- 3.5 Complete catalogue for two loci . . . 18

4 Linkage disequilibrium 23
- 4.1 Pairwise linkage disequilibrium . . . 24
- 4.2 Standardised LD . . . 26
- 4.3 High order of Linkage disequilibrium . . . 26

5 Data application of visualisation and genetic interpretation of Higher-order Linkage Disequilibrium. 28
- 5.1 Compute LD and standardized LD for the catalogue . . . 29
- 5.2 Compute LD and standardized LD for real data (HapMap project) . . . 30
- 5.3 Create a Euclidean distance matrix . . . 30

5.4 Visualize the patterns using heatmap plots . . . 31
5.5 Interpreting heatmap plots . . . 31

6 Inference on higher order LD **37**
6.1 Introduction to Monte Carlo methods . . . 37
6.2 Parameter testing of parametric bootstrap procedure . . . 37
6.3 Data generation under the Null hypothesis . . . 39
6.4 Algorithm of parametric bootstrap . . . 39
6.5 Hypothesis testing using pairwise differences of distance measurements . 41
6.6 Performance evaluation . . . 41
6.6.1 Simulations settings . . . 41
6.6.2 Type I error . . . 44
6.6.3 Power . . . 47

7 Data Example **52**
7.1 Multiple testing correction . . . 54

8 Discussion **58**

Appendix A **61**
.1 Re-parametrization . . . 61
.2 Standardised LD for arbitrary number of loci . . . 62

Appendix B **64**

References **92**

Chapter 1

Introduction

The joint distribution of closely linked (*i.e.* roughly co-transmitted from generation to generation) genetic markers contains important information that can be used in genetic association studies (Gorelick and Laubichler [2004],Lewontin [1988],McPeek and Strahs [1999]) and population genetics. Deviation from probabilistic independence of alleles at two different markers (allelic association) has been characterised by various measures. Allelic association is also known as linkage disequilibrium (LD). One application of estimating LD in gene mapping (association studies) is to avoid redundancy caused by strong correlations between markers and allows to optimize association studies. Multilocus LD (higher order LD) (Balliu et al.) extends the concept of pairwise LD as the latter is not adequate enough to detect simultaneous allele associations among multiple markers. Genetic recombination, mutations and changes in allele frequencies (genetic drift) play an essential role in shaping the patterns of LD in a population. Therefore, the dependency structure of markers as characterized by multilocus LD contains information about ancestry.

Humans and their respective genome are quite diverse. Genetic studies such as the human genome project have unveiled a vast number of polymorphisms (variable locations within the genome). This staggering complexity is one of the reasons that every person is unique (Alberts [2007]). This variability is shaped by evolutionary forces such as drift, selection, bottlenecks (sharp reduction in the population size) and more. Genetic ancestry as characterized in this thesis is based on the assumption of neutral evolution (Kimura [1968]) (Chapter 2.4). This is a simplification of reality but can serve as a good approximation of real data.

An important question in Biology and its related fields is to identify the relationship between alleles or SNPs and their respective effect on the expression of genes result-

ing in a specific observed and sometimes unobserved phenotype. The way that those SNPs affect their target genes is mostly dependent on the location of the SNP by their relation with genes or regulating elements (e.g. enhancer, silencer, promoter; Alberts [2007]). Gene mapping is affected by genetic ancestry in at least two ways. First, it is well known that both phenotypes and genotypes distributions differ strongly between ethnicities, with the latter to be an important confounder in association studies (Spielman et al. [2007], Consortium [2003]). Second, strong correlations between genetic markers can make it impossible to distinguish their individual contribution. In this case the evolutionary history of the SNPs can contribute additional information. It is therefore important to unveil the genealogy of a given set of joint genotypes at markers of interest.

Single nucleotide polymorphims (SNPs) are genetic markers for which exactly two outcomes (alleles) are possible. Variability of SNPs has a large contribution to total genetic variability and plays an important role in genetic association studies. SNPs are often used for association but their genetic history is not taken into account. The importance of SNPs is that they can measured very cost-effectively and the most additional more complex genetic variation, such as deletions, insertions, or inversions can be associated with them, so that they can serve as a universal markers. Based on comprehensive studies, the human population contains about 10 million SNPs. The HapMap project (Consortium [2003]) is one of the efforts characterising the joint distribution of SNPs, i.e. the haplotype distribution. It provides the information about the location of the SNPs in the genome, their frequencies in several populations, and haplotype frequencies. This database is open assess and is meant to be used by scientists to improve association studies as well as theoretical investigations. It is used in this thesis to apply the developed statistical methods.

1.1 Aim of this thesis

The aim of the thesis is to provide a genetic interpretation of Higher-order Linkage Disequilibrium (LD). For example, recombination between loci results in a reduction of the dependence between the alleles, hence a reduction of LD. We chose a setup where we assume an exponential growing population, which approximately allows us to ignore fixation event, i.e. the loss of alleles due to the random walk exhibited by allele frequencies over time. This leaves mutations and recombinations as major evolutionary forces that are considered here. We show that after proper standardisation, standardised higher-order LD can summarise population history through cumulant patterns of

SNPs. These patterns are summarised in a catalogue that can be used for comparison with actual data. Tests are developed to determine whether such given data is more likely to stem from a particular catalogue entry. Simulations are complemented by an analysis of part of the public HapMap project.

Chapter 2

Biological concepts

2.1 DNA

The human genome consists of chromosomes, which contains all the inherited information. Joe Hin Tjio in 1955, determined that humans contain 46 chromosomes (22 autosomal and 2 sex chromosomes, XX for females and XY for males) with two copies of each chromosome (one inherited from the mother and one from the father). In 2004, the human genome was almost fully sequenced and revealed that it consists of approximately 3.2 x 109 base pairs (bp) (Alberts [2007]).

Each chromosome is composed of long strands of the Deoxyribonucleic Acid (DNA). The DNA was first discovered by Friedrich Miescher in 1869 and its 3D structure was proposed in 1953 by James Watson and Francis Crick after the pioneering work by Rosalind Franklin and Maurice Wilkins (Watson and Crick [1953]). The DNA contains all the necessary information to build up an organism. That information is encoded in the DNA with format of four different nucleotides, adenine (A), guanine (G), thymine (T), and cytocine (C). The DNA is double stranded, hence if on a specific position on one strand we have A, then on the same position on the other strand will be T. The same is true for G and C.

Genes are regions of the DNA, which contain the necessary information to encode the proteins (via transcription a process which produces different types of RNA, such as mRNA, tRNA and rRNA) are necessary for all the different functions of an organism (Alberts [2007]). When a cell needs a particular protein, the gene that encodes that protein will transcribe to a single strand RNA (from the double stranded DNA), which subsequently will translate to a functional protein. Genes, contain regions, the exons, which are translated to proteins, and the introns, which are spliced out during the

maturation of RNA. A genomic locus is a region of DNA which contains one or more genes.

Interestingly, humans contain approximately 400 different cell types (Alberts [2007]) with every one containing the same DNA. However, it is the transcriptome of each cell that provides the cell's unique identity. Hence, different combination of proteins and/or different levels of expression of specific genes will result to different cell types.

It is apparent that the diversity of different cell types in each individual person as well as the diversity between people in a population could be due to different reasons. SNPs can be one of those aforementioned reasons. In normal conditions, a specific genomic region, which contains a gene and somewhere close to it its regulatory elements such as an enhancer, will behave normally. However, as a result of evolutionary processes not all the people will have the exact DNA composition at this region.

We can discriminate the below two different cases. Some will have different nucleotides in a region (introns or intergenic genomic regions) with no, as yet, apparent functional role and no apparent phenotype. The other case is when those SNPs are located in exons and can lead to two different outputs; either a SNP causing a silent phenotype or causing a new phenotype. The first is when the change in the nucleotide composition leads to the exact same phenotype like in the normal condition, i.e. the same amino acid, hence the same protein without any obvious severe effect. The majority of the cases, lies on those aforementioned categories, with no observed phenotype, disease or trait. However, in some cases, we observe variation in phenotype, when a SNP in an exon or another functionally relevant position, leads to encoding a different amino acid, hence either different protein or no functional protein at all.

More recently, the importance of SNPs in the genes' regulatory elements has been described. These include enhancers, silencers and promoters which are important for the proper transcriptional control of their target genes. SNPs in these regulatory elements can lead to abnormal transcriptional control with either increased or decreased expression of the target gene when compared to the baseline levels. Likewise, as compared to coding SNPs, these deviations leads to variation in phenotypes.

2.2 Alleles and haplotype blocks

A particular location in a chromosome is called genetic locus. That genetic locus can exhibit more than one sequence variants and is often called a marker (Alberts [2007]). Polymorphisms is a marker with at least two of it's alleles having frequency above 1% in the population. The different variants of a locus are called alleles. Humans are diploid

organisms; they contain two copies of the same chromosome, hence two copies of the same gene. In a diploid organism, each gene will typically have two alleles occupying the same position (locus) on the homologous chromosomes. A haplotype is a combination of specific alleles, which is inherited to an organism from a single parent.

Humans, on an evolutionary scale, are quite young. Our ancestors were living in Africa, about 100,000 years ago. Since we are separated by them only by a few thousand generations, large pieces of the chromosomes without any alteration, were inherited from parents to their offspring. It is also known, that sets of alleles are passed from parent to child as one group. These ancestral chromosomal segments are called haplotype blocks, which have been passed from generation to generation with little genetic variation. Often these blocks are regions of high order LD (see Chapter 4). Haplotype blocks may harbour haplotypes that have a low variability in the human population, each one representing an allele combination passed down from a shared ancestor long time ago, specific for a particular population. Hence by studying the haplotype blocks, we can decipher the genotype of our ancestors; our genetic evolutionary history about how our genome is shaped through different generations as a result of the different evolutionary forces.

The completion of the sequence of the human genome at 2002, provided the knowledge to the scientific community of the nucleotide composition of the genome. From 2002 until now, the human genome has been updated with new sequences with more accurate information about the nucleotide sequence. The improvements in the next generation sequencing technology have been instrumental towards that update (Romanoski et al. [2015]). It became apparent that the genetic variance of the population should be unveiled. The result of the international HapMap Project, a multinational effort started at 2002, is a haplotype map of the human genome (HapMap), a database which describes the common patterns of human sequence variation (Consortium [2003]). It provides the information about the location of the SNPs in the genome and their frequency in the population. This database is open free access and is meant to be used by scientists to improve study design, the analysis of studies and provide deeper insight into the genome.

Scientists hope that the haplotype maps will provide better insight into the identification of disease-causing and disease-susceptibility genes. Instead of looking for all the million SNPs of the genome and to find out which are ones causing a disease, scientists have to pinpoint the haplotype block that appears to be inherited by individuals with the disease. By identifying the haplotypes within blocks that are inherited by individuals with the disease, scientists can narrow down significantly the mutations linked

or causing that disease. Subsequently, by pinpointing the specific haplotype block, scientists can unveil the specific gene associated with the disease.

Interestingly, haplotype blocks can also provide an insight about the ancestral history of a specific allele and whether it has been favoured by natural selection. That can lead to the identification of specific blocks which were inherited from to generation to generation and maintained in the population. If an allele does not offer a selective advantage on the individual, it will be more rare in a population. However, if an allele provides an evolutionary advantage to an individual, it will be more common in the population hence older in the evolutionary history. In this case the haplotype blocks surrounding it will be smaller because it will have had many chances of being separated from its neighbouring variations by the recombination events.

New alleles, which for example provide a resistance to a disease, can appear in the population and will spread quick since those individuals will survive and pass the mutation on to their offspring. Let's imagine a population of 100 cockroaches; 99 have the same allele whereas 1 has a different allele for a specific chromosomal locus offering great resistance to pesticide. In normal conditions, all the 100 cockroaches are alive. However, when putting them under selection, for example with a pesticide, the 99 will die and the 1 will survive as a result of the selective advantage conferred by that allele. This cockroach will pass to its offspring the resistance allele which through the subsequent generations will become prominent

2.3 Homologous recombination

Homologous recombination (also termed general recombination) is the genetic exchange that takes place between a pair of homologous DNA sequences (Figure 2.1) (Alberts [2007]). The importance of homologous recombination is apparent in two distinct biological processes, DNA damage repair and creating new combinations of DNA sequences in each chromosome.

Radiation, chemicals as well as faults in DNA replication can result in double-strand DNA breaks. Unless the organism corrects those DNA breaks, they can have catastrophic downstream consequences. Homologous recombination can repair double-stranded breaks accurately, without any loss or alteration of nucleotides at the site of repair. Briefly, the strand from the "normal" sister chromatid invades to the broken sister chromatid. Subsequently, the first is used a template, to synthesize and fill up the missing nucleotides.

A law of genetics is that each parent makes an equal genetic contribution to the

offspring. In meiosis, double-strand breaks are intentionally produced along each chromosome and homologous recombination is initiated by exchanging DNA segments either *in cis* (on the same chromosome) or *in trans* (between chromosomes). The latter often has catastrophic results leading to lethality as a result of diseases such as cancer like acute myeloid leukemia, whereas the first often leads to creating genetic diversity. Homologous recombination, preferentially takes place between the maternal and paternal chromosomes. It is apparent that when homologous recombination takes place, a specific sequence of SNPs from either the paternal or the maternal chromosomes will be mixed resulting in new combinations in the offspring. The evolutionary benefit of that procedure is that it creates new combinations of genes, new alleles, which can perhaps be beneficial for the organisms. The recombination during meiosis, results in greater diversity of the genetic pool of a population, which in terms of evolutionary biology is a hallmark for evolution and development of species. In this thesis, we consider homologous recombination only *in cis*.

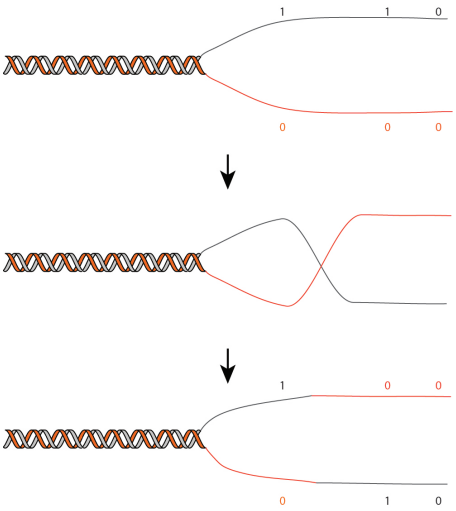


Figure 2.1: The plot illustrates recombination event between to chromatids

2.4 Neutral selection

The theory of neutral selection has become central to study the evolution (Duret [2008]). It was Motoo Kimura who at 1968 proposed that theory (Kimura [1968]).Its principle is that at the molecular level the evolutionary changes are not caused by natural selection but by random drift of alleles that are neutral and can be explained by stochastic pro-

cesses. New alleles are introduced by mutations with the alleles frequencies to change from generation to generation. Mutations with a selective advantage (an advantageous allele) will have a higher probability of fixation by natural selection; the opposite is true for deleterious mutations. Growth patterns are important to be considered. Growth results in less fixation. However, exponential growth, ignores fixation as approximation. Humans grow exponentially.

Neutral theory suggests that a lot of genetic variation is the result of mutation and genetic drift, although selection has been proven to take place. Its main advantage is that it can lead to conclusions which can be tested against actual data. Neutral mutations are the ones which do not affect an organism. The theory also claims genetic drift control those mutations. The genetic drift is controlling the fate of the neutral mutations.

Chapter 3

Haplotype patterns resulting from genetic events

Observed haplotypes either represent the ancestral DNA sequence or have been derived from it through a series of genetic events. There is a large number of possible such events mentioned earlier and we focus here on the two events mutation and recombination. Moreover, we limit ourselves to bi-allelic markers such as SNPs for describing variation in DNA sequence. Both simplifications are discussed later.
As a preparation to the later goals of the thesis we ask here: if a certain sequence of mutations and recombinations occurs which will be the resulting haplotype distribution. As the actual haplotype frequencies vary across generations we interest ourselves with the existence of haplotypes and ignore the specific frequency. We call this a hpalotype pattern in the following. In order to solve this problem, we make the following assumptions.

1. Mutations occur only once, *i.e.* a specific allele at a given locus is created by a single mutation event. This corresponds to the infinite sites model used in neutral models.

2. Once a new haplotype is created through a mutation or recombination, it is not lost again. Drift is a possible genetic force leading to loss of alleles/haplotypes. The current model is valid for situations when drift does not have a large effect (such as in large, exponentially growing populations for which humans are a good example).

3. Every possible sequence of mutations and recombinations can happen.

Given these assumptions, possible haplotype patterns are explored.

3.1 Catalogue

In order to relate a haplotype pattern with corresponding genetic events, possible sequences of genetic events have to be explored until a given haplotype pattern is found. As the correspondence is not one-by-one, all possible event sequences have to be explored for a given pattern. This suggests to explore event sequences once and store all haplotype patterns that are generated. This tabulation is called the catalogue in the following.

In the following, we restrict ourselves to bi-allelic loci such as SNPs. We arbitrary assign 0 or 1 to distinct nucleotides at each locus. A single haplotype becomes an N-tuple of binary digits if we consider N loci. If we denote with B the set of possible haplotypes, then $|B| = 2^N - 1$, i.e. $B = \{B_n\}$,$B_n = (a_1^{(n)}, a_2^{(n)}, \ldots, a_N^{(n)}$, where $a_l^{(n)}) \in \{0, 1\}$ is the allele at locus $l \in \{0, ..., N\}$ for haplotype n.

For example, in case of $N = 3$ loci, the sequence 001 represents the haplotype with alleles 0 at locus 1 and locus 2 and alleles 1 at locus 3. One obvious conclusions is - by our assumption that each mutation occurs at a different locus - that the presence of all possible haplotypes implies the occurance of a recombination event somewhere in the evolutionary history (Song and Hein [2005]). However, some of the recombination events are not detectable as they do not generate a new haplotype pattern. For example, consider two haplotypes in a random mating population:

$$B_0 = 0 \quad 0 \quad 0$$

$$B_2 = 0 \quad 1 \quad 0$$

Let's consider the case, where there is a recombination between locus 1 and locus 2. The two possible haplotypes that we can observe after the recombination happened are exactly the same with the original ones. This is also the case even if we take a recombination event between locus 2 and locus 3. Therefore, in both cases the recombination events are not detectable and we therefore have to focus on events that do create new haplotype patterns.

Another limitation is that more than a single recombination can happen between two loci. Only if that number is odd, an actual recombination is observed. It is impossible to infer the exact number of recombination events (or rather cross-overs) that have occurred in a given sample. That is not the goal of the present thesis. Again,

we focus on the outcome of changes in haplotype patterns. The model presented so far, allows reconstruction of genetic history as far as it is observable by genetic markers. Biological knowledge suggests that there are also genetic events that cannot be captured by our model.

A first summary of the algorithm to generate the catalogue comprises the following steps:

1. consider ancestral haplotype B_i which is the only haplotype in the ancestral population
2. consider all possible sequences of events (mutation and/or recombination) for the given number of loci
3. collect all possible haplotype patterns after applying event sequences from the previous step to the ancestral haplotype
4. calculate the frequency pattern (allele/haplotypes frequency; discussed in Chapter 4), we assume uniform distribution
5. compute Higher order LD (discussed in Chapter 4)

This algorithm generates a comprehensive list of possible haplotype patterns together with their genetic history. The last two steps are needed for data applications and are discussed later.

3.2 Example for three loci

Consider the case of three bi-allelic markers. For the given DNA segment with three loci, the following haplotypes might be observed, resulting in 2^3 possible haplotypes.

locus 1	locus 2	locus 3
0	0	0
1	0	0
0	1	0
1	1	0
0	0	1
1	0	1
0	1	1
1	1	1

Table 3.1: Haplotypes for three bi-allelic loci; 0 denotes the wild type DNA composition and 1 a mutation/SNP

However, in biological data not all possible haplotypes might be observed. There are some potential explanations for this phenomenon. Either some recombination events have never occurred between specific loci, or recombination events have taken place but the resulting haplotype leads to lethality and is not observed in the population, or we have a bottleneck effect or very low haplotype frequencies (Griffiths and Marjoram [1996]).

Now, we consider the following five haplotypes from three loci:

$$B_0 = 0 \quad 0 \quad 0$$

$$B_2 = 0 \quad 1 \quad 0$$

$$B_3 = 1 \quad 1 \quad 0$$

$$B_4 = 0 \quad 0 \quad 1$$

$$B_5 = 1 \quad 0 \quad 1$$

In this example, the B_0 haplotype indicates the base haplotype, i.e the DNA segment in which no mutation has yet occurred. B_2 might have arisen from a mutation at locus 2. B_5 might have been arisen from a mutation at locus 1 on the background B_2. B_4 is the result of a mutation at locus 3. B_5 cannot explained by a mutation at locus 1 on a B_4 background based on the infinite sites assumption, but has to result from a recombination.

A very common way of illustrating the history of a sample of chromosomes, is the ancestral recombination graph (ARG). We can describe and trace the evolutionary history as an inverted tree. In our examples, the ancestral recombination in Figure 3.1, describes the genealogy of our sample $B = \{B_0, B_2, B_3, B_4, B_5\}$ and shows the importance of recombination events to generate new haplotypes and increase the genetic diversity. Therefore, the evolution of each haplotype is unfolded backward in time.

The root of the tree is haplotype B_0 : 0 0 0. Moreover, M_l denotes the mutation event with $l = \{1, 2, 3\}$ to indicate where the mutation takes place. Based on the plot, B_4 haplotype has been arisen from a mutation at locus 3. B_5 haplotype derives from a mutation at locus 3 and subsequently a mutation at locus 1. The haplotype B_2, is derived from a mutation at locus 2. In this tree only one recombination event has occurred which we denote with $R(B_i, B_j, pos)$ where $B_i, B_j \in \{B_n\}$, $B_i \neq B_j$. The argument pos indicates the position where the recombination occurs i.e $pos \in \{1, 2, 3, \ldots, N-1\}$ and N is the number of loci. Based on the graph the recombination event has occurred between B_2 and B_5 haplotypes which generated B_3 haplotype. This example ARG represents one evolutionary history from the many that exists for the given data $B = \{B_0, B_2, B_3, B_4, B_5\}$.

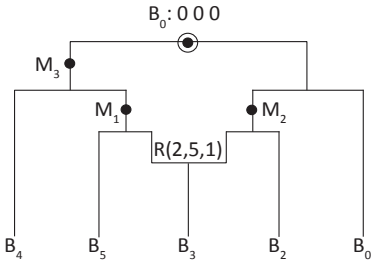


Figure 3.1: A hypothesized ancestral recombination graph for one example, shows the effect of recombination in evolutionary process

Note, that B_0 represented the ancestral sequence in this example. The ancestral sequence is not known in general. Subsequently, we consider all possible haplotypes as ancestral haplotypes.

3.3 Algorithm to construct the catalogue

Here, we describe the algorithm that explores all genetic event sequences in detail.

The algorithm considers vectors of indicator variables, where each entry indicates the presence or absence of a haplotype. By the assumption of no loss of alleles, a haplotype never disappears from a pattern once created. We use index $i \in \{0, 1, 2, 3, \ldots, 2^N - 1\}$ which corresponds to the haplotype represented by the binary representation of i and consider bi-allelic loci.

Initially, the algorithm starts with a vector containing a single "1", the base haplotype B_i, where $B_i \in \{B_n\}$ for $n \in \{0, \ldots, 2^N - 1\}$. In the case of three loci, mutations

can occur in different order starting from either the first, second or third loci. Therefore we have to consider all possible permutations on a set of 3 loci where a mutation can take place. Specifically, for 3 loci, there are $3! = 1 \cdot 2 \cdot 3 = 6$ permutations of $\{1, 2, 3\}$, namely $\{1, 2, 3\}, \{1, 3, 2\}, \{2, 1, 3\}, \{2, 3, 1\}, \{3, 1, 2\}$, and $\{3, 2, 1\}$. Between mutations a number of recombinations is tried to generate new haplotypes. Since we start with a single haplotype, we begin with a mutation instead or recombination, because in this case only a mutation can lead to a new haplotype.

Taking into consideration all these different order of mutations can lead to different haplotypes. For 3 loci, four possible sequence of events can occur:

- $M \rightarrow M \rightarrow M \rightarrow R$
- $M \rightarrow R \rightarrow M \rightarrow M \rightarrow R$
- $M \rightarrow M \rightarrow R \rightarrow M \rightarrow R$
- $M \rightarrow R \rightarrow M \rightarrow R \rightarrow M \rightarrow R$

Here, M indicates a mutation event and R any number of recombinations. Recombinations are performed between all possible pairs of haplotypes and all possible locations and the result checked for new haplotypes. This recombination procedure will be terminate when no new haplotypes could be generated. The results are saved in a vectors with 8 elements, where each element indicates the existence of a haplotype (see section 3.4). These entries are annotated with the sequences as listed above. If two sequences lead to the same pattern, catalogue entries are merged by concatenating the alternative sequences. We detail the algorithm by the following pseudo-code.

1. For all base haplotypes B_i
   (a) Initialize current haplotype pattern with B_i.
   (b) For all permutations $P_L = (p_{l_1}, ..., p_{l_N})$ of loci
   (c) Call [Mutation function] with current permutation p, haplotype B_i, current pattern

Mutation function : Take first element e of provided permutation p, remainder p_r

i. Apply mutation at e to provided haplotype h
ii. Add new haplotype to current pattern, store new pattern with current annotation

iii. Repeat: call [Recombination pattern] with current pattern, p_r, stop if pattern did not change

iv. Return current pattern

Recombination function :

(a) For all pairs of haplotypes, all recombination positions

i. Apply recombination to current pair of haplotypes at given position

ii. If new haplotype is generated, add haplotype, store new pattern with current annotation

iii. Call [Recombination function] with current pattern, p_r

iv. For all haplotypes in current pattern h

A. Call [Mutation function] with p_r, h, current pattern

(b) Return current haplotype pattern

In the above, "store" means that the current pattern is saved in a global structure. Whenever a pattern is saved it is merged with pre-existing descriptions for the pattern as described above.

3.4 Structure of catalogue

Each catalogue entry is comprised of a character vector describing the genetic events that happened and an indicator vector containing information on which haplotypes exist. For three loci, the catalogue contains a list of vectors with 8 entries (Table 3.2). Each position represents the existence of one of the specific haplotypes ($B0, ..., B7$) in the population. We denote with 1 the presence of a haplotype and 0 its absence. Each indicator, in turn, represents a haplotype.

Next, some examples from the catalogue are shown. The first sub-list (Figure 3.2), indicates that only the haplotype B_0 in position 1 appears in the sample population. The character string B_0 therefore describes the history in this case.

```
$B0
[1]  1  0  0  0  0  0  0  0
```

Figure 3.2: A catalogue entry which shows that only one haplotype (B_0) appears in the sample population

	position of indicator							
position	1	2	3	4	5	6	7	8
Haplotypes	B_0	B_1	B_2	B_3	B_4	B_5	B_6	B_7
locus 1	0	1	0	1	0	1	0	1
locus 2	0	0	1	1	0	0	1	1
locus 3	0	0	0	0	1	1	1	1

Table 3.2: An overview about the haplotypes in each position of the vector for three loci.

In Figure 3.3 we observe two haplotypes (position 1 and 2). Two possible histories explain this pattern which are given as '[history1,history2]'. The two histories are starting with base haplotype B_0 or B_1 each time followed by a mutation at locus M_1 (one time a 0 is flipped into a 1 and the other time the other way round).

```
$ ‘ [ B0–>M1, B1–>M1 ] ‘
[1]  1  1  0  0  0  0  0  0
```

Figure 3.3: A catalogue entry which shows that two haplotypes (B_0 and B_1) appear in the sample population

The mutation (M) or recombination (R) events which lead to the appearance of a specific haplotype are depicted above the specific vector. In Figure 3.4, the population contains four haplotypes $\{B_0, B_1, B_2, B_3\}$. The genetic history is described by a more complicated, nested structure with several alternatives. Briefly, in order to explain this pattern, there are four potential base haplotypes onto which a combinations of mutations (M) and recombination (R) events is applied. For example starting from haplotype B_0, a mutation at either locus 1 (M_1) or at locus (M_2) results in B_1 and B_2 haplotypes respectively. Subsequently, a recombination event (R_1) between locus 1 and 2 for the haplotypes B_1 and B_2 leads to haplotype B_3. To keep notation manageable, the particular haplotypes that were recombined are not mentioned.

```
$ ‘ [ B0−>[M1–>M2–>R1 , M2–>M1–>R1 ] , B1−>[M1–>M2–>R1 , M2–>M1–>R1 ] ,
    B2−>[M1–>M2–>R1 , M2–>M1–>R1 ] , B3−>[M1–>M2–>R1 , M2–>M1–>R1 ] ] ‘
[ 1 ]  1  1  1  1  0  0  0  0 ,
```

Figure 3.4: A catalogue entry which shows that four haplotype (B_0, B_1, B_2, B_3) appear in the sample population

The catalogue for three loci has 154 entries, which is different from 256 (2^8 combinations of 0 and 1) that are possible theoretically. This discrepancy is discussed below.

3.5 Complete catalogue for two loci

As the descriptions of the histories of the catalogue for three loci become very long (several 1000 characters), we here focus on the minimal case of two loci for which the complete catalogue can be shown. We assume that we have two bi-allelic markers and we again denote the haplotypes as $B = \{B_n\}$, where $n = \{0, 1, 2, 3\}$, $B_n = (0, 1)^2$. In this particularly example, it produces the following four (2^N; N number of loci) haplotypes:

locus 1	locus 2
0	0
1	0
0	1
1	1

Table 3.3: Haplotypes for two bi-allelic loci; 0 denotes the wild type DNA composition and 1 a mutation/SNP

For two loci, the catalogue consists of 13 entries out of the possible 16.

position	1	2	3	4
Haplotypes	B_0	B_1	B_2	B_3

Table 3.4: An overview about the haplotypes in each position of the vector for two loci.

The Table 3.5 depicts the full catalogue for 2 loci. We use the same notation as previously described, by denoting M the mutation and R the recombination events.

Ancestor History	Observed Haplotypes
1. B_0	1 0 0 0
2. B_1	0 1 0 0
3. $[B_0 \to M_1, B_1 \to M_1]$	1 1 0 0
4. B_2	0 0 1 0
5. $[B_0 \to M_2, B_2 \to M_2]$	1 0 1 0
6. $[B_0 \to [M_1 \to M_2, M_2 \to M_1]$, (or) $B_1 \to M_1 \to M_2, B_2 \to M_2 \to M_1]$	1 1 1 0
7. B_3	0 0 0 1
8. $[B_1 \to M_2, B_3 \to M_2]$	0 1 0 1
9. $[B_0 \to M_1 \to M_2$,(or) $B_1 \to [M_1 \to M_2, M_2 \to M_1]$, (or) $B_3 \to M_2 \to M_1]$	1 1 0 1
10. $[B_2 \to M_1, B_3 \to M_1]$	0 0 1 1
11. $[B_0 \to M_2 \to M_1$, (or) $B_2 \to [M_1 \to M_2, M_2 \to M_1]$, (or) $B_3 \to M_1 \to M_2]$	1 0 1 1
12. $[B_1 \to M_2 \to M_1$, (or) $B_2 \to M_1 \to M_2$, (or) $B_3 \to [M_1 \to M_2, M_2 \to M_1]]$	0 1 1 1
13. $[B_0 \to [M_1 \to M_2 \to R_1, M_2 \to M_1 \to R_1]$,(or) $B_1 \to [M_1 \to M_2 \to R_1, M_2 \to M_1 \to R_1]$, (or) $B_2 \to [M_1 \to M_2 \to R_1, M_2 \to M_1 \to R_1]$, (or) $B_3 \to [M_1 \to M_2 \to R_1, M_2 \to M_1 \to R_1]]$	1 1 1 1

Table 3.5: The complete catalogue for two loci

The vectors 1, 2, 4 and 7 indicate the base haplotypes B_0, B_1, B_2, and B_3, respectively. The aforementioned vectors, describe the cases where only one haplotype is occurred.

Adding a mutation leads to new haplotypes. For example the vector 3, describes a sample population where two haplotypes are observed, the B_0 and B_1. Here we have two potential genealogical histories which result in the observed haplotypes. In details:

1. $B_0 \rightarrow M_1$: The ancestor haplotype is B_0. After a mutation at loci 1, the observed haplotype is the B_1.

2. $B_1 \rightarrow M_1$: Similarly with above, the ancestor haplotype is B_1 and after a mutation at loci 1 we get the B_0 haplotype.

In total, vectors 3, 5, 8, and 10 represent the cases of a single mutation occurring.

There are four haplotype patterns in the catalogue, which contain three haplotypes. For example the vector 12 describes three potential genealogical histories which lead to the observed haplotype B_1, B_2 and B_3. In details:

1. $B_1 \rightarrow M_2 \rightarrow M_1$: The ancestor haplotype is the B_1. The observed haplotypes were obtained by mutation at loci 2, resulting in the haplotype B_3. In the latter haplotype, a mutation at loci 1 leads to the haplotype B_2.

2. $B_2 \rightarrow M_1 \rightarrow M_2$: Here the sequence of events take place in similar manner with the above case. The ancestor haplotype is the B_2. The observed haplotypes were obtained by mutation at loci 1, resulting in the haplotype B_3. In the latter haplotype, a mutation at loci 2 leads to the haplotype B_1.

3. $B_3 \rightarrow [M_1 \rightarrow M_2, M_2 \rightarrow M_1]$: In this case, the ancestor haplotype is B_3. In order to have in the population the haplotypes B_1 and B_2, a mutation takes place at the ancestor haplotype either at locus 1 (leading to haplotype B_2) or at locus 2 (resulting to haplotype B_1).

Entries 6, 9, 11, and 12 represent the cases where three haplotypes have occurred.

Finally, vector 13, describes a case where all four possible haplotypes have been generated. The latter can be achieved by different sequence of events. In contrast with the previous cases, now we also see recombination (between locus 1 and 2) events. For example, the ancestor haplotype for the first below sequence of events is the the B_0. A mutation at locus 1 or at locus 2 for the haplotype B_0, leads to haplotype B_1 or B_2 respectively. A recombination between locus 1 and 2 between the haplotypes B_1 and

B_2 results to haplotype B_3. Similarly we can explain the rest of the below sequence of events.

1. $[B_0 \to [M_1 \to M_2 \to R_1, M_2 \to M_1 \to R_1]$
2. $B_1 \to [M_1 \to M_2 \to R_1, M_2 \to M_1 \to R_1]$
3. $B_2 \to [M_1 \to M_2 \to R_1, M_2 \to M_1 \to R_1]$
4. $B_3 \to [M_1 \to M_2 \to R_1, M_2 \to M_1 \to R_1]]$

All histories mentioned earlier occur as a subsequence of entry 13. As all haplotypes have been generated the algorithm always stops as no new haplotypes can be generated any more.

One example of a missing haplotype pattern is the entry $(1, 0, 0, 1)$, i.e. B_0 and B_3 form the population. This pattern is impossible as B_3 contains two "1" alleles, whereas B_0 does not contain any. To go from B_0 to B_4 (or the other way round) two mutations would have to occur at once, which we have excluded by our population model.

Chapter 4

Linkage disequilibrium

Linkage disequilibrium (LD) is a important statistical concept for genetic mapping of traits in humans or other in organisms. It is defined as the non-random assortment of marker alleles at different loci in a population and usually is denoted by D or δ. It can also be seen as non-zero covariance between allele indicators. In the literature, sometimes is referred to other synonymous terms such as gametic phase disequilibrium or allelic association (Lewontin [1988]). Linkage equilibrium (LE) denote the absence of association between two loci; the alleles at each loci are independent.

The patterns of LD can been affected by many factors like genetic drift, recombinations, natural selection or by the geographical structure and changes in population size. As an example of the latter, the strength of LD relies on the number of founding haplotypes (after a population bottleneck occurred), on the size of population (small sized population drifts with bigger effect) and on the number of generations (the number of populations for which the population existed). Consequently, we expect weak allelic association (decay of LD) when there have been more meiosis thus, more opportunities for recombination. However, it difficult to interpret the raw covariance δ in absolute terms. For this reason, other standardized measures have been proposed. The most important among them are the coefficients D' and r^2, which play a role in different applications. In the current thesis, the parameter D' is more useful since it can be related to the occurrence of historical recombination, while r^2 (the squared correlation of allele indicators) is more appropriate for the design of association studies (Balding [2006]).

4.1 Pairwise linkage disequilibrium

Let us consider the case of two bi-allelic markers located on the same chromosome. If the alleles are far apart on the chromosome, then recombination events occur a higher frequency as compared to alleles more closely located together. Alleles which are close together on the same chromosome will recombine less frequently, hence having the potentially higher LD.

There are four possible allele combinations at two loci:

locus 1	locus 2
0	0
0	1
1	0
1	1

Table 4.1: Haplotypes for two bi-allelic loci

Let $p_{i\cdot}$ and $p_{\cdot j}$ be the marginal (or single) frequencies of alleles i and j at loci 1 and 2 respectively, where:

$$i, j = \begin{cases} 0, & \text{if allele is 0} \\ 1, & \text{if allele is 1} \end{cases} \tag{4.1}$$

Thus $p_{0\cdot}$ and $p_{1\cdot}$ denote the allele frequency of allele 0 and 1 at loci 1 respectively. Moreover, we assume that haplotypes follow a multinomial distribution $h \sim Mult(1, p_{ij})$, where p_{ij} denotes the haplotype frequency (i, j). Thus, p_{01} denotes the probability of a randomly selected haplotype being the haplotype with alleles 0 or 1 at the two loci 1 and 2. The marginal frequency of alleles 1 at locus 1 is obtained by summing up the frequencies of all haplotypes contain alleles 1 at locus 1:

$$p_{1\cdot} = p_{10} + p_{11}$$

Similarly, we can obtain the marginal frequencies of the other alleles.

The relationship between haplotype frequencies and alleles frequencies depends on whether the alleles at two loci are dependent (i.e. in LD) or independent (i.e. in LE).

1. If the alleles at two loci are independent from each other then the joint probability is equal to the product of marginal probabilities. For instant, $p_{11} = p_{1\cdot} \cdot p_{\cdot 1}$

2. If the alleles at two loci are not independent from each other, then joint probabilities deviate from the product of marginal probabilities (by ± D).

The probabilities can be arranged in a contingency table:

			locus 2: 0	locus 2: 1	Marginal prob.
locus 1	0	actual	p_{00}	p_{01}	
		expect	$p_{0\cdot}p_{\cdot 0}$	$p_{0\cdot}\ p_{\cdot 1}$	$p_{0\cdot}$
	1	actual	p_{10}	p_{11}	
		expect	$p_{1\cdot}p_{\cdot 0}$	$p_{1\cdot}\ p_{\cdot 1}$	$p_{1\cdot}$
		Marginal prob.	$p_{\cdot 0}$	$p_{\cdot 1}$	1

Table 4.2: Association between two bi-allelic loci, showing the actual haplotype frequencies and the expected haplotype frequencies when the loci are in linkage equilibrium. The marginal probabilities represent the allele frequencies

The pairwise coefficient of linkage disequilibrium for two bi-allelic markers is defined as:

$$D_{ij} = p_{ij} - p_{i\cdot}p_{\cdot j} \tag{4.2}$$

As we mentioned above, the definition of LD provides a measure of deviation from the independence case. If two events A and B are independent then $P(A \cap B) = P(A)P(B)$, therefore departure from independence can be measured as $D = P(A \cap B) - P(A)P(B)$ (Mueller [2004], Gorelick and Laubichler [2004]). This concept can be utilised for expressing the independence between two loci. D is also the covariance between allele indicators at two loci. In the case of indicators independency, that is equivalent to uncorreletedness.

In a bi-allelic system, all D_{ij}'s only differ by sign, as can easily be seen by summing the D_{ij} marginally over one of the two loci:

$$D_{ij} = (-1)^{i+j}D \tag{4.3}$$

where,

$$D = p_{11} - p_{1\cdot}p_{\cdot 1} \tag{4.4}$$

Thus, any choice of reference alleles leads to the same absolute value. Note, that the value of the D is sensitive to the marginal probabilities and bounds for D depend

on allele frequencies. Bounds for D are given as follows:

$$D \geq -min(0, 1 - p_{1\cdot} - p_{\cdot 1}) - p_{1\cdot}p_{\cdot 1}, \tag{4.5}$$

$$D \leq min(p_{1\cdot}, p_{\cdot 1}) - p_{1\cdot}p_{\cdot 1} \tag{4.6}$$

4.2 Standardised LD

To better interpret D, standardized versions have been proposed. The underlying concept is to normalise D to take values in the fixed interval $(0, 1)$. The standardised LD is denoted as D$'$ and is defined as:

$$D'_{ij} = \frac{D_{ij}}{D_{ij}^{max}} \tag{4.7}$$

where,

$$D_{ij}^{max} = \begin{cases} min(p_{1\cdot}, p_{\cdot 1}) - p_{1\cdot}p_{\cdot 1} & : D_{ij} > 0 \\ -min(0, 1 - p_{1\cdot} - p_{\cdot 1}) - p_{1\cdot}p_{\cdot 1}) & : D_{ij} < 0 \end{cases} \tag{4.8}$$

Two extreme cases of D$'$ are worth consideration:

1. D$'$=0, which is the case of complete equilibrium (D=0)
2. D$'$=1, when at least one haplotype is missing, and there is no evidence for recombination between markers.

When the value of D$'$ is lower than 1, there is evidence for the existence of historic recombination events. When we observe high value for D$'$, that does not imply that the two SNPs can predict each other well. Also note, that D' does not reparametrize D, i.e. we cannot re-commute D from D' in general. Therefore also a signed version can be considered, i.e. D' lives on $(-1, 1)$. We will consider a signed version in the following.

4.3 High order of Linkage disequilibrium

LD can be generalized to more than two loci. LD for three or four loci has been considered in the literature before(Weir [1996]). It turns out that these intuitive definitions

coincide with joint cumulants and LD can therefore be defined for any number of loci using their joint cumulants.

As an illustration, we show the formula for 3 bi-allelic loci which is a sum of haplotype frequencies of all possible subsets of the loci (i.e. allele frequencies, pairwise frequency, haplotype frequency of all three loci). The joint cumulant of three loci is given by:

$$D_{ijk} = p_{ijk} - p_{i\cdot\cdot}p_{\cdot jk} - p_{\cdot j\cdot}p_{i\cdot k} - p_{\cdot\cdot k}p_{ij\cdot} + 2p_{i\cdot\cdot}p_{\cdot j\cdot}p_{\cdot\cdot k}, \tag{4.9}$$

where i, j, k denote the three loci and variables p_L denote marginal haplotype frequency for subset (of loci) L for which one of the alleles at each locus is chosen in a fixed way for all terms p_L. As a convention we will assume alleles $0, 1$ and p_L to denote the haplotype of alleles 1 at each locus in L.
The standardisation method that have been described in section 4.2 can be generalised to establish standardised LD measurements for any number of loci (3,...,N). This method is described in Appendix A (8). This standardization has been developed elsewhere (Balliu et al.) and the formulas are more involved. A similar interpretations as for the pairwise standardised LD exist, namely, standardised higher-order LD being 1, has the interpretation that at least one haplotype is unobserved. This implies that not all possible recombinations have occurred yet.

For completeness, LD for N SNPs can be defined as follows. Briefly, we consider $A = \{A_1, A_2, ..., A_N\}$ to be a set of random variables with indicator variable $A_j \in \{0, 1\}$, $j = 1, 2, .., N$. If $Par(A)$ refers to the set of partitions of set A into non empty subsets, then the joint cumulant of the set of random variables A is:

$$f(A) = f(A_1, A_2, ..., A_N) = \sum_{\tau \in Par(A)} (-1)^{|\tau|-1}(|\ \tau\ | -1)! \prod_{\beta \in \tau} \mathbb{E}\Big(\prod_{A \in \beta} A \Big) \tag{4.10}$$

where $\tau \in P(A)$ ($|\ \tau\ |$ denotes the cardinality of set τ) and each $\beta \in \tau$ is a block, i.e. a member of the partition (more details in Appendix 1 (8)). The terms $\prod_{A\in\beta} A$ correspond to marginal haplotype frequencies. For example, the Function 4.10 corresponds to the mean for one loci(N=1), i.e. $f(A_1) = E(A_1)$, while for 2 loci (N=2) corresponds to the covariance, i.e. $f(A_1, A_2) = E(A_1A_2) - E(A_1)E(A2)$, therefore to the pairwise LD.

Chapter 5

Data application of visualisation and genetic interpretation of Higher-order Linkage Disequilibrium.

The aim of this chapter is to visualise and interpret Higher-order Linkage Disequilibrium. Also, we will generalize genetic interpretations relating to recombination and mutation events with respect to the pair-wise situation.

In Chapter 3 and Chapter 4 we have described the way we have contructed the catalogue. We use the HapMap dataset to obtain SNP data for human chromosome 21. Chromosome 21 has been implicated in many genetic disorders among them Down syndrome as well as chromosomal translocations resulting in leukemia such as Acute Myeloid Leukemia (AML). The HapMap data considered in this analysis, consists of 120 haplotypes where we consider a SNP arbitrarily selected in chromosome 21 and subsequently investigated 100 SNPs downstream of the selected one. For the purpose of this chapter, I have selected arbitrarly one SNP (rs3843783) and then selected 3 SNPs based on their location; one intronic (rs2829806), one exonic (rs1057885) and one intergenic (rs11088561) in order to assess whether detectable differences are visible for the different classes of SNPs(Consortium [2003]). A window of SNPs around these anchor SNPs is then analysed together. For graphical representation purposes, we will use heatmap plots which allow to depict relationships in a data matrix where data values are mapped to a colour range.

The data analysis can be divided into the following steps:

1. Consider the catalogue for three SNPs
2. For each catalogue entry assume a distribution, for each non-empty subset of three SNPs, compute LD and standardized LD (or D').
3. For the real data, around an anchor SNP, select a window of SNPs and consider a sliding window of three SNPs in this set
4. Compute LD and standardized LD for each sliding window of three as for the catalogue.
5. Create a Euclidean distance matrix for the catalogue and HapMap dataset by computing the Euclidean distance of the real data cumulant signature with every entry of the catalgue.
6. Visualize the patterns using Heat maps plots.

5.1 Compute LD and standardized LD for the catalogue

Each catalogue entry contains an existence pattern of haplotypes. For the purpose of computing standardized joint cumulants, the existence pattern is transformed into a uniform distribution on the existing haplotypes. Standardized joint cumulants are then computed for all non-empty subsets of loci, resulting in seven entries for three loci. The choice of the uniform distribution leads to joint cumulants of 0 if all haplotypes exist for a subset (unless a single locus). This corresponds to the case of LE, between alleles where recombinations have been occurred, i.e. we assume that once a recombination has occurred further recombinations have eliminated all remaining correlation. For example, uniformly distributed haplotypes on two loci lead to lack of correlation between the SNPs. Missing haplotypes lead to a standardized cumulant of either -1 or 1.

As actual haplotype frequencies in a given sample are subject to genetic drift and are not uniform in general, the comparisons should not depend on a particular choice of frequencies. We achieve this by excluding marginal frequencies from the cumulant signature when performing actual comparisons (see below). The cumulant signature is added to the catalogue.

5.2 Compute LD and standardized LD for real data (HapMap project)

Data from the HapMap project is offered in a phased version, i.e. haplotypes have been determined from genotype data using family information and statistical inference. Haplotype frequencies can therefore directly be estimated using sample frequencies.

As described above, for chromosome 21, we selected some anchor SNPs and for each such anchor SNPs a window of SNPs around it (in this example 100 SNPs). Within each such window, continuous sets of three SNPs are selected and analysed in a sliding window approach. That means that two subsequent windows will overlap by two SNPs. The data set consists of 60 individuals, resulting in 120 haplotypes. This dataset has to be considered very small. The fact that this is a small sub-sample from the population implies that the assumption that no haplotypes were lost is violated. As a matter of fact, the sampling process can be considered a bottleneck effect. Interpretation of the analysis has therefore to take this into account.

Analogously to the catalogue, standardized LD is calculated based on sample haplotype frequencies. In total, 100 such cumulant vectors are generated for the 100 SNP windows of size three.

5.3 Create a Euclidean distance matrix

In order to describe ancestry of the SNP data, similarity of the data with catalogue entries is computed. In principle, there are many possible ways to find numeric similarities. Here we use the Euclidean distance between cumulant vectors as a distance measure. In order to reduce or eliminate the influence of genetic drift on the analysis, cumulant entries describing allele frequencies were removed from the vectors prior to calculating the Euclidean distance. Thereby, four elements remained in the cumulant vectors (three pair-wise, and one three-wise LD entries).

Let us denote by $\kappa_1 = (x_1, x_2, x_3, x_4)$ an cumulant signature (from the 154) from the catalogue with the standardised LD values. Similarly, let us consider $\kappa_2 = (y_1, y_2, y_3, y_4)$ to be a vector including the standardised LD values from the data set. Then, the Euclidean distance of these two vectors will be

$$d(\kappa_1, \kappa_2) = \sqrt{\sum_{i=1}^{4} (x_i - y_i)^2} \tag{5.1}$$

Following the aforementioned principle, we can obtain a Euclidean distance matrix consisting of all distances between the catalogue and HapMap datasets.

$$M = \begin{pmatrix} d_{1,1} & d_{1,2} & ... & d_{1,100} \\ d_{2,1} & d_{2,2} & ... & d_{2,100} \\ \vdots & \vdots & ... & \vdots \\ d_{154,1} & d_{154,2} & ... & d_{154,100} \end{pmatrix}$$

where $d_{i,j}$ denotes the Euclidean distance between i catalogue entry and j HapMap dataset. Hence,

$$M = [d_{ij}]$$

where $i = 1, 2, ..., 154$ and $j = 1, 2, ..., 100$.

5.4 Visualize the patterns using heatmap plots

Chromosome 21 from the HapMap project consists from 33863 SNPs. Initially, we selected arbitrary the 100th SNP and calculated the LD and D' for the next 99 SNPs, resulting in a total of 100 SNPs. Considering the Euclidean distance of the D' values between the catalogue and the current subset, we visualised that difference in a heatmap (Figure 5.1). Subsequently, we considered two SNPs, one lying in an intron (Figure 5.2) and another in an exon (Figure 5.3) respectively of the gene MRPL39 and repeated the aforementioned analysis for 100 SNPs in this genomic region. Finally we selected an intergenic SNP lying between the genes USP25 and C21ORF34 (Figure 5.4).

5.5 Interpreting heatmap plots

The aforementioned clustering analysis for the Euclidean distance of their D', reveals some distinct clusters (Figure 5.1, Figure 5.2, Figure 5.3, Figure 5.4). There is a group of events with the same Euclidean distance (red and/or black colour) and lack of recombination (few recombination/mutation events) since D' is different, meaning we cannot conclude anything about the genealogy of those SNPs. Moreover, there is a cluster (green colours) which are close to have the same genealogy, since they have a small Euclidean distance, meaning that the history between the SNPs and the catalogue entries cannot be distinguished based on Euclidean distance. Furthermore, we observe a cluster (orange colour) where the distance is close to 0, therefore the catalogue entries can explain the genealogy of the SNPs. Also, we observe a cluster with

mixed orange and green colours which is combination of the previous two states. It is of note, that some catalogue entries represent cases where not all mutations have occurred whereas the data set is conditioned on the fact that polymorphic data exists (i.e. the corresponding mutation has occurred). For example, catalogue entry 1 contains only the haplotype B_0. In this case, this sample population cannot be observed in real data based on data ascertainment. By ignoring allele frequencies, we make it impossible to detect a case where a mutation has "just" occurred (i.e. only very few alleles exist yet). This indicates that other choices than the Euclidean distance and the elimination of allele frequencies in the comparisons might be reasonable. The current plots a therefore only useful when the questions mentioned above are not important.

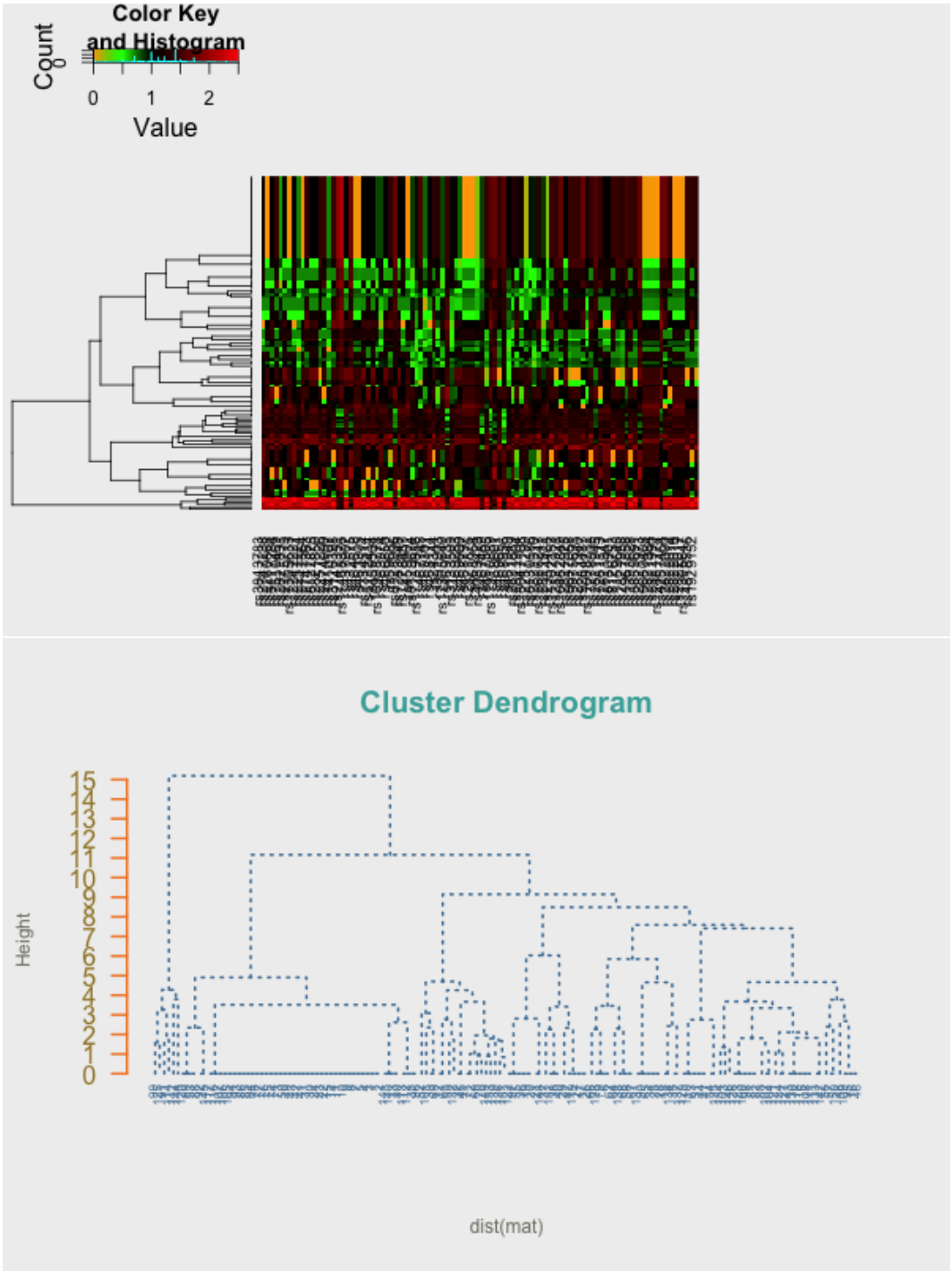


Figure 5.1: The heat map and the dendrogram plot of the Euclidean distance for the 100 arbitrary selected SNPs.

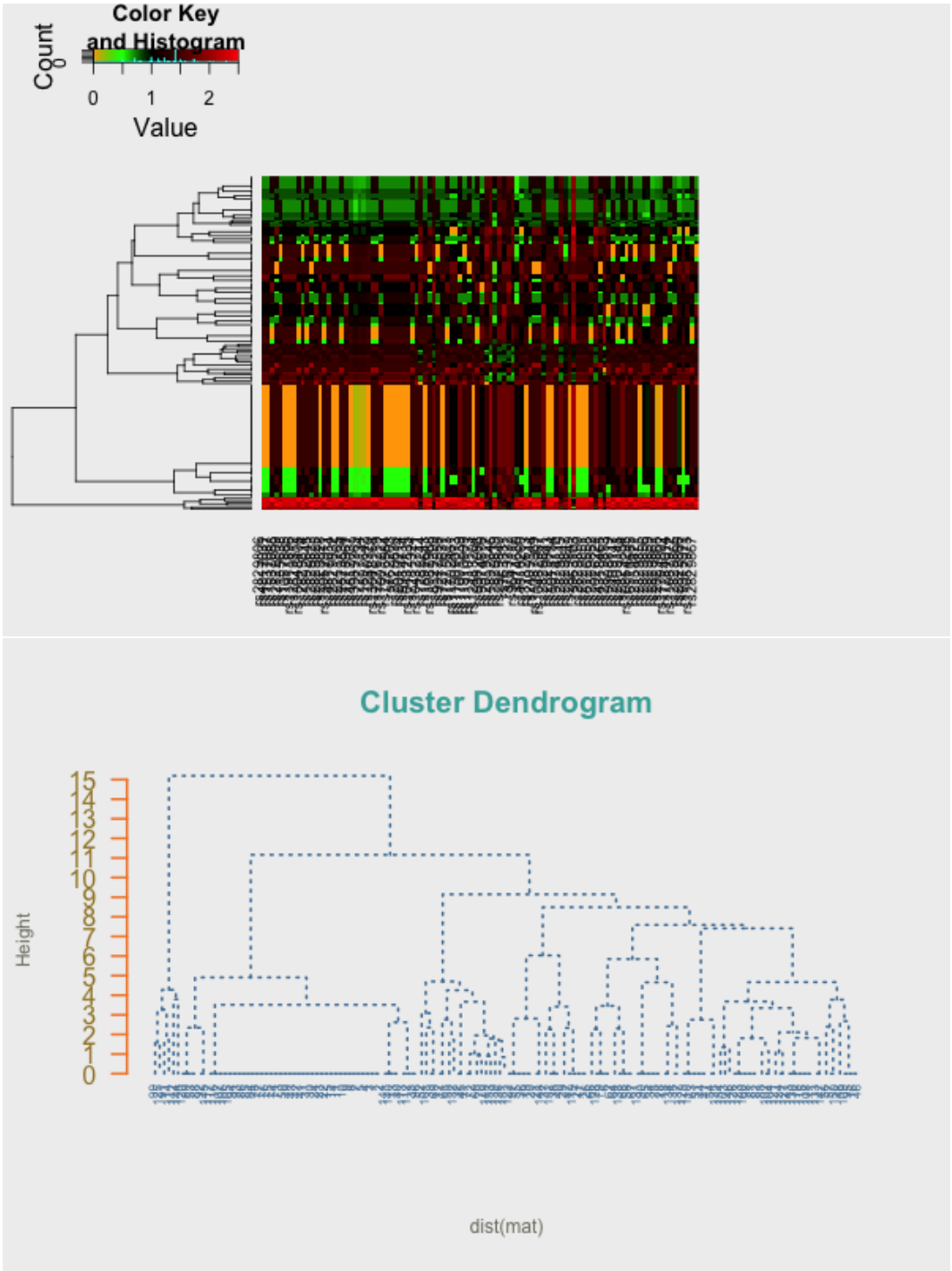


Figure 5.2: The heat map and the dendrogram plot of the Euclidean distance for the intronic SNP rs2829806.

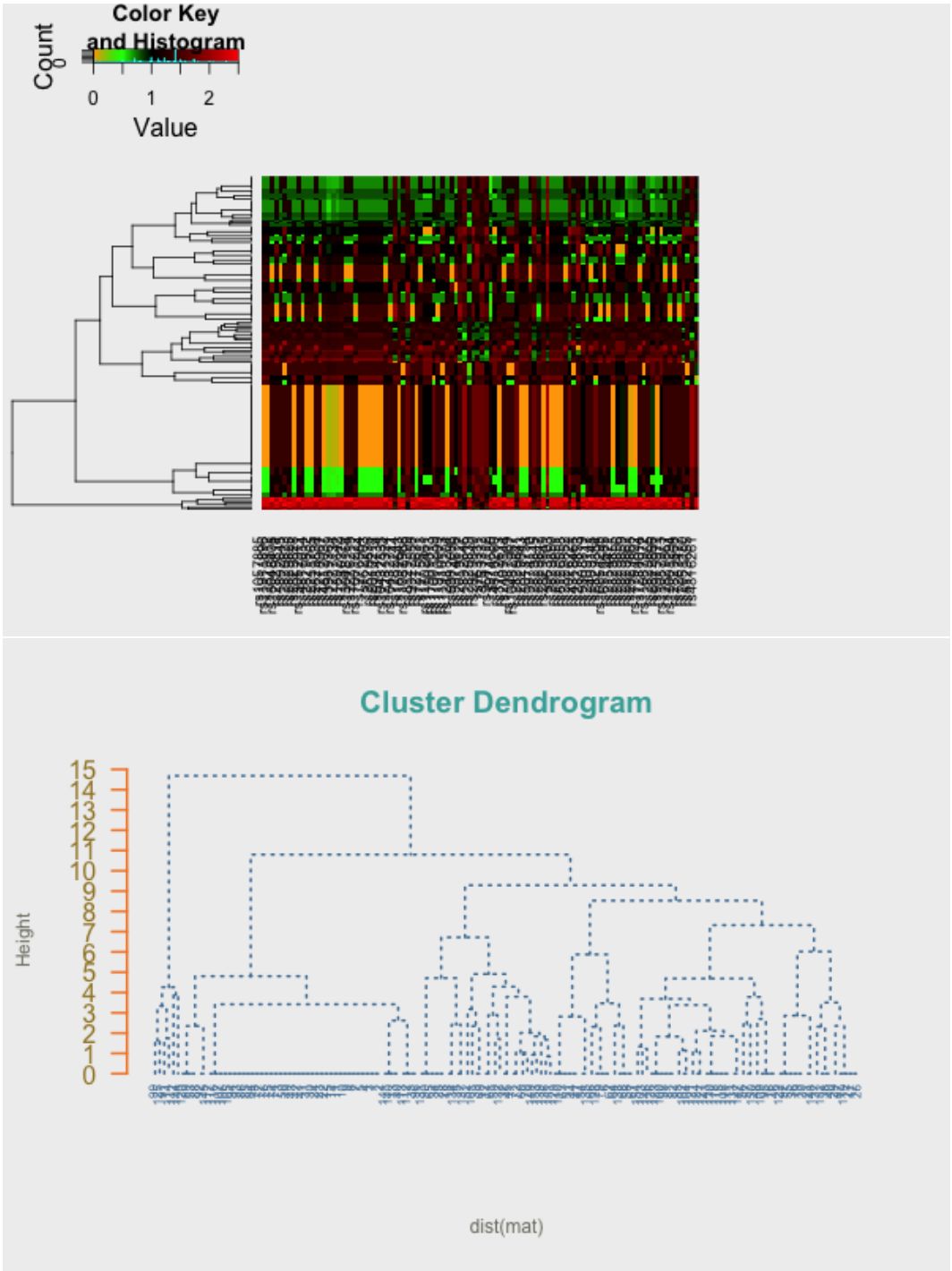


Figure 5.3: The heat map and the dendrogram plot of the Euclidean distance for the exonic SNP rs1057885.

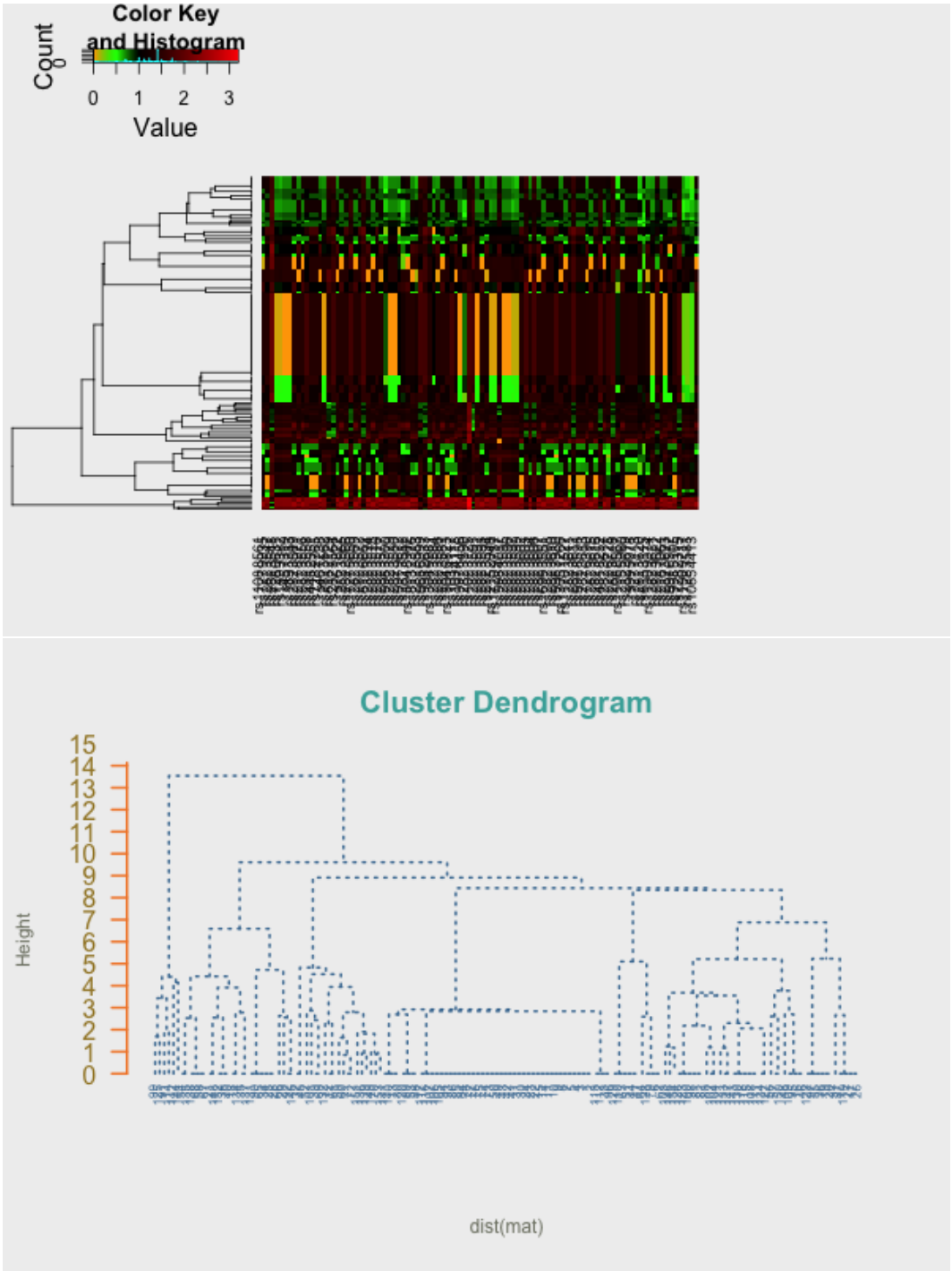


Figure 5.4: The heat map and the dendrogram plot of the Euclidean distance for the intergenic SNP rs11088561.

Chapter 6

Inference on higher order LD

In the previous chapters, we have introduced descriptive ways to analyse the data by means of heatmap plots. In this chapter, we will develop tests in order to assess which catalogue entry among two is closer to actual data. Therefore, it is possible to decide which of two ancestors is more likely to explain the data. We rely on bootstrap technique to develop the tests and apply them to the HapMap data.

6.1 Introduction to Monte Carlo methods

Monte Carlo methods (also known as Monte Carlo simulations) are computational tools which are increasingly used in recent years as a result of the great improvements in computer performance. It refers to statistical methods used to approximate solutions to problems through repeated random sampling. It can be applied to estimate parameters of interest and to develop statistical tests. Monte Carlo simulations have been applied in this thesis to make inference on the genetic history of given genotype data using hypothesis testing. The parametric bootstrap which generates datasets using repeated sampling from a known probability model has been used to develop the tests.

6.2 Parameter testing of parametric bootstrap procedure

In this thesis, we focus on the comparison of pairs of catalogue entries using parametric bootstrap techniques. Two catalogue entries are pre-specified and it is to be determined whether a given data set is more likely to stem from one of the histories.
Let us consider M bi-allelic loci and haplotype $h\{0,1\}^M$ as multinomially distributed, i.e. $h \sim Mult(1,p)$, where $p = (p_1, ..., p_{2^M})$ denotes the haplotype frequencies and

$\sum p_i = 1$.

Assume that we are interested to compare catalogue entries h_1 and h_2. Denote with κ_1 and κ_2 corresponding cumulant signatures. In principle, all the points in the space of cumulant signatures, that have equal distances from κ_1 and κ_2 represent the null hypothesis. We simplify this situation, by assuming that the a single representative point can been chosen. In the following we choose the mean κ_m of κ_1 and κ_2 as the null, hence,

$$\kappa_m = \frac{\kappa_1 + \kappa_2}{2}. \tag{6.1}$$

A justification of this simplification is, that we test the most direct path between the two histories. For illustration, imagine that κ_1 and κ_2 are located in a two dimensional space and κ_m is their mean (Figure 6.1). Then, we denote with d_1, d_2, the Euclidean distances of the cumulant signatures of catalogue entries κ_1 and κ_2 with κ_m, respectively. The null hypothesis can be restated as no differences between d_1 and d_2, $H_0 : d_1 = d_2$. We also consider the null hypotheses $H_0 : d_1 \leq d_2$ and $H_0 : d_1 \geq d_2$.

Rejecting the null hypothesis implies that either of the two distance d_1 and d_2, is smaller. Therefore one catalogue entry is a better explanation for the given data. We will to try to assess the significance of the hypothesis with the statistic T which is the difference of the distances d_1 and d_2.

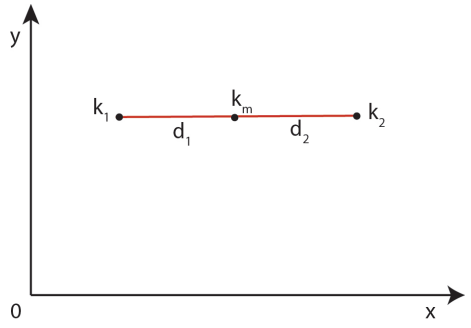


Figure 6.1: The plot illustrates the Euclidean distance of LD values between catalogue entry κ_1 and κ_2 with theκ_m. In this case the $d_1 = d_2$. The null hypothesis is always directly between two entries

6.3 Data generation under the Null hypothesis

In order to generate datasets under the null hypothesis under a parametric model, we will generate haplotypes, which follow the multinomial distribution $h \sim Mult(1, p)$. Haplotype frequencies p will be defined by mapping back κ_m to haplotype frequencies (Balliu et al.).

The following steps are performed:

1. Define which two entries from the catalogue are to be tested, with cumulant signatures κ_1, κ_2.
2. Compute κ_m which represents the null hypothesis.
3. Estimate allele frequencies from the data and replace allele frequencies in the cumulant signature κ_m with data frequencies. κ_1, κ_2 are computed based on uniformly distributed haplotypes (on those who exist) thereby resulting in arbitrary allele frequencies. Higher order LD parameters reflect presence/absence of haplotypes and do not depend on allele frequencies.
4. Transform the LD values (from the κ_m) into haplotype frequencies p.
5. The sample haplotype frequencies p are used to generate haplotypes from the corresponding multinomial distribution which consists of 3 loci and N individuals, where N is the number of individuals in the data.
6. Haplotype frequencies are estimated from the generated data and cumulant signatures are computed, denoted with $\kappa^{(i)}$.

6.4 Algorithm of parametric bootstrap

For the parametric bootstrap procedure, the following steps were defined. The below steps 1, 2, and 3 are described in detail in the previous section.

Algorithm
For fixed cumulant signature κ_1, κ_2:
Step 1 : Repeat for $i = 1, ..., B$; typically B=1000.
Step 2 : Draw random sample of size N from the multinomial haplotype distribution defined by k_m.
Step 3 : Estimate haplotypes frequencies and transform to cumulant signature.
Step 4 : Compute the Euclidean distances $d_1 = d(\kappa_1, \kappa^{(i)}), d_2 = d(\kappa_2, \kappa^{(i)})$
Step 5 : Compute the test statistic $T = d_2 - d_1$
Step 6 : Compute quantiles of T_{data} from

$$\hat{F}(T) = \frac{1}{B}\sum_{i=1}^{B} I\{x \geq T^{(i)}\}$$

In the implementation the Bootstrap samples are stored as a matrix with B columns and two rows. Each row represents the a catalogue entry and each column is the Bootstrap sample.

$$\begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,B} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,B} \end{pmatrix}$$

where $d_{i,j}$ denotes the Euclidean distance between cumulant signature κ_i and the cumulant signature of the bootstrapped sample.

After the collection of the bootstrap Euclidean distances, the test statistic is computed by row-wise differences. Finally, we obtain a vector which contains B bootstrap test statistics $T^{(i)}$

$$T = (d_{2,1} - d_{1,1}, d_{2,2} - d_{2,2}, \cdots, d_{2,B} - d_{1,B}) = (T^{(1)}, T^{(2)}, \cdots, T^{(B)})$$

Suppose that we wish to find a confidence level $1 - \alpha$ interval for the pairwise differences of distance measurements T. First, we sort the observed B i.i.d realizations to get $T_{(1)}, ..., T_{(B)}$ and then we use $Q_\alpha = T(\lfloor \alpha B + 0.5 \rfloor)$ to estimate the $\alpha \cdot 100\%$ quantile of the distribution of T. We can state with probability $1 - \alpha$ that the true difference is covered with $1 - \alpha$ probability in a long sequence of experiments, and will fall between $\alpha/2$ and $1 - \alpha/2$ quantiles of the bootstrap distribution of $\hat{T}$. The desired $100(1 - \alpha)\%$ is:

$$[Q_{\alpha/2}, Q_{1-\alpha/2}]$$

This procedure results in confidence intervals. For α=0.05 and $B = 1000$ the

confidence interval which corresponds to a confidence level of 95%, can be calculated as

$$[T_{(25)}, T_{(975)}]$$

6.5 Hypothesis testing using pairwise differences of distance measurements

We consider three hypotheses concerning distances d_{κ_1} and d_{κ_2}, the true Euclidean distances between the parameter vector of the data distribution and the catalogue entries. The hypotheses of interest are:

$$H_0^1 : d_{\kappa_2} \leq d_{\kappa_1} \quad vs \quad H_A^1 : d_{\kappa_2} > d_{\kappa_1}$$

$$H_0^2 : d_{\kappa_2} \geq d_{\kappa_1} \quad vs \quad H_A^1 : d_{\kappa_2} < d_{\kappa_1}$$

$$H_0^3 : d_{\kappa_2} = d_{\kappa_1} \quad vs \quad H_A^3 : d_{\kappa_2} \neq d_{\kappa_1}$$

These lead to the rejection rules for given bootstrap sample $T_{(1)}, T_{(2)}, ..., T_{(B)}$:

1. Reject H_0^1 if $d_{\kappa_2} - d_{\kappa_1} < Q_{(\alpha)}$
2. Reject H_0^2 if $d_{\kappa_2} - d_{\kappa_1} > Q_{(1-\alpha)}$
3. Reject H_0^3 if $d_{\kappa_2} - d_{\kappa_1} \in (Q_{(\alpha/2)}, Q_{(1-\alpha/2)})$

Two types of error can occur in statistical hypothesis testing. A Type I error occurs if the null hypothesis is rejected when the null hypothesis is true. A Type II error occurs if the null hypothesis is not rejected when it is false. In this thesis we investigate Type I error rate by simulations under the null and power of test when increasing the sample size of the population under certain alternative scenarios.

6.6 Performance evaluation

6.6.1 Simulations settings

Simulations can be used to evaluate the performance of this procedure for finite sample sizes. In our experiments, we have conducted 500 replications. We have analysed 12 different cases consisting of pairwise comparisons of two catalogue entries each. The pairs were chosen to represent pairs with both small and big Euclidean distance. First, we will assess how the distance influences the outcome of the analysis. Selecting two

catalogue entries based on their Euclidean distance, i.e. entries which are either close together (small Euclidean distance but above zero), far apart (maximum distance) or having an average distance, gives an impression on which ancestries can be distinguished for the realistic sample sizes. The Table 6.1 illustrates the Euclidean distances of examples investigated. Second, we investigated sample size, therefore, in each simulated datasets different sample size N have been used. Simulated datasets with N equal to 120, 240, 480 and 720 haplotypes (observations) were generated.

catalogue entries	Euclidean distance
1 and 91	2.29
1 and 120	0.5
1 and 62	1
72 and 1	1.72
72 and 60	0.23
72 and 101	0.52
103 and 114	2.16
103 and 146	0.23
103 and 128	0.971
62 and 99	2.5
62 and 1	1
62 and 147	0.577

Table 6.1: The Euclidean distance of standardised LD value between two catalogue entries

Illustrative example

The following example shows the bootstrap distribution for $N = 120$ haplotypes, and $B = 1000$ bootstrap samples. 500 replications were performed and data was generated under the null hypothesis. For this simulation, we compare entries 1 and 91 of the catalogue and analyse which of these entries is closer to observed data. From the Table 6.1 we can see that they have a Euclidean distance equal to 2.29. The Figure 6.2 and 6.3 visualize the bootstrap distribution of the test statistic.
In Figure 6.2 the red line indicates the quantiles 5% and 95% and the blue line the true expected value of the test statistic T. The right histogram has $T_{obs} = 0.77$, $Q_{5\%} = 0.488$ and $Q_{95\%} = 0.922$. Based on this plot we can test one sided hypothesis H^1 and H^2 and we can not reject either hypothesis. On the contrary, the resulting terms of the left plot, gives $T_{obs} = 0.48$, $Q_{5\%} = 0.51$ and $Q_{95\%} = 0.954$. For this example we reject the hypothesis H^1 whereas, we cannot reject hypothesis H^2.

In Figure 6.3 the red lines indicates the upper and low bound of the 2.5% and 97.5% quantiles and the blue line the true observed test statistic T (see above) . With these plots we can test two tailed hypothesis H^3. The right plot gives $T_{obs} = 1.020$, $Q_{2.5\%}$ and $Q_{97.5\%}$ and we reject hypothesis H^3. Whereas, the left plot indicates that the data are consistent with the null hypothesis and we can not reject the H^3 since T_{obs}, which is equal to 0.48, is lying on the confidence interval $[0.478, 1.012]$. Figure 6.4 illustrates the distribution of the test statistic T_{obs} of the 500 simulations. Comparing with the rest of the histograms, we can conclude that under the same null hypothesis T is approximately normal distributed with mean around 0.7.

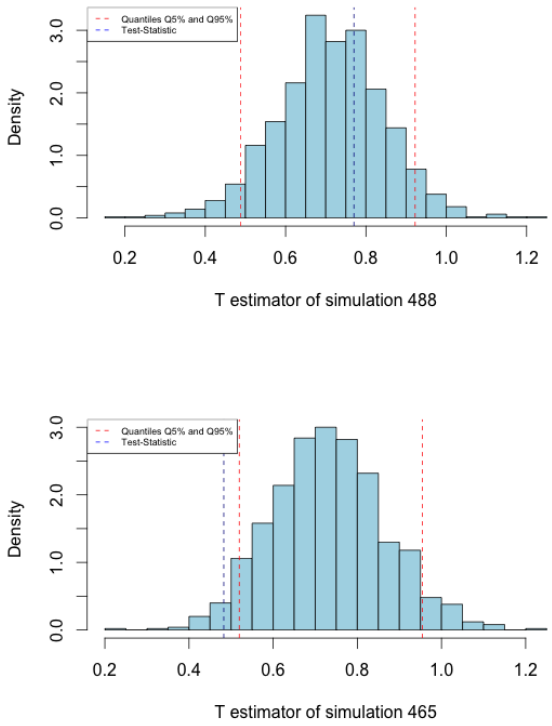


Figure 6.2: Both histograms depict the sampling distribution of the test statistics when we test the catalogue entries 1 and 91 for two experimental simulations. The red line indicates the quantiles 5% and 95% and the blue line the observed test statistic T. We can test the hypotheses H^1 and H^2.

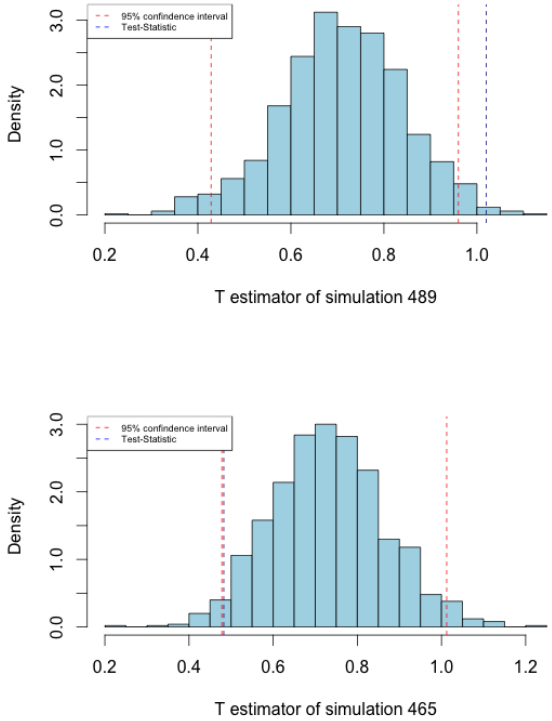


Figure 6.3: Both histograms depict the sampling distribution of the test statistics when we test the catalogue entries 1 and 91 for two experimental simulations. The red line indicates the 95% confidence interval and the blue line the observed test statistic T. We can test the hypothesis H^3.

6.6.2 Type I error

Table 6.4 lists all results for type I error simulations for all tests. We test at the 0.05 level and theoretically, the probability to reject the null hypothesis should be $\alpha = 0.05$. In simulations studies, finite sample properties are investigated and the estimated Type I error can deviate, but it should be close to the nominal level $\alpha = 0.05$ because the simulated dataset was generated under the null hypothesis. With $s = 500$ replications, the standard error of rejection rate is approximately $\sqrt{0.05 \cdot 0.95/s} = 0.0097$. Therefore, the Type I error rate lies in the interval [0.030,0.069] is acceptable taking into account sampling fluctuation caused by the replications. From the Table 6.4, we observe that all the tests do not maintain the nominal level $a = 0.05$. However, if a type I error is below 0.05 then this does not invalidate the test but it makes it

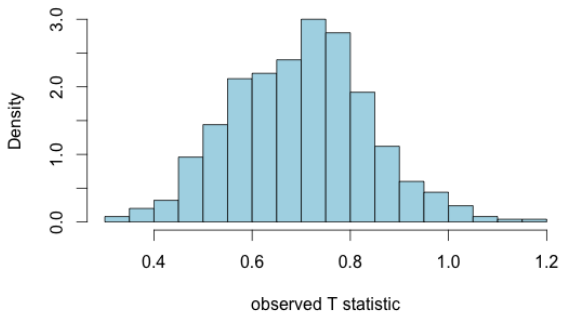


Figure 6.4: The histogram illustrate the distribution of the observed test statistic T.

conservative. In most of the cases, in test 3 (hypothesis H^3), the Type I error tend to be inflated, making the test invalid. Type I error greater than the acceptable range, indicates that the test is too liberal.

However, for the cases where we compare the catalogue entry 1 with 120, catalogue entry 72 with 60, catalogue entry 103 with 114, Type I error is lying in an acceptable range. For test 2 (hypothesis H^2) and test 3, Type I error is either conservative or liberate, with many cases indicating liberal behaviour. Increasing the sample size we do not observe stabilization of error rates or clear trends of Type I error. For some of the investigated examples we can identify an opposite trend between test 1 (hypothesis H^1) and test 2. When Type I error of test 1 becomes conservative, then Type I error of test 2 is more liberate and via versa. Test 1 maintained Type I error rates at nominal levels when we compare the catalogue 72 with 60. Whereas, it is slightly inflated in the case of catalogue 72 with 60.

Two potential reasons why the type I error is not maintained are investigated next. First, the low number of replications that have been performed leads to type I error estimates with big confidence intervals. Second, when generating the null hypothesis for the parametric bootstrap, allele frequency estimates from the data are plugged into the cumulant signature. This is the only difference in the data generation as compared to the simulation of the true simulation model. First, we investigate the influence of bootstrap replications. Table 6.3 lists an example concerning the comparison between catalogue entries 1 and 91. For these examples we have performed simulation analysis increasing the number of bootstrap and simulations procedure to 2000 and 1000 respectively. However, we can not observe any improvement, with the results almost

identical to previous results. Second, Table 6.2 illustrates two examples where we perform a comparison between catalogue entries 1 and 91 and between 1 and 120. For this comparison all the simulated datasets have been set to have allele frequencies equal to 0.5. This implies that the bootstrap samples come from the exact true null distribution. Thus, instead of sample estimates, true allele frequencies are plugged into the cumulant signature. In this example, all the tests maintain the Type I error rate at nominal levels and the plugging in of estimated allele frequencies therefore explains the size violations of the test. Standardized higher order cumulants depend on lower order cumulants (e.g. pairwise D' depends on allele frequencies) which is an intuitive explanation of this behaviour. On the other hand, with increasing sample size the estimation of allele frequencies should become more accurate improving maintenance of the α-level. However, this behaviour could not be confirmed by the simulations performed. A better understanding of this behaviour would require more extensive simulations and theoretical work that is beyond the scope of this thesis.

In summary, the presented testing procedures are therefore valid for known allele frequencies but are conservative or liberal for the sample sizes which were investigated.

catalogue entries		sample size =120	sample size =240	sample size =480	sample size =720
			Type I error rate		
1 and 91	test 1	0.044	0.056	0.048	0.05
	test 2	0.062	0.044	0.046	0.056
	test 3	0.052	0.064	0.054	0.052
1 and 120	test 1	0.05	0.044	0.04	0.048
	test 2	0.054	0.056	0.056	0.04
	test 3	0.044	0.044	0.04	0.046

Table 6.2: Setting the three allele frequencies equal to 0.5

catalogue entries		sample size =120	sample size =240	sample size =480	sample size =720
			Type I error rate		
1 and 91	test 1	0.077	0.069	0.104	0.104
	test 2	0.041	0.032	0.02	0.014
	test 3	0.058	0.053	0.051	0.066

Table 6.3: Increase bootstrap replication to 2000 and simulations to 1000

6.6.3 Power

The probability of rejecting the null hypothesis when it is false is called power of the test and it is defined as follows:

$$power = 1 - prob(no\ reject\ H_0|\ H_0\ false) = 1 - prob(Type\ II\ error)$$

Similar to the type I error of a test, which quantifies the probability of taking a wrong decision (by rejecting the null when is true), power quantifies the probability of making a correct decision (by rejecting the null when is false). In our case, power allows to assess the ability to detect differences between two catalogue entries when given data are truly closer to one of the entries. Typically, 80% or greater power is considered sufficient for actual studies. Otherwise, the test would be likely to be inconclusive for the given data. The simulation procedure to calculate the power, only differs from the procedure to calculate the Type I error, in that we sample under the alternative hypothesis.

In this thesis, two scenarios are used for generating data sets. In the first scenario, we simulate data sets that have been generated by the point κ_H, where:

$$\kappa_H = \frac{\kappa_m + \kappa_2}{2} \tag{6.2}$$

and we assume that the artificial population which is generated by the point k_H (which is represented by LD values) is closer to catalogue entry k_2 (Figure 6.5).

Power is given in Table 6.5 for the same four sample size scenarios that were used to assess type I error. Generally, a test becomes more powerful as sample size increases. Ideally, we would like to have power of 80% for all samples, however, power varies strongly across scenarios. In the majority of the tests seem to be underpowered.

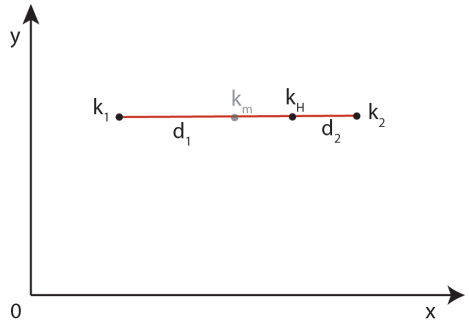


Figure 6.5: The plot illustrates the Euclidean distance of LD values between catalogue entry κ_1 and κ_2 with the κ_H. In this case the $d_1 > d_2$

One example from Table 6.5 is where we test the differences between catalogue entry 1 and 91. The resulting power for the test H^1 when the sample size is 720, indicates that we have a 55.6% chance of drawing the incorrect conclusion by not rejecting H^0 and 44.4% chance of drawing the correct conclusion that the two catalogue entries do not have to the same Euclidean distance to κ_m for the considered sample. This test may result in missing interesting findings that one of the two catalogue entries has a better "story" to tell about the evolutionary history and genealogy of sample population. For this and other examples, sample size needs to be increased to achieve reasonable power.

However, some examples show power of 81.8% and 71.6% for tests H^2 and H^3 respectively, for the comparison between catalogue entries 103 and 128 with sample size equal to 720.

An alternative scenario is to consider the extreme case where we simulated datasets that are generated based on one of the catalogue entries κ_1 or κ_2. Arbitrarily, we select to generate the data based on κ_2. Here, we increase the difference of the alternative to the null, as compared to the previous simulation and again consider the same four sample sizes. We expect to see increased power as compared to the previous examples. Results are listed in Table 6.6. Comparing with the previous simulations, we observe two additional cases having good power. For instance, the comparison between catalogue entry 1 and catalogue 91 results to power 98.6% and 95.8% for the tests H^1 and H^3 respectively. In this example, we observe strong power increase as the sample size increases. This indicates that for samples originating from "pure" catalogue entries, moderate sample sizes of <1000 are sufficient to reach acceptable power. Some of the scenarios do not maintain Type I error. The consequences for the interpretation of power are discussed below.

catalogue entries		sample size =120	sample size =240	sample size =480	sample size =720
			Type I error rate		
	test 1	0.074	0.066	0.09	0.116
1 and 91	test 2	0.052	0.048	0.016	0.01
	test 3	0.06	0.066	0.062	0.062
	test 1	0.032	0.026	0.01	0.012
1 and 120	test 2	0.064	0.064	0.1	0.114
	test 3	0.05	0.046	0.058	0.048
	test 1	0.032	0.034	0.038	0.006
1 and 62	test 2	0.114	0.124	0.136	0.142
	test 3	0.082	0.104	0.098	0.1
	test 1	0.038	0.036	0.036	0.062
72 and 1	test 2	0.082	0.066	0.06	0.062
	test 3	0.072	0.07	0.086	0.076
	test 1	0.056	0.048	0.062	0.062
72 and 60	test 2	0.054	0.068	0.076	0.096
	test 3	0.058	0.052	0.06	0.06
	test 1	0.052	0.056	0.044	0.058
72 and 101	test 2	0.082	0.034	0.054	0.064
	test 3	0.078	0.042	0.056	0.086
	test 1	0.024	0.052	0.032	0.056
103 and 114	test 2	0.054	0.056	0.062	0.058
	test 3	0.032	0.058	0.052	0.04
	test 1	0.04	0.07	0.044	0.038
103 and 146	test 2	0.06	0.038	0.06	0.046
	test 3	0.052	0.068	0.048	0.046
	test 1	0.04	0.028	0.022	0.01
103 and 128	test 2	0.078	0.078	0.11	0.182
	test 3	0.07	0.058	0.074	0.118
	test 1	0.064	0.07	0.08	0.054
62 and 99	test 2	0.054	0.08	0.074	0.09
	test 3	0.054	0.066	0.058	0.08
	test 1	0.046	0.058	0.054	0.09
62 and 131	test 2	0.08	0.078	0.054	0.086
	test 3	0.07	0.062	0.082	0.102
	test 1	0.172	0.352	0.568	0.758
62 and 147	test 2	0.004	0.002	0	0
	test 3	0.092	0.242	0.458	0.61

Table 6.4: Results of Type I error comparing four sample size scenario

catalogue entries		sample size =120	sample size =240	sample size =480	sample size =720
			Power(%)		
	test 1	12.4	17.2	30.8	44.4
1 and 91	test 2	2.8	0.8	0.02	0
	test 3	9.4	9.8	18.2	30.4
	test 1	4.4	2.2	2.2	2.4
1 and 120	test 2	7.4	5.6	6.2	6.2
	test 3	4.8	3.8	4.8	4.2
	test 1	3.6	1	0.8	2
1 and 62	test 2	20.4	23	31	34
	test 3	13.6	15.2	22.8	28.6
	test 1	6	4.8	4.2	4.8
72 and 1	test 2	6.6	11.6	12.6	16
	test 3	7.6	9	10	13.4
	test 1	3.8	4.6	3.4	1.2
72 and 60	test 2	6	8.6	9.4	10
	test 3	5.2	7.8	8	5.6
	test 1	4.2	4.6	3.4	3.4
72 and 101	test 2	7	5.8	6.2	8
	test 3	6	7.2	4.4	6.2
	test 1	9.4	12.4	19.4	28.2
103 and 114	test 2	2.6	1.2	0.4	0
	test 3	6.8	6.2	11.2	17.6
	test 1	4	5.6	6.8	4.2
103 and 146	test 2	4.2	5.2	3.8	6.2
	test 3	4	6	5	6
	test 1	1.8	0.2	0	0
103 and 128	test 2	22.2	33.4	62.8	81.8
	test 3	15.6	23.8	50	71.6
	test 1	7.6	11.6	16.8	19.2
62 and 99	test 2	2.2	2	1.8	1.8
	test 3	5.6	8.8	10	12.2
	test 1	2.8	2.4	2	2.4
62 and 131	test 2	11.2	15.2	18.2	19
	test 3	8.6	10.6	12.6	14.2
	test 1	10.2	18.6	30.8	32.8
62 and 147	test 2	1.6	0.4	0.2	0
	test 3	6.2	10.4	21.6	23.6

Table 6.5: Results of power in approach 1

catalogue entries		sample size =120	sample size =240	sample size =480	sample size =720
			Power(%)		
	test 1	42.2	70.6	91.4	98.6
1 and 91	test 2	0.6	0	0	0
	test 3	29.6	58	86.6	95.8
	test 1	2.4	3.4	2.2	4
1 and 120	test 2	8	4.4	9.6	7.2
	test 3	5	4.4	7.4	6
	test 1	2.8	1.6	0.2	0.6
1 and 62	test 2	20.4	23.8	34.8	42.6
	test 3	15.6	17.4	25.8	32
	test 1	5	4.2	4	6.2
72 and 1	test 2	6.6	8	7.2	6
	test 3	6.6	7.4	6.6	6.2
	test 1	5	4.6	6.6	6.6
72 and 60	test 2	4.8	5.6	6.6	7.8
	test 3	4.6	5.6	7.4	8.2
	test 1	6.2	4.2	7.6	6.4
72 and 101	test 2	3.4	4.6	4.8	8
	test 3	5.4	4.4	6.4	7.4
	test 1	2.6	4.2	2	1.8
103 and 114	test 2	8.8	11.8	11.8	15.4
	test 3	5.8	9	6.4	12
	test 1	5	5.2	7	6.2
103 and 146	test 2	6.2	6.6	6.2	5.4
	test 3	5.2	8.2	7	6
	test 1	0	0.2	0	0
103 and 128	test 2	57.8	88.4	97.8	99.6
	test 3	49.2	80	97.2	99.4
	test 1	27.6	48.8	75	87.6
62 and 99	test 2	0.6	0	0	0
	test 3	18	33.6	60.4	76.8
	test 1	1.4	0.4	8	2
62 and 131	test 2	21.4	32.6	33	45.2
	test 3	14.4	21.8	23.6	33.6
	test 1	8	9.2	12	18.8
62 and 147	test 2	5	2.2	1.4	4
	test 3	7.4	5.4	8	11.2

Table 6.6: Results of power in approach 2

Chapter 7

Data Example

To illustrate the proposed methods, we analyse seven SNPs from chromosome 21 of the HapMap data set. The data set for the SNPs consists of 120 haplotypes. SNPs were selected to represent different biological categories, namely intragenic and intergenic SNPs. The analysis is performed using the methods described in Chapter 5 and Chapter 6.

For a selected SNP, we create a window including neighbouring SNPs, resulting in a window of three SNPs each. Then, we extract the data of haplotypes for each of these windows of SNPs. Subsequently, we compute the haplotype frequency followed by mapping to LD and standardised LD (D'). Subsequently, we tested whether this test statistic is closer to one of two preselected catalogue entries (6). Again we use cumulant signatures, i.e. standardised LD parameters without allele frequencies.

The test performed for the selected set of SNPs, corresponds to the ones for the simulated datasets described in Chapter 6. For this analysis, we have repeated the parametric bootstrap with $B = 1000$ iterations and we have collected bootstrap samples of the test statistic $T = (T_1, T_2, \cdots, T_B)$. Table 7.1 shows the raw p-values of investigated examples and seven sets of SNPs. If the p-value is equal or less than the significant level $\alpha = 0.05$ then the null hypothesis is rejected. Based on the reported p-values we observe that in many cases we have a very strong evidence against the null hypothesis. If the p-value is small (close to 0) then the sample we have obtained is impossible or highly unlikely under the null hypothesis meaning that one of the ancestors is to be preferred in the interpretation.

In the case of comparing the catalogue entries 103 and 128 for the SNP_1 (rs2834508) the empirical distribution of the test statistic T is depicted in in Figure 7.1. The Euclidean distances are d(103, 128) = 0.971 (Table 6.1; 103, 128 denoting catalogue

entry numbers), d_1=d(103, SNP_1) = 0.943 (Table 7.2) and d_2=$d(128, SNP_1)$ = 1 (Table 7.2). The observed test statistics is $T_{obs} = d_2 - d_1 = 0.058$. The resulting p-values from Table 7.1 indicate that the null hypothesis H^3 is rejected. The p-value is small indicating that the data sample is not under the null. This is graphically illustrated in Figure 7.1, where the T_{obs} is located left from the support of the density (the blue point). In contrast, test H^1 and test H^3 are not rejected and cannot identify differences between the catalogue entries 103 and 128. A high p-value close to the upper boundary (0.999) for hypothesis H^2 indicates that the result is not significant and therefore we can not reject the null hypothesis. High p-values can not be used to draw any conclusions and do not provide any evidence in favour of the null hypothesis, in general. That is also the case, when we test one sided hypotheses and we get high p-value, testing the other tail may or may not result in low p-values.

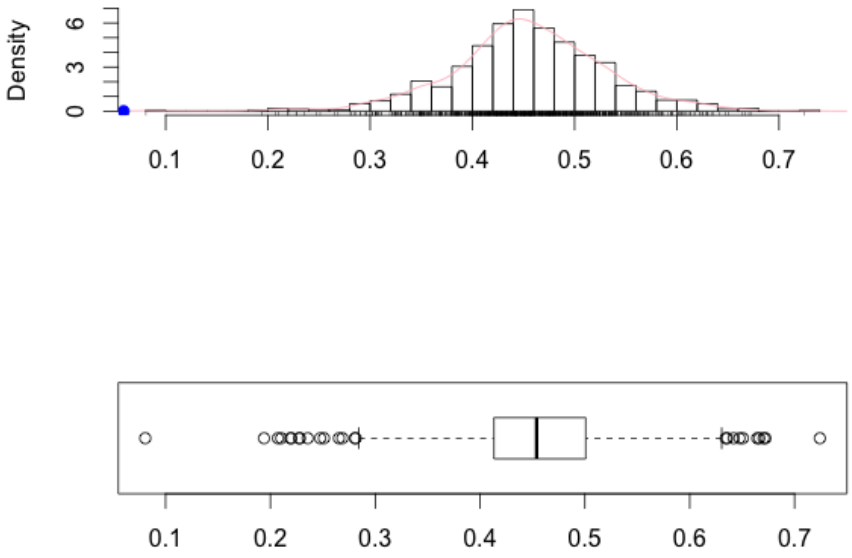


Figure 7.1: The resulting histogram and box plot of the comparison of the two catalogue entries 103 with 128

Based on Table 7.1, we can identify six cases where p-values show strong or moderate evidence against the null hypothesis. For the pair of catalogue entries 72 and 1 which

are tested with SNP1 (rs2834508) and SNP5 (rs11088561), we have found that the p-value of the test H^3 gives a moderate evidence in favour of alternative hypothesis. Hence, we conclude that the Euclidean distance between the two catalogue entries and the SNP differ. Comparing the p-values of test H^1 and H^2 we do not have any evidence to reject the null hypothesis of test H^1, since the test yield a p-value greater than 0.05. Therefore, we conclude that the Euclidean distance between the catalogue entry 72 and SNP_1 ($d_2 = d(72, SNP_1)$) is smaller than the Euclidean distance of catalogue entry 1 and selected SNP_1 ($d_1 = d(1, SNP_1)$). The same conclusions are drawn for the tests H^1, H^2 and H^3 when comparing the aforementioned pair of the catalogue entries with SNP_5. Additionally, for test H^3, the p-values of the catalogue entries 72 with 101 (for SNP_4), 103 with 114 (for SNP_7), 103 with 128 (for SNP_7) and 62 with 99 (for SNP_6) are consider significant.

7.1 Multiple testing correction

In the current analysis, we are testing each hypothesis seven times. The probability to observe at least one significant result, just by chance, is $1 - (1 - 0.05)^7 = 0.30$, assuming independent tests. There are many methods to deal with multiple testing problems. This is often achieved by adjusting α or p-value in order to avoid false positive results. One option is to apply the Bonferroni correction by adjusting the p-values by multiplying the number of SNPs with the "raw" p-values. Another option is to apply the Holm correction, which is similar to Bonferoni but strictly more powerful. In this method the p-values are ranked from smallest to largest, with the first one to be multiplied by the number of SNPs present in the analysis, in this case seven. The second p-value is multiplied by the number of SNPs-1, the third with the number of SNPs-2, etc. All adjusted p-values are considered in that order and all p-value less than the threshold 0.05 are considered significant until the threshold is exceeded for the the first time.

The above procedures are not optimal for a set of tests that are dependent, such as is case in overlapping window analyses. We here take the point of view of an exploratory analysis of the HapMap data and we do not apply any correction in the presented tables.

catalogue entries		rs2834508 SNP1	rs3843783 SNP2	rs2829806 SNP3	rs1057885 SNP4	rs11088561 SNP5	rs1534 SNP6	rs1537118 SNP7
				P value				
	test 1	0.971	0.999	0.999	0.999	0.999	0.998	0.209
1 and 91	test 2	0.029	0.001	0.001	0.001	0.001	0.002	0.791
	test 3	0.058	0.002	0.002	0.002	0.002	0.004	0.418
	test 1	0.999	0.999	0.999	0.999	0.001	0.001	0.001
1 and 120	test 2	0.001	0.001	0.001	0.001	0.999	0.999	0.999
	test 3	0.001	0.001	0.001	0.001	0.001	0.001	0.001
	test 1	0.999	0.999	0.999	0.999	0.999	0.001	0.001
1 and 62	test 2	0.001	0.001	0.001	0.001	0.001	0.999	0.999
	test 3	0.001	0.001	0.001	0.001	0.001	0.001	0.001
	test 1	0.978	0.999	0.999	0.999	0.978	0.971	0.104
72 and 1	test 2	0.022	0.001	0.001	0.001	0.022	0.029	0.896
	test 3	0.044	0.002	0.002	0.002	0.044	0.058	0.208
	test 1	0.167	0.999	0.998	0.996	0.001	0.001	0.001
72 and 60	test 2	0.833	0.001	0.002	0.004	0.999	0.999	0.999
	test 3	0.334	0.002	0.002	0.008	0.001	0.001	0.001
	test 1	0.001	0.327	0.999	0.995	0.999	0.999	0.862
72 and 101	test 2	0.999	0.673	0.001	0.005	0.001	0.001	0.138
	test 3	0.001	0.654	0.002	0.01	0.002	0.002	0.276
	test 1	0.406	0.453	0.936	0.938	0.999	0.999	0.992
103 and 114	test 2	0.594	0.546	0.064	0.062	0.001	0.001	0.008
	test 3	0.812	0.908	0.128	0.124	0.002	0.002	0.016
	test 1	0.001	0.001	0.512	0.511	0.001	0.001	0.383
103 and 146	test 2	0.999	0.999	0.524	0.526	0.001	0.001	0.655
	test 3	0.001	0.001	0.952	0.948	0.002	0.002	0.69
	test 1	0.001	0.001	0.001	0.001	0.999	0.999	0.978
103 and 128	test 2	0.999	1	0.999	0.999	0.001	0.001	0.021
	test 3	0.001	0.001	0.001	0.001	0.002	0.002	0.042
	test 1	0.999	0.999	0.999	0.999	0.047	0.018	0.936
62 and 99	test 2	0.001	0.001	0.001	0.001	0.953	0.982	0.064
	test 3	0.002	0.002	0.002	0.002	0.094	0.036	0.128
	test 1	0.999	0.999	0.999	0.999	0.001	0.001	0.001
62 and 131	test 2	0.001	0.001	0.001	0.001	0.999	0.999	0.999
	test 3	0.002	0.002	0.002	0.002	0.001	0.001	0.001
	test 1	0.999	0.999	0.999	0.999	0.001	0.001	0.501
62 and 147	test 2	0.001	0.001	0.001	0.001	0.999	0.999	0.499
	test 3	0.002	0.002	0.002	0.002	0.001	0.001	0.998

Table 7.1: Results of p-values

catalogue entries	rs2834508 SNP1	rs3843783 SNP2	rs2829806 SNP3	rs1057885 SNP4	rs11088561 SNP5	rs1534 SNP6	rs1537118 SNP7
				Euclidean Distance			
1	1.373	1.000	0.000	0.000	1.373	1.414	1.732
91	2.292	2.062	2.291	2.291	2.453	2.500	2.291
120	1.461	1.118	0.500	0.500	1.066	1.118	1.500
62	1.002	0.000	1.000	1.000	1.663	1.732	1.414
72	1.277	1.414	1.528	1.528	1.246	1.291	1.155
60	1.255	1.269	1.394	1.394	1.346	1.394	1.269
101	1.795	1.810	1.810	1.810	1.471	1.509	1.509
103	0.943	0.972	0.972	0.972	0.705	0.782	0.782
114	2.062	2.062	2.291	2.291	2.465	2.500	2.291
146	0.759	0.782	0.972	0.972	0.900	0.972	0.782
128	1.002	0.000	1.000	1.000	1.663	1.732	1.414
99	2.266	2.500	2.291	2.291	2.063	2.062	2.291
131	1.415	1.000	1.414	1.414	1.329	1.414	1.000
147	0.794	0.577	0.816	0.816	1.086	1.155	1.000

Table 7.2: Results of Euclidean distance between catalogue entries and SNPs

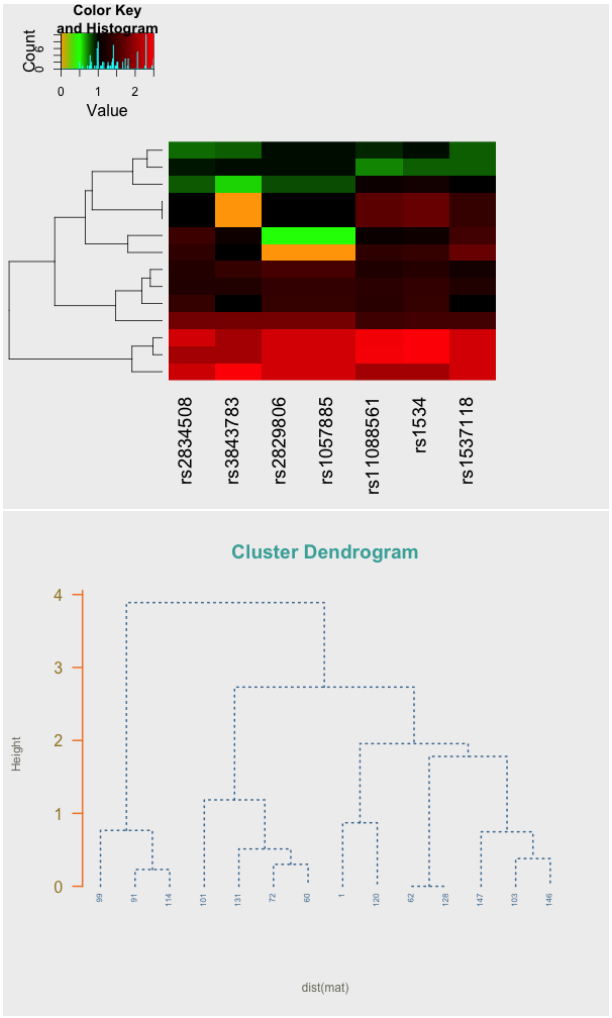


Figure 7.2: The heat map and the dendrogram plot of the Euclidean distance for the seven SNPs.

Chapter 8

Discussion

In order to infer details of evolutionary processes as well as to decipher the genealogy of given data, linkage disequilibrium can be used. Genealogy is here seen as the description of a sequence of genetic events generating the data. Evolutionary forces can be divided into neutral (mutation, drift, recombination, migration etc.) and non-neutral forces (selection). Arguably, in an exponentially growing population such as humans, a model that only account for mutation, recombination, and drift, while ignoring fixation events, can explain the major part of genetic variation in the population. Comparison of the empirical patterns of linkage disequilibrium to the ones expected under such a model, can provide insight about the genetic diversity and the genealogy of the sample. The aim of this thesis is to reveal the genealogy of given data by comparing a theoretical population to observed data. As an example, data acquired from the HapMap project was used. This is achieved both by visualization in terms of heatmap plots that reveal closeness of SNP data to possible genealogies (the catalogue) and by developing a hypotheses testing procedures based on the parametric bootstrap.

All methods are based on haplotypes which exploit more information from the data as compared to single SNP analysis. However, haplotype based methods can suffer from problems of their own, such as dealing with rare haplotypes. When a haplotype is rare, implying that also its allele frequencies are low, hence, the LD values will be strongly influenced by small changes in allele frequencies. Most of the testing scenarios considered in this thesis showed limited maintenance of Type I error. In a limited investigation it was shown that Type I error is well maintained when allele frequencies are known. This indicates that the strong dependency of standardised LD on allele frequencies is the most likely reason for this behaviour. More simulations would be needed to extract exact reasons and possible remedies for this problem. Due to time constraints they were outside the scope of this thesis. A consequence of inflated type I

error is that power and p-values cannot be interpreted strictly for the cases where type I error was not maintained. We have provided nominal p-values in this thesis and their interpretation must be seen in this context. They can still be used as an exploratory, descriptive measure. Furthermore, when we insert known allele frequencies instead of estimating them, we maintain the the α-level accurately.

In Chapter 5, we visualised the LD patterns of the catalogue entries using a selected set of SNPs. The results of the heatmap plots, indicate that there are specific catalogue entries which can be excluded as an explanation for the given SNP data. For example, catalogue entry 99 cannot explain the genealogy of the four set of 100 SNPs, indicating few recombination events have taken place into the sample. However, integrating LD patterns based on heatmap plots does not allow to make any inference on their genealogy. These plots help to get a quick overview of the behaviours of linkage disequilibrium across a whole genetic regions.

In Chapter 6, we proposed three hypotheses to inferential ends. Apart from the problems of type I maintenance above, simulations indicate that powerful inference needs big sample sizes in the thousands of individuals. Big sample sizes are also preferable to meet the assumption that no haplotypes are lost due to genetic drift. The sampling process itself can be seen as creating a new population which has just gone through the bottleneck of sampling. For the HapMap sample consisting of 60 independent individuals certainly that effect is extremely pronounced and results presented here must be seen as merely indicative of certain genealogies. Through sampling, only few haplotypes may be present in the data implying that D' values may be biased upwards, indicating that only few historical recombination events appear between two markers. Also this problem can be remedied by increasing sample size.

In Chapter 7 we apply the proposed tests to real datasets, derived from the HapMap project. It is not reasonable to test data against any given catalogue pair. If the data is far away from both entries that are to be compared, a significant result is meaningless in practice. The inferential procedure should therefore always be accompanied by the descriptive methods discussed above. Characterising the pairs of the catalogue that can be used for meaningful comparisons, is another interesting problem for future research. Under the caveats mentioned above, among all the catalogue entries, three (72, 103 and 99) gave promising results in the dataset and are more likely to explain their genealogy as compared to their counterparts.

For computational reasons, the analyses were limited to three loci at a time. All the computations of this thesis were conducted on a UNIX server with 64 CPUs. Supercomputers and/or cluster/grids of servers can be used to apply those computations for

more than 3 loci. However, it seems difficult to go beyond 5 loci, as the computation of the catalogue is exponential in the square of the number of loci.

In conclusion, we provide both visualization and inferential techniques to analyse the genealogy of haplotype or genotype data. The visualization allows to quickly assess whole regions simultaneously. Testing allows to distinguish between two scenarios but the tests only maintain type I error for certain comparisons in their current form. Limitations are computational cost, some strong assumptions, and the needed for relatively big samples.

Appendix A

.1 Re-parametrization

If we denote as N to be the number loci of a DNA segment, then we consider that $A = \{A_l\}$, $A_l \in \{0, 1\}$ for every $l \in 1, 2, ..., N$, to be a random random variable for l markers. Then we denote as $\beta_s = \{b \in < 0, 1 > | A_l \in s \iff b_l = 1\}$ a set of random variables A. For example, if N=3 and $A = \{A_1, A_2, A_3\}$ then all possible sets S of A is:

$$S = \{A_1\}, \{A_2\}, \{A_3\}, \{A_1, A_2\}, \{A_1, A_3\}, \{A_2, A_3\}, \{A_1, A_2, A_3\}\}$$

Then, for instance, $\beta_{\{A_1,A_2\}}$ will contain two haplotypes with allele 1 at loci 1 and 2. Therefore $\beta_{\{A_1,A_2\}} = \{\{1, 1, 0\}, \{1, 1, 1\}\}$.
Let's consider $P(A)$ to be a family of sets of all possible partitions of sets $s \in S$ of A into non empty subsets (blocks). Therefore, for $\tau \in P(A)$ each $\beta \in \tau$ is a block. Then the joint cumulant of set of random variables A is:

$$\delta_s = \sum_{\tau \in P(A)} (-1)^{|\tau|-1}(|\ \tau\ | -1)! \prod_{\beta \in \tau} \mathbb{E}\Big(\prod_{A \in \beta} A \Big) = D$$

where,

$$\prod_{\beta \in \tau} \mathbb{E}\Big(\prod_{A \in \beta} A \Big) = \sum_{B \in \beta_s} p_B = \eta_s$$

refers to be the joint expectation of random variable s, for $s \in S$ of haplotype frequency $p_B = p_{ijk}$ of B haplotype. Equivalently, $\eta_s = (\eta_1, \eta_2, \eta_3, \eta_{12}, \eta_{12}, \eta_{23}, \eta_{123}) = (p_{2\cdot\cdot}, p_{\cdot 2\cdot}, p_{\cdot\cdot 2}, p_{22\cdot}, p_{2\cdot 2}, p_{\cdot 22}, p_{222})$. In case of N=3 and $A = \{A_1, A_2, A_3\}$ then all possible partitions of A is :

$$P(A) = \{\{A_1, A_2, A_3\}, \{\{A_1, A_2\}, \{A_3\}\}, \{\{A_1, A_3\}, \{A_2\}\}, \{\{A_2, A_3\}, \{A_1\}\}, \{\{A_1\}, \{A_2\}, \{A_3\}\}\}$$

Then the joint cumulant is:

$$\delta_s = (-1)^{1-1}(1-1)!p_{222} + (-1)^{2-1}(2-1)!p_{..2}p_{.22} + (-1)^{2-1}(2-1)!p_{.2.}p_{22.}$$

$$+(-1)^{2-1}(2-1)!p_{2..}p_{2.2} + (-1)^{3-1}(3-1)!p_{2..}p_{.2.}p_{..2}$$

$$= p_{222} - p_{..2}p_{.22} - p_{2..}p_{2.2} - p_{2..}p_{2.2} - p_{2..}p_{.2.}p_{..2}$$

.2 Standardised LD for arbitrary number of loci

To generalise the concept of pairwise LD for arbitrary number of loci, re-parametrization method of LD are needed. The joint cumulants δ_A can been written:

$$\delta_A = \eta_A - \sum_{\tau \in P(A)} (-1)^{|\tau|}(|\ \tau\ | -1)! \prod_{\beta \in \tau} \eta_\beta$$

$$= \eta_A - \sum_{\tau \in P(A)} R_\delta(\tau) = \eta_A - R_\delta$$

where. $R(\tau)$ depends on loci $\beta \in \tau$ with $|\ \beta\ | < N$. These rest terms $R_\delta(\tau)$ are considered fixed and bounds for δ_A are to be determined completely analogous to the two locus case based on η_A. First, η_A is upper bound by all lower-order η_s and lower bound by 0. That is:
$\eta_A \leq U_1(A) = \min\{\eta_s \mid s \in S \backslash A\}$
$\eta_A \geq L_1(A) = 0$

Second, further constraints are imposed by the relationship between η_A and lower order haplotype frequencies η_s. It is straightforward to see that η_s can be restated as:

$$\eta_s = p_s + \sum_{t \in S, s \subset t} (-1)^{|t|-|s|-1}\eta_t$$

Here, p_{B_s} is the frequency for haplotype $B_s = \{b \in \{0,1\}^N | A_l \in s \iff b_l = 1\}$,

the haplotype with N loci with 1-alleles at loci s and 0-alleles elsewhere. Note, that all the sums above include η_A.Solving for η_A gives us:

$$\eta_A = (-1)^{|A|-|s|-1}\{\eta_s - p_{B_s} - \sum_{t\in S, s\subset t}(-1)^{|t|-|s|-1}\}\eta_t$$

$$= (-1)^{|A|-|s|}\{p_{B_s} - \sum_{t\in S, s\subset t}(-1)^{|t|-|s|}\eta_t\}$$

$$= \sigma_s(p_{B_s} - R_s),$$

where $\sigma = (-1)^{|A|-|s|}$ and $R_s = \sum_{t\in S, s\subset t}(-1)^{|t|-|s|}\eta_t$. Each η_s therefore contributes an upper and lower bound toη_A by choosing $p_{B_s} = 0$ or $p_{B_s} = 1$:

$$\eta_s \leq U_s =$$

$$\eta_s \leq U_s = \begin{cases} max(1 - R_s, 0) & : if\ \sigma \geq 0 \\ min(R_s, 0) & : if\ \sigma < 0 \end{cases}$$

$$\eta_s \geq L_s = \begin{cases} max(-R_s, 0) & : if\ \sigma \geq 0 \\ min(1 - R_s, 0) & : if\ \sigma < 0 \end{cases}$$

With $U_2(A) = min\{U_s | s \in S\backslash\{A\}\}$ and $L_2(A) = max\{L_s | s \in S\backslash\{A\}\}$, we get
$\eta_A^{max} = min\{U_1(A), U_2(A)\}$,
$\eta_A^{min} = min\{L_1(A), L_2(A)\}$.
Then η_A^{max} and η_A^{min} can be used as above to standardised δ_A

Appendix B

Load necessary packages

```
library(partitions)
library(sets)
library(gtools)
library(devtools)
library(Rcpp)
library(GeneticsLD)
library(pander)
library(partitions)
library(sets)
library(ggplot2)
library(gplots)
```

The package GeneticsLD was created by Dr Stefan Boehringer and was necessary for the completion of the project.

Helpful functions to load and save lists

```
save.list <-function(x,file){
  save(x,file=file)
}

load.list <- function(file){
  load(file)
  return(x)
}
```

We use the two functions ord2bin and bin2ord which convert a number into binary representation and vice versa.In the algorithm we index haplotypes from 0 to 3. The binary representation then gives the alleles corresponding to this haplotype.

```
bin2ord = function(b, base = 2)
  as.vector(b %*% sapply(1:length(b), function(i)2^(i-1)))

ord2bin = function(o, digits = 2, base = 2)
  sapply(1:digits, function(i){(o %/% 2^(i-1)) %% 2})
```

In order to calculate the mutation the below fuctions have been used:

```
Mutation <- function(htfs, pos.hap, v = rep(0,3)){
  #htfs= haplotype frequency
  # pos.hap= position of locus

  v[pos.hap]=1 - v[pos.hap]
  htfs[bin2ord(v)+1]=1
  htfs
}

mutation.name <- function(htfs, pos.hap,v = rep(0,3)){
```

```
  m<-list(Mutation(htfs[[1]], pos.hap,v))
  n<- if (is.null(names(htfs)))
    sprintf("M(%d)", pos.hap) else
      sprintf("%s->M(%d)", names(htfs)[1], pos.hap)
  names(m) <- n
  m
}

mut.name.multi <- function(htfs,pos,v = rep(0,3)){

  d <- lapply(1:length(htfs), function(i){
    mutation.name(htfs[i],pos,v)

  })
  d <- unlist(d,recursive = F)
  d
}
```

For the recombination event the below functions have been used:

```
partition <- function(k,v,pos){
  #htBin1 = ord2bin(ht1, L)
  #htBin2 = ord2bin(ht2, L)

  htBinNew1 = c(k[1:pos], v[-(1:pos)]);
  htBinNew2 = c(v[1:pos], k[-(1:pos)]);
  l<-list(htBinNew1,htBinNew2)
  return(l)


}

recomb <- function(htfs,k,v,pos){
  if(htfs[bin2ord(k)+1]==0 || htfs[bin2ord(v)+1]==0 ){

    return(htfs)
  }
  rec<-partition(k,v,pos)
  htfs[bin2ord((rec[[1]]))+1]=1
  htfs[bin2ord((rec[[2]]))+1]=1
  htfs

}

rec.name <- function(htfs,k,v,pos){
  m<-list(recomb(htfs[[1]], k,v,pos))
  n<- if (is.null(names(htfs)))
```

```
    sprintf("R(%d|%d,%d)",bin2ord(k),bin2ord(v),pos) else
      sprintf("%s->R(%d|%d,%d)", names(htfs)
[1],bin2ord(k),bin2ord(v),pos)

  names(m) <- n
  m
}

all.rec <- function(htfs,N,pos){
  hf<-which(htfs[[1]]>0)

  a=set_combn(as.set(1:length(hf)),2)  # set with all possiple
partitions
  mat <- list() #storage the results
  h<-c()
  for(i in 1:length(a)){
    set <- unlist(as.list(as.list(a)[[i]]))
    hap <- hf[set]
    r <-rec.name(htfs,ord2bin(hap[1]-1,N),ord2bin(hap[2]-1,N),pos)
    mat[[i]] <-r

  }
  mat <- unlist(mat,recursive = F)
  return(mat)

}
```

Create a table which take all possible sets of all recombinations and check if they are identical or not.

```
table.rec <- function(htfs,N,pos){
  rec <- all.rec(list(htfs),N,pos)
  k <- length(rec)
  if(k==1){
    return(data.frame(Recomb.1=1,Recomb.2=1,Identical=T)[-1,])
  }
  a <- set_combn(as.set(1:k),2)
  mat <- matrix(NA,ncol=3,nrow=length(a))

  for(i in 1:length(a)){
    set <- unlist(as.list(as.list(a)[[i]]))
    hap <- rec[set]
    id  <- identical(hap[[1]],hap[[2]])

    mat[i,1] <- names(hap)[1]
    mat[i,2] <- names(hap)[2]
    mat[i,3] <- id
```

```
  }
  mat <- as.data.frame(mat)
  colnames(mat)<- c("Recomb.1","Recomb.2","Identical")
  return(mat)
}
```

Create a function which returns all possiple recombinations and check which of them return the same haplotype.

```
CheckF <- function(htfs,N,pos){
  Table <- table.rec(htfs,N,pos)
  lev <- unique(c(levels(factor(Table[,1])),levels(factor(Table[,2]))))
  Re <- all.rec(list(htfs),N,pos)
  p <- list()
  if(nrow(Table)==0){
    p<-list(htfs)
    names(p)<-"()"
    return(p)
  }
  for(i in 1:length(lev)){
    sub.T <- subset(Table,Table[,1]==lev[i]|Table[,2]==lev[i])
    rowF <- which(sub.T[,3]==F)
    rowT <- which(sub.T[,3]==T)
    Ftbls <- sub.T[rowF,]
    Ttbls <- sub.T[rowT,]
    r <- list(Re[[i]])

    if( length(rowT)==0){

      p[[length(p)+1]]<-(Re[i])

    } else if(length(rowT)>0){
      select <- unique(c(levels(factor(Ttbls[,
1])),levels(factor(Ttbls[,2]))))
      select <- select[order(select)]

      if(select[1]==lev[i]){
        names(r) <- sprintf("(%s)",(paste(select,collapse=",")))
        p[[length(p)+1]] <-r

      }
    }
  }
  p <- unlist(p,recursive = F)
  return(p)
```

```
}

check.name <- function(htfs,N,pos){
  m<-(CheckF(htfs[[1]],N,pos))
  for(i in 1:length(m)){
    names(m)[i]<- if (is.null(names(htfs)))
      names(m)[i] else
        sprintf("%s->%s", names(htfs),names(m)[i])
    #print(names(m)[i])
  }
  return(m)
}

check.name.multi <- function(htfs,N,pos){

  d <- lapply(1:length(htfs), function(i){
    check.name(htfs[i],N,pos)

  })
  d <- unlist(d,recursive = F)
  d
}
```

The names of all recombinations which result to the same haplotypes are saved in a list.

```
deduplicate <- function(htfs){
  hfsMat <- t(sapply(htfs, identity))
  hfsMatUnique <- unique(as.data.frame(hfsMat))

  p <- list()

  for(i in 1:nrow(hfsMatUnique)){
    selM <- apply(hfsMat, 1, function(r)(r == hfsMatUnique[i,]))
    sel <- apply(selM, 2, all)
    index <- which(sel)

    r <-list(htfs[[index[1]]])
    names(r) <- sprintf("(%s)",(paste(names(htfs)
[index],collapse=",")))
    p[[length(p)+1]] <-r
  }
  p <- unlist(p,recursive = F)
  return(p)

}
```

The catalogue is created based on the aforementioned code and taking into account the possible events that can been happened between three loci:

```
tot.catalog <- function(v = rep(0,3)){
  N <- length(v)  # number of loci
  L <- N-1        #
  htfs=rep(0,2^N)
  htfs[bin2ord(v)+1]=1; htfsl=list(htfs);
  names(htfsl) <-sprintf("ht(%s)", paste(v,collapse=""))

  #vector <- c("M","NA","M","NA","M","NA","M","R"...)
  vector<-c(rep(c("M","NA"),L),"M","R")
  seqs <- c(0:(2^L-1))
  recombs <- ord2bin(0:(2^L-1),L)
  bigvector <- t(replicate(length(seqs),vector))

  for(i in 1:length(seqs)){
    k <- which(recombs[i,]==1)
    bigvector[i,2*k] <- as.character("R")
  }

  bigvector[bigvector=="NA"] <- NA

  # all possible mutations and recombinations events
  events <- c()
  for(i in 1:nrow(bigvector)){
    events[[i]] <- as.vector(na.omit(bigvector[i,]))
  }
  #permutations
  loci <- 1:N
  per.loci <-permutations(n=N,r=N,v=loci) # all possible permutations
of N numbers
  row=nrow(per.loci)

  catalog<-list()
  total.catalog <-list()
  final.catalog <- list

  for (e in 1:length(events)){
    check <- events[[e]]=="M"

    for(r in 1:(row)){
      results <- list()
      rec_loci <- list()
      count=1
      results[[1]] <- mut.name.multi(htfsl,per.loci[r,][count],v)  #r
```

```
      for(j in 2:length(check)){
        if(check[j]==T){
          count=count+1
          locus <- per.loci[r,][count] #r
          results[[j]] <- (mut.name.multi(results[[j-1]],locus,v))

        }else{
          for(s in 1:L){
            rec_loci[[s]]<-check.name.multi(results[[j-1]],N,s) #bale N
          }
          results[[j]] <- deduplicate(unlist(rec_loci,recursive=F))
        }
      }
      results<-unlist(results ,recursive=F)
      results <- deduplicate(results)

      final <- list()
      iter=1
      results1 <-list()
      results1[[iter]]<-results

      repeat{
        iter=iter+1
        for(s in 1:L){
          final[[s]]<-check.name.multi(results1[[iter-1]],N,s) # N
        }
        results1[[iter]] <-deduplicate(unlist(final,recursive=F))
        if (length(results1[[iter]])==length(results1[[iter-1]])){
          break
        }
      }
      catalog[[r]]<-deduplicate(unlist(results1,recursive=F))
    }
    total.catalog[[e]] <- deduplicate(unlist(catalog,recursive=F))
  }
  final.catalog<-deduplicate(unlist(total.catalog,recursive=F))
  return(final.catalog)
}
```

Using the above R codes we create the catalogue that has been used to this thesis. A part of the catalogue is illustrated below:

```
## $B0
## [1] 1 0 0 0 0 0 0 0
##
## $B1
```

```
## [1] 0 1 0 0 0 0 0 0
##
## $`[B0->M1,B1->M1]`
## [1] 1 1 0 0 0 0 0 0
##
## $B2
## [1] 0 0 1 0 0 0 0 0
##
## $`[B0->M2,B2->M2]`
## [1] 1 0 1 0 0 0 0 0
##
## $`[B0->[M1->M2,M2->M1],B1->M1->M2,B2->M2->M1]`
## [1] 1 1 1 0 0 0 0 0
##
## $B3
## [1] 0 0 0 1 0 0 0 0
##
## $`[B1->M2,B3->M2]`
## [1] 0 1 0 1 0 0 0 0
##
## $`[B0->M1->M2,B1->[M1->M2,M2->M1],B3->M2->M1]`
## [1] 1 1 0 1 0 0 0 0
##
## $`[B2->M1,B3->M1]`
## [1] 0 0 1 1 0 0 0 0
##
## $`[B0->M2->M1,B2->[M1->M2,M2->M1],B3->M1->M2]`
## [1] 1 0 1 1 0 0 0 0
##
## $`[B1->M2->M1,B2->M1->M2,B3->[M1->M2,M2->M1]]`
## [1] 0 1 1 1 0 0 0 0
##
## $`[B0->[M1->M2->R1,M2->M1->R1],B1->[M1->M2->R1,M2->M1->R1],B2->[M1-
>M2->R1,M2->M1->R1],B3->[M1->M2->R1,M2->M1->R1]]`
## [1] 1 1 1 1 0 0 0 0
##
## $B4
## [1] 0 0 0 0 1 0 0 0
##
## $`[B0->M3,B4->M3]`
## [1] 1 0 0 0 1 0 0 0
##
## $`[B0->[M1->M3,M3->M1],B1->M1->M3,B4->M3->M1]`
## [1] 1 1 0 0 1 0 0 0
##
```

```
## $`[B0->[M3->M2,M2->M3],B2->M2->M3,B4->M3->M2]`
## [1] 1 0 1 0 1 0 0 0
##
## $`[B0->[M2->[M3->M1,M1->M3],[M1->[M2->M3,M3->M2],M3->[M1->M2,M2-
>M1]]],B1->M1->[M2->M3,M3->M2],B2->M2->[M3->M1,M1->M3],B4->M3->[M1-
>M2,M2->M1]]`
## [1] 1 1 1 0 1 0 0 0
##
## $`[B0->[M3->M1->M2,M1->[M2->M3,M3->M2]],B1->[M2->M1->M3,M1->[M2-
>M3,M3->M2]],B3->M2->M1->M3,B4->M3->M1->M2]`
## [1] 1 1 0 1 1 0 0 0
##
## $`[B0->[M3->M2->M1,M2->[M3->M1,M1->M3]],B2->[M1->M2->M3,M2->[M3-
>M1,M1->M3]],B3->M1->M2->M3,B4->M3->M2->M1]`
## [1] 1 0 1 1 1 0 0 0
##
## $...
## [1] 1 1 1 1 1 0 0 0
##
## $B5
## [1] 0 0 0 0 0 1 0 0
##
## $`[B1->M3,B5->M3]`
## [1] 0 1 0 0 0 1 0 0
##
## $`[B0->M1->M3,B1->[M1->M3,M3->M1],B5->M3->M1]`
## [1] 1 1 0 0 0 1 0 0
##
## $`[B0->[M2->M1->M3,M1->[M2->M3,M3->M2]],B1->[M3->M1->M2,M1->[M2-
>M3,M3->M2]],B2->M2->M1->M3,B5->M3->M1->M2]`
## [1] 1 1 1 0 0 1 0 0
##
## $`[B1->[M3->M2,M2->M3],B3->M2->M3,B5->M3->M2]`
## [1] 0 1 0 1 0 1 0 0
##
## $`[B0->M1->[M2->M3,M3->M2],B1->[M2->[M3->M1,M1->M3],[M1->[M2->M3,M3-
>M2],M3->[M1->M2,M2->M1]]],B3->M2->[M3->M1,M1->M3],B5->M3->[M1->M2,M2-
>M1]]`
## [1] 1 1 0 1 0 1 0 0
##
## $`[B1->[M3->M2->M1,M2->[M3->M1,M1->M3]],B2->M1->M2->M3,B3->[M1->M2-
>M3,M2->[M3->M1,M1->M3]],B5->M3->M2->M1]`
## [1] 0 1 1 1 0 1 0 0
##
## $...
## [1] 1 1 1 1 0 1 0 0
```

```
##
## $`[B4->M1,B5->M1]`
## [1] 0 0 0 0 1 1 0 0
##
## $`[B0->M3->M1,B4->[M1->M3,M3->M1],B5->M1->M3]`
## [1] 1 0 0 0 1 1 0 0
##
## $`[B1->M3->M1,B4->M1->M3,B5->[M1->M3,M3->M1]]`
## [1] 0 1 0 0 1 1 0 0
##
## $`[B0->[M1->M3->R1,M3->M1->R1],B1->[M1->M3->R1,M3->M1->R1],B4->[M1-
>M3->R1,M3->M1->R1],B5->[M1->M3->R1,M3->M1->R1]]`
## [1] 1 1 0 0 1 1 0 0
##
## $`[B0->[M2->M3->M1,M3->[M1->M2,M2->M1]],B2->M2->M3->M1,B4->[M1->M3-
>M2,M3->[M1->M2,M2->M1]],B5->M1->M3->M2]`
## [1] 1 0 1 0 1 1 0 0
##
## $...
## [1] 1 1 1 0 1 1 0 0
##
## $`[B1->[M2->M3->M1,M3->[M1->M2,M2->M1]],B3->M2->M3->M1,B4->M1->M3-
>M2,B5->[M1->M3->M2,M3->[M1->M2,M2->M1]]]`
## [1] 0 1 0 1 1 1 0 0
##
## $...
## [1] 1 1 0 1 1 1 0 0
##
## $`[B0->[M3->[M1->M2->R1,M2->M1->R1],M2->[M3->M1->R1,M1->M3->R1]],B2-
>[M1->M2->M3->R1,M2->[M3->M1->R1,M1->M3->R1]],B3->M1->M2->M3->R1,B4-
>[M1->M3->M2->R1,M3->[M1->M2->R1,M2->M1->R1]],B5->M1->M3->M2->R1]`
## [1] 1 0 1 1 1 1 0 0
##
## $`[B1->[M3->[M1->M2->R1,M2->M1->R1],M2->[M3->M1->R1,M1->M3->R1]],B2-
>M1->M2->M3->R1,B3->[M1->M2->M3->R1,M2->[M3->M1->R1,M1->M3->R1]],B4-
>M1->M3->M2->R1,B5->[M1->M3->M2->R1,M3->[M1->M2->R1,M2->M1->R1]]]`
## [1] 0 1 1 1 1 1 0 0
##
## $...
## [1] 1 1 1 1 1 1 0 0
##
## $B6
## [1] 0 0 0 0 0 0 1 0
##
## $`[B2->M3,B6->M3]`
## [1] 0 0 1 0 0 0 1 0
```

```
##
## $`[B0->M2->M3,B2->[M3->M2,M2->M3],B6->M3->M2]`
## [1] 1 0 1 0 0 0 1 0
##
## $`[B0->[M1->M2->M3,M2->[M3->M1,M1->M3]],B1->M1->M2->M3,B2->[M3->M2-
>M1,M2->[M3->M1,M1->M3]],B6->M3->M2->M1]`
## [1] 1 1 1 0 0 0 1 0
##
## $`[B2->[M1->M3,M3->M1],B3->M1->M3,B6->M3->M1]`
## [1] 0 0 1 1 0 0 1 0
##
## $`[B0->M2->[M3->M1,M1->M3],B2->[M2->[M3->M1,M1->M3],[M1->[M2->M3,M3-
>M2],M3->[M1->M2,M2->M1]]],B3->M1->[M2->M3,M3->M2],B6->M3->[M1->M2,M2-
>M1]]`
## [1] 1 0 1 1 0 0 1 0
##
## $`[B1->M2->M1->M3,B2->[M3->M1->M2,M1->[M2->M3,M3->M2]],B3->[M2->M1-
>M3,M1->[M2->M3,M3->M2]],B6->M3->M1->M2]`
## [1] 0 1 1 1 0 0 1 0
##
## $...
## [1] 1 1 1 1 0 0 1 0
##
## $`[B4->M2,B6->M2]`
## [1] 0 0 0 0 1 0 1 0
##
## $`[B0->M3->M2,B4->[M3->M2,M2->M3],B6->M2->M3]`
## [1] 1 0 0 0 1 0 1 0
##
## $`[B0->[M1->M3->M2,M3->[M1->M2,M2->M1]],B1->M1->M3->M2,B4->[M2->M3-
>M1,M3->[M1->M2,M2->M1]],B6->M2->M3->M1]`
## [1] 1 1 0 0 1 0 1 0
##
## $`[B2->M3->M2,B4->M2->M3,B6->[M3->M2,M2->M3]]`
## [1] 0 0 1 0 1 0 1 0
##
## $`[B0->[M3->M2->R2,M2->M3->R2],B2->[M3->M2->R2,M2->M3->R2],B4->[M3-
>M2->R2,M2->M3->R2],B6->[M3->M2->R2,M2->M3->R2]]`
## [1] 1 0 1 0 1 0 1 0
##
## $...
## [1] 1 1 1 0 1 0 1 0
##
## $`[B2->[M1->M3->M2,M3->[M1->M2,M2->M1]],B3->M1->M3->M2,B4->M2->M3-
>M1,B6->[M2->M3->M1,M3->[M1->M2,M2->M1]]]`
## [1] 0 0 1 1 1 0 1 0
```

```
##
## $...
## [1] 1 0 1 1 1 0 1 0
##
## $...
## [1] 1 1 1 1 1 0 1 0
##
## $`[B0->[M1->[M2->M3->R2,M3->M2->R2],M2->[M3->M1->R2,M1->M3->R2]],B1-
>[M3->M1->M2->R2,M1->[M2->M3->R2,M3->M2->R2]],B2->[M3->M2->M1->R2,M2-
>[M3->M1->R2,M1->M3->R2]],B5->M3->M1->M2->R2,B6->M3->M2->M1->R2]`
## [1] 1 1 1 0 0 1 1 0
##
## $`[B1->[M3->M2->M1->R2,M2->[M3->M1->R2,M1->M3->R2]],B2->[M3->M1->M2-
>R2,M1->[M2->M3->R2,M3->M2->R2]],B3->[M1->[M2->M3->R2,M3->M2->R2],M2-
>[M3->M1->R2,M1->M3->R2]],B5->M3->M2->M1->R2,B6->M3->M1->M2->R2]`
## [1] 0 1 1 1 0 1 1 0
##
## $...
## [1] 1 1 1 1 0 1 1 0
##
## $`[B4->[M1->M2,M2->M1],B5->M1->M2,B6->M2->M1]`
## [1] 0 0 0 0 1 1 1 0
##
## $`[B0->M3->[M1->M2,M2->M1],B4->[M2->[M3->M1,M1->M3],[M1->[M2->M3,M3-
>M2],M3->[M1->M2,M2->M1]]],B5->M1->[M2->M3,M3->M2],B6->M2->[M3->M1,M1-
>M3]]`
## [1] 1 0 0 0 1 1 1 0
##
## $`[B1->M3->M1->M2,B4->[M2->M1->M3,M1->[M2->M3,M3->M2]],B5->[M3->M1-
>M2,M1->[M2->M3,M3->M2]],B6->M2->M1->M3]`
## [1] 0 1 0 0 1 1 1 0
##
## $...
## [1] 1 1 0 0 1 1 1 0
##
## $`[B2->M3->M2->M1,B4->[M1->M2->M3,M2->[M3->M1,M1->M3]],B5->M1->M2-
>M3,B6->[M3->M2->M1,M2->[M3->M1,M1->M3]]]`
## [1] 0 0 1 0 1 1 1 0
##
## $...
## [1] 1 0 1 0 1 1 1 0
##
## $`[B1->M3->M1->M2->R2,B2->M3->M2->M1->R2,B4->[M1->[M2->M3->R2,M3-
>M2->R2],M2->[M3->M1->R2,M1->M3->R2]],B5->[M3->M1->M2->R2,M1->[M2->M3-
>R2,M3->M2->R2]],B6->[M3->M2->M1->R2,M2->[M3->M1->R2,M1->M3->R2]]]`
## [1] 0 1 1 0 1 1 1 0
```

```
##
## $...
## [1] 1 1 1 0 1 1 1 0
##
## $`[B2->[M1->M3->M2->R1,M3->[M1->M2->R1,M2->M1->R1]],B3->M1->M3->M2-
>R1,B4->[M1->M2->M3->R1,M2->[M3->M1->R1,M1->M3->R1]],B5->M1->M2->M3-
>R1,B6->[M3->[M1->M2->R1,M2->M1->R1],M2->[M3->M1->R1,M1->M3->R1]]]`
## [1] 0 0 1 1 1 1 1 0
##
## $...
## [1] 1 0 1 1 1 1 1 0
##
## $...
## [1] 0 1 1 1 1 1 1 0
##
## $...
## [1] 1 1 1 1 1 1 1 0
##
## $B7
## [1] 0 0 0 0 0 0 0 1
##
## $`[B3->M3,B7->M3]`
## [1] 0 0 0 1 0 0 0 1
##
## $`[B1->M2->M3,B3->[M3->M2,M2->M3],B7->M3->M2]`
## [1] 0 1 0 1 0 0 0 1
##
## $`[B0->M1->M2->M3,B1->[M1->M2->M3,M2->[M3->M1,M1->M3]],B3->[M3->M2-
>M1,M2->[M3->M1,M1->M3]],B7->M3->M2->M1]`
## [1] 1 1 0 1 0 0 0 1
##
## $`[B2->M1->M3,B3->[M1->M3,M3->M1],B7->M3->M1]`
## [1] 0 0 1 1 0 0 0 1
##
## $`[B0->M2->M1->M3,B2->[M2->M1->M3,M1->[M2->M3,M3->M2]],B3->[M3->M1-
>M2,M1->[M2->M3,M3->M2]],B7->M3->M1->M2]`
## [1] 1 0 1 1 0 0 0 1
##
## $`[B1->M2->[M3->M1,M1->M3],B2->M1->[M2->M3,M3->M2],B3->[M2->[M3-
>M1,M1->M3],[M1->[M2->M3,M3->M2],M3->[M1->M2,M2->M1]]],B7->M3->[M1-
>M2,M2->M1]]`
## [1] 0 1 1 1 0 0 0 1
##
## $...
## [1] 1 1 1 1 0 0 0 1
##
```

```
## $`[B0->[M3->M1->M2->R2,M1->[M2->M3->R2,M3->M2->R2]],B1->[M1->[M2-
>M3->R2,M3->M2->R2],M2->[M3->M1->R2,M1->M3->R2]],B3->[M3->M2->M1-
>R2,M2->[M3->M1->R2,M1->M3->R2]],B4->M3->M1->M2->R2,B7->M3->M2->M1-
>R2]`
## [1] 1 1 0 1 1 0 0 1
##
## $`[B0->[M3->M2->M1->R2,M2->[M3->M1->R2,M1->M3->R2]],B2->[M1->[M2-
>M3->R2,M3->M2->R2],M2->[M3->M1->R2,M1->M3->R2]],B3->[M3->M1->M2-
>R2,M1->[M2->M3->R2,M3->M2->R2]],B4->M3->M2->M1->R2,B7->M3->M1->M2-
>R2]`
## [1] 1 0 1 1 1 0 0 1
##
## $...
## [1] 1 1 1 1 1 0 0 1
##
## $`[B5->M2,B7->M2]`
## [1] 0 0 0 0 0 1 0 1
##
## $`[B1->M3->M2,B5->[M3->M2,M2->M3],B7->M2->M3]`
## [1] 0 1 0 0 0 1 0 1
##
## $`[B0->M1->M3->M2,B1->[M1->M3->M2,M3->[M1->M2,M2->M1]],B5->[M2->M3-
>M1,M3->[M1->M2,M2->M1]],B7->M2->M3->M1]`
## [1] 1 1 0 0 0 1 0 1
##
## $`[B3->M3->M2,B5->M2->M3,B7->[M3->M2,M2->M3]]`
## [1] 0 0 0 1 0 1 0 1
##
## $`[B1->[M3->M2->R2,M2->M3->R2],B3->[M3->M2->R2,M2->M3->R2],B5->[M3-
>M2->R2,M2->M3->R2],B7->[M3->M2->R2,M2->M3->R2]]`
## [1] 0 1 0 1 0 1 0 1
##
## $...
## [1] 1 1 0 1 0 1 0 1
##
## $`[B2->M1->M3->M2,B3->[M1->M3->M2,M3->[M1->M2,M2->M1]],B5->M2->M3-
>M1,B7->[M2->M3->M1,M3->[M1->M2,M2->M1]]]`
## [1] 0 0 1 1 0 1 0 1
##
## $...
## [1] 0 1 1 1 0 1 0 1
##
## $...
## [1] 1 1 1 1 0 1 0 1
##
## $`[B4->M1->M2,B5->[M1->M2,M2->M1],B7->M2->M1]`
```

```
## [1] 0 0 0 0 1 1 0 1
##
## $`[B0->M3->M1->M2,B4->[M3->M1->M2,M1->[M2->M3,M3->M2]],B5->[M2->M1-
>M3,M1->[M2->M3,M3->M2]],B7->M2->M1->M3]`
## [1] 1 0 0 0 1 1 0 1
##
## $`[B1->M3->[M1->M2,M2->M1],B4->M1->[M2->M3,M3->M2],B5->[M2->[M3-
>M1,M1->M3],[M1->[M2->M3,M3->M2],M3->[M1->M2,M2->M1]]],B7->M2->[M3-
>M1,M1->M3]]`
## [1] 0 1 0 0 1 1 0 1
##
## $...
## [1] 1 1 0 0 1 1 0 1
##
## $`[B3->M3->M2->M1,B4->M1->M2->M3,B5->[M1->M2->M3,M2->[M3->M1,M1-
>M3]],B7->[M3->M2->M1,M2->[M3->M1,M1->M3]]]`
## [1] 0 0 0 1 1 1 0 1
##
## $`[B0->M3->M1->M2->R2,B3->M3->M2->M1->R2,B4->[M3->M1->M2->R2,M1-
>[M2->M3->R2,M3->M2->R2]],B5->[M1->[M2->M3->R2,M3->M2->R2],M2->[M3->M1-
>R2,M1->M3->R2]],B7->[M3->M2->M1->R2,M2->[M3->M1->R2,M1->M3->R2]]]`
## [1] 1 0 0 1 1 1 0 1
##
## $...
## [1] 0 1 0 1 1 1 0 1
##
## $...
## [1] 1 1 0 1 1 1 0 1
##
## $`[B2->M1->M3->M2->R1,B3->[M1->M3->M2->R1,M3->[M1->M2->R1,M2->M1-
>R1]],B4->M1->M2->M3->R1,B5->[M1->M2->M3->R1,M2->[M3->M1->R1,M1->M3-
>R1]],B7->[M3->[M1->M2->R1,M2->M1->R1],M2->[M3->M1->R1,M1->M3->R1]]]`
## [1] 0 0 1 1 1 1 0 1
##
## $...
## [1] 1 0 1 1 1 1 0 1
##
## $...
## [1] 0 1 1 1 1 1 0 1
##
## $...
## [1] 1 1 1 1 1 1 0 1
##
## $`[B6->M1,B7->M1]`
## [1] 0 0 0 0 0 0 1 1
##
```

```
## $`[B2->M3->M1,B6->[M1->M3,M3->M1],B7->M1->M3]`
## [1] 0 0 1 0 0 0 1 1
##
## $`[B0->M2->M3->M1,B2->[M2->M3->M1,M3->[M1->M2,M2->M1]],B6->[M1->M3-
>M2,M3->[M1->M2,M2->M1]],B7->M1->M3->M2]`
## [1] 1 0 1 0 0 0 1 1
##
## $`[B0->[M1->M2->M3->R1,M2->[M3->M1->R1,M1->M3->R1]],B1->M1->M2->M3-
>R1,B2->[M3->[M1->M2->R1,M2->M1->R1],M2->[M3->M1->R1,M1->M3->R1]],B6-
>[M1->M3->M2->R1,M3->[M1->M2->R1,M2->M1->R1]],B7->M1->M3->M2->R1]`
## [1] 1 1 1 0 0 0 1 1
##
## $`[B3->M3->M1,B6->M1->M3,B7->[M1->M3,M3->M1]]`
## [1] 0 0 0 1 0 0 1 1
##
## $`[B1->M2->M3->M1,B3->[M2->M3->M1,M3->[M1->M2,M2->M1]],B6->M1->M3-
>M2,B7->[M1->M3->M2,M3->[M1->M2,M2->M1]]]`
## [1] 0 1 0 1 0 0 1 1
##
## $`[B0->M1->M2->M3->R1,B1->[M1->M2->M3->R1,M2->[M3->M1->R1,M1->M3-
>R1]],B3->[M3->[M1->M2->R1,M2->M1->R1],M2->[M3->M1->R1,M1->M3->R1]],B6-
>M1->M3->M2->R1,B7->[M1->M3->M2->R1,M3->[M1->M2->R1,M2->M1->R1]]]`
## [1] 1 1 0 1 0 0 1 1
##
## $`[B2->[M1->M3->R1,M3->M1->R1],B3->[M1->M3->R1,M3->M1->R1],B6->[M1-
>M3->R1,M3->M1->R1],B7->[M1->M3->R1,M3->M1->R1]]`
## [1] 0 0 1 1 0 0 1 1
##
## $...
## [1] 1 0 1 1 0 0 1 1
##
## $...
## [1] 0 1 1 1 0 0 1 1
##
## $...
## [1] 1 1 1 1 0 0 1 1
##
## $`[B4->M2->M1,B6->[M1->M2,M2->M1],B7->M1->M2]`
## [1] 0 0 0 0 1 0 1 1
##
## $`[B0->M3->M2->M1,B4->[M3->M2->M1,M2->[M3->M1,M1->M3]],B6->[M1->M2-
>M3,M2->[M3->M1,M1->M3]],B7->M1->M2->M3]`
## [1] 1 0 0 0 1 0 1 1
##
## $`[B0->[M1->M3->M2->R1,M3->[M1->M2->R1,M2->M1->R1]],B1->M1->M3->M2-
>R1,B4->[M3->[M1->M2->R1,M2->M1->R1],M2->[M3->M1->R1,M1->M3->R1]],B6-
```

```
>[M1->M2->M3->R1,M2->[M3->M1->R1,M1->M3->R1]],B7->M1->M2->M3->R1]`
## [1] 1 1 0 0 1 0 1 1
##
## $`[B2->M3->[M1->M2,M2->M1],B4->M2->[M3->M1,M1->M3],B6->[M2->[M3-
>M1,M1->M3],[M1->[M2->M3,M3->M2],M3->[M1->M2,M2->M1]]],B7->M1->[M2-
>M3,M3->M2]]`
## [1] 0 0 1 0 1 0 1 1
##
## $...
## [1] 1 0 1 0 1 0 1 1
##
## $...
## [1] 1 1 1 0 1 0 1 1
##
## $`[B3->M3->M1->M2,B4->M2->M1->M3,B6->[M2->M1->M3,M1->[M2->M3,M3-
>M2]],B7->[M3->M1->M2,M1->[M2->M3,M3->M2]]]`
## [1] 0 0 0 1 1 0 1 1
##
## $`[B0->M3->M2->M1->R2,B3->M3->M1->M2->R2,B4->[M3->M2->M1->R2,M2-
>[M3->M1->R2,M1->M3->R2]],B6->[M1->[M2->M3->R2,M3->M2->R2],M2->[M3->M1-
>R2,M1->M3->R2]],B7->[M3->M1->M2->R2,M1->[M2->M3->R2,M3->M2->R2]]]`
## [1] 1 0 0 1 1 0 1 1
##
## $...
## [1] 1 1 0 1 1 0 1 1
##
## $...
## [1] 0 0 1 1 1 0 1 1
##
## $...
## [1] 1 0 1 1 1 0 1 1
##
## $...
## [1] 1 1 1 1 1 0 1 1
##
## $`[B5->M2->M1,B6->M1->M2,B7->[M1->M2,M2->M1]]`
## [1] 0 0 0 0 0 1 1 1
##
## $`[B1->M3->M2->M1,B5->[M3->M2->M1,M2->[M3->M1,M1->M3]],B6->M1->M2-
>M3,B7->[M1->M2->M3,M2->[M3->M1,M1->M3]]]`
## [1] 0 1 0 0 0 1 1 1
##
## $`[B0->M1->M3->M2->R1,B1->[M1->M3->M2->R1,M3->[M1->M2->R1,M2->M1-
>R1]],B5->[M3->[M1->M2->R1,M2->M1->R1],M2->[M3->M1->R1,M1->M3->R1]],B6-
>M1->M2->M3->R1,B7->[M1->M2->M3->R1,M2->[M3->M1->R1,M1->M3->R1]]]`
## [1] 1 1 0 0 0 1 1 1
```

```
##
## $`[B2->M3->M1->M2,B5->M2->M1->M3,B6->[M3->M1->M2,M1->[M2->M3,M3-
>M2]],B7->[M2->M1->M3,M1->[M2->M3,M3->M2]]]`
## [1] 0 0 1 0 0 1 1 1
##
## $`[B1->M3->M2->M1->R2,B2->M3->M1->M2->R2,B5->[M3->M2->M1->R2,M2-
>[M3->M1->R2,M1->M3->R2]],B6->[M3->M1->M2->R2,M1->[M2->M3->R2,M3->M2-
>R2]],B7->[M1->[M2->M3->R2,M3->M2->R2],M2->[M3->M1->R2,M1->M3->R2]]]`
## [1] 0 1 1 0 0 1 1 1
##
## $...
## [1] 1 1 1 0 0 1 1 1
##
## $`[B3->M3->[M1->M2,M2->M1],B5->M2->[M3->M1,M1->M3],B6->M1->[M2-
>M3,M3->M2],B7->[M2->[M3->M1,M1->M3],[M1->[M2->M3,M3->M2],M3->[M1-
>M2,M2->M1]]]]`
## [1] 0 0 0 1 0 1 1 1
##
## $...
## [1] 0 1 0 1 0 1 1 1
##
## $...
## [1] 1 1 0 1 0 1 1 1
##
## $...
## [1] 0 0 1 1 0 1 1 1
##
## $...
## [1] 0 1 1 1 0 1 1 1
##
## $...
## [1] 1 1 1 1 0 1 1 1
##
## $`[B4->[M1->M2->R1,M2->M1->R1],B5->[M1->M2->R1,M2->M1->R1],B6->[M1-
>M2->R1,M2->M1->R1],B7->[M1->M2->R1,M2->M1->R1]]`
## [1] 0 0 0 0 1 1 1 1
##
## $...
## [1] 1 0 0 0 1 1 1 1
##
## $...
## [1] 0 1 0 0 1 1 1 1
##
## $...
## [1] 1 1 0 0 1 1 1 1
##
```

```
## $...
## [1] 0 0 1 0 1 1 1 1
##
## $...
## [1] 1 0 1 0 1 1 1 1
##
## $...
## [1] 0 1 1 0 1 1 1 1
##
## $...
## [1] 1 1 1 0 1 1 1 1
##
## $...
## [1] 0 0 0 1 1 1 1 1
##
## $...
## [1] 1 0 0 1 1 1 1 1
##
## $...
## [1] 0 1 0 1 1 1 1 1
##
## $...
## [1] 1 1 0 1 1 1 1 1
##
## $...
## [1] 0 0 1 1 1 1 1 1
##
## $...
## [1] 1 0 1 1 1 1 1 1
##
## $...
## [1] 0 1 1 1 1 1 1 1
```

Visualization LD using the catalogue and dataset from the HapMap project.

In this section we need to crate a function which compute the Euclidean distance between two vectors(which represent LD values), a vector from the catalogue and a vector from the real datasets(HapMap). In order to calculate the haplotypes frequencies and the LD values, the below functions have been used.

```
# functions to calculate the cumulants
std <- function(v)(v/sum(v))

#' haplotypes to std.cumulants
hfs2std.cumu = function(hfs)new(parametrizer)$multinomial2cumuStd(hfs)
```

```
#' std.cumulants to haplotypes
cumu2hfs = function(cumuStd){
  hfs <-new(parametrizer)$cumuStd2multinomial(cumuStd)
  std(hfs+1^(-6))
}

# Euclidean distance matrix
Euc.distanceMatrix <- function(data.cumulants,cat.cumulants){
  mat <-
matrix(0,ncol=length(data.cumulants),nrow=length(cat.cumulants))
  for(i in 1:length(data.cumulants)){
    for(j in 1:length(cat.cumulants)){
      mat[j,i]<-
dist(rbind(as.numeric(unlist(data.cumulants[i])),as.numeric(unlist(cat.
cumulants[j]))))
    }
  }
  return(mat)
}

################################
#    Catalogue cumulants     #
################################
cat.cumulants <- lapply(catalog,function(x)hfs2std.cumu(unlist(x)) )

# We delete the allele frequencies of the cat.cumulants:
remove <- c (2,3,5)
for(i in 1:length(cat.cumulants)){
  cat.cumulants[[i]]<-cat.cumulants[[i]][-remove]
}
```

The same steps we follow using the datasets from the HapMap. Finally, we take a matrix ("mat") with all Euclidean differences between catalogue and real datasets. We illustrate the distances of LD using Heatmaps plots.

```
mat <-Euc.distanceMatrix(cumulants,cat.cumulants)

#colnames(mat)<-snp.list

# visualization

#Heatmap
#colfunc1 <- colorRampPalette(c("orange","green", "black"))
#colfunc2 <- colorRampPalette(c("black", "red"))
#hmcols <- c(colfunc1(200), colfunc2(200*(max(mat) - 1)))

#op = par(bg = "#EFEFEF")
#heatmap.2(mat,Colv=F,Rowv=T,dendrogram="row"
```

```
#          ,scale="none",col=hmcols,trace="none"
#          , margin=c(5,10), key=T,labRow=NA)
#

#hc = hclust(dist(mat))
```

Monte Carlo Simulations under the null hypothesis

In this part, we use parametric bootstrap techniques to simulate dataset based on two catalogue entries. First, we select the two catalogue entries then compute the std.cumulantsuse the function cumu2hfs in order to get the mean of haplotypes frequencies of the 2 entries:. Simulate based on this hfs

```
# simulate datasets
sim.dataset <-function(N,nsize,catalog,allelefreq){
  # calculate the std. cumulants
  # the catalogue should have two entries
  cat.cumulants <- lapply(catalog,function(x)hfs2std.cumu(x) )
  # the mean between cumulants
  lambda <- (1/2)*(cat.cumulants[[1]]+cat.cumulants[[2]])

  if(!missing(allelefreq)) {
    remove <-c(2,3,5)
    lambda[remove]<-allelefreq
  }
  hfs <- cumu2hfs(lambda)
  #simulate datasets
  seqs <- c(0:(2^N-1))
  index <-seqs %*% rmultinom(nsize, 1, hfs)
  snp.sim.data <-t(sapply(index,function(x) {ord2bin (x,N)}))
  return(snp.sim.data)
}

####
distance <- function(sim.data,catalog){

  # catalogue std.cumulants:
  cat.cumulants <- lapply(catalog,function(x)hfs2std.cumu(x) )

  # sim.data: calculate the the haplotype frequency
  htfs.snps <- haplotypeFrequencies(sim.data)

  # calculate the cumulants and std.cumulants
  sim.cumulants <- hfs2std.cumu(htfs.snps)

  #Delete the allele frequency
  remove <- c (2,3,5)
```

```
  sim.cumulants<- sim.cumulants[-remove]

  for(i in 1:length(cat.cumulants)){
    cat.cumulants[[i]]<-cat.cumulants[[i]][-remove]
  }

  R <- Euc.distanceMatrix.2(sim.cumulants,cat.cumulants)
  return(R)
}

# Parametric bootstrap
rboot <- function(B,data,catalog){
  # storage the results
  R.b <- matrix(0,nrow=length(catalog),ncol=B)
  # data: calculate the hfs, cumulants and allele frequencies
  data.hfs <- haplotypeFrequencies(data)
  data.cumu <- hfs2std.cumu(data.hfs)
  remove <- c(2,3,5)
  allelefreq <- data.cumu[remove]
 #allelefreq <-c(0.5,0.5,0.5)
  for(b in 1:B){
    Bsnps <- sim.dataset(ncol(data),nrow(data),catalog,allelefreq)
    R.b[,b] <-distance(Bsnps,catalog)
  }
  return (R.b)
}

#  Estimated Test Statistic T=d2-d1
row.differences <- function(data){
  theta <- NULL
  data <- as.data.frame(data)
  theta <-data[2,]-data[1,]
  return(theta)
}

test.data <- function(B,data,catalog){
  #simulate datasets
 datasim <-data

  # the euc.distance between catalogue and sim.data
  true.parameter <- distance(datasim,catalog)
  # the differences of two entries
  cat.diff.Tparameter<-row.differences(true.parameter)
  # bootstrap replications
  boot <- rboot(B, datasim,catalog)
  #simulation diffenences
```

```
  sim.diff <-row.differences(boot)
  return(list(cat.diff.Tparameter,sim.diff))
}

test.statistic <- function(B,N,nsize,catalog,allelefreq){
  #simulate datasets
  datasim <- sim.dataset(N,nsize,catalog,allelefreq)
  test.data(B,datasim,catalog)
}


# Repeat the bootstrap procedure fon rep=500/1000 times

sim.results <- function(B,rep,N,nsize,catalog,allelefreq){
  sboot<-list()
  for(i in 1:rep){
  sboot[[i]]<-test.statistic(B,N,nsize,catalog,allelefreq)
  }
  return(sboot)
}
```

Testing procedure

```
quantiles <-function(data,lower,upper){
  quant <- quantile(data[[2]],prob=c(lower,upper))
  quant$test.statistic <-data[[1]]
  return(quant)
}


make.data.frame1 <- function(confidence.interval){
  q<-matrix(0,ncol=3,nrow=length(confidence.interval))
  for(i in 1:length(confidence.interval)){
    q[i,]<-(t(as.numeric(confidence.interval[[i]])))
  }
  q<-as.data.frame(q)
  colnames(q) <- c( "5%","95%","test statistic" )
  return(q)
}


# Test of Hypothesis 1
test1 <- function(data){
  #data = is the table with the quantiles
  #data$"Ho.1"<-rep(0,nrow(data))
  for(i in 1:nrow(data)){
    if(data[i,1]>=data[i,3]){
```

```
      data[i,4] <- (" reject")
    }else{
      data[i,4] <- ("Not reject")
    }
  }
  #colnames(data[,4]) <-"Ho.1"
  return(data)
}

# Test of Hypothesis 2
test2 <- function(data){
  #data = is the table with the quantiles

  for(i in 1:nrow(data)){
    ifelse(data[i,2]<=data[i,3], data[i,4] <- ("reject")
           ,data[i,4] <- ("Not reject"))
  }
  #colnames(data[,4]) <-"Ho.2"
  return(data)
}

# Test of Hypothesis 3
test3 <-function(data){
  #data = is the table with the quantiles
  #if the prod(ci)<0 then the 0 exist into the interval
  for(i in 1:nrow(data)){
    ifelse(data[i,1]<=data[i,3] & data[i,2]>=data[i,3], data[i,4] <-
("Not reject")
           ,data[i,4] <- ("reject"))
  }
  return(data)
}
```

Example

The example shows the comparison between catalogue 1 and catalogue 91 for different seanarios. (Bootstrap=1000, simulations=500, number of loci=3, number of haplotypes=120)

```
# 1_91

results1_91 <-sim.results(1000,500,3,120,c(catalog[1],catalog[91]))
save.list(results1_91,"results1_91")

results1_91.2 <-sim.results(1000,500,3,2*120,c(catalog[1],catalog[91]))
save.list(results1_91.2,"results1_91.2")

results1_91.3 <-sim.results(1000,500,3,4*120,c(catalog[1],catalog[91]))
```

```
save.list(results1_91.3,"results1_91.3")

results1_91.4 <-sim.results(1000,500,3,6*120,c(catalog[1],catalog[91]))
save.list(results1_91.4,"results1_91.4")

results <-load.list("results1_91")


quantiles.results <- lapply(results,function(x)quantiles(x,0.05,0.95))
quantiles.results.d <-make.data.frame1(quantiles.results)
conf.interval <- lapply(results,function(x)quantiles(x,0.025,0.975))
conf.interval.d <-make.data.frame1(conf.interval)

# Tests

test1(quantiles.results.d)
test2(quantiles.results.d)
test3(conf.interval.d)

t1<-test1(quantiles.results.d)
mean(as.character(t1[,4])==" reject")

t2<-test2(quantiles.results.d)
mean(as.character(t2[,4])=="reject")

t3<-test3(conf.interval.d)
mean(as.character(t3[,4])=="reject")
```

Two alternative approaches have been proposed to this thesis: i) to increase the iteration of simulation procedure. This can been implemented easily, by introducing B=2000 and rep=1000. ii) to set up alleles frequencies equal to 0.5. Using the attribute "allelefreq=c(0.5,0.5,0.5)" and modifing a little bit the bootstrap function.

Monte Carlo Simulations under the alternative hypothesis

Here, only the simulation procedure have been modified, since the testing procedure remains the same.

A) For the *approach 1*.

```
sim.dataset <-function(N,nsize,catalog,allelefreq){
  # calculate the std. cumulants
  # the catalogue should have two entries
  cat.cumulants <- lapply(catalog,function(x)hfs2std.cumu(x) )
  # the mean between cumulants
  kappa <- (1/2)*(cat.cumulants[[1]]+cat.cumulants[[2]])
  lambda <- (1/2)*(kappa + cat.cumulants[[2]])
  if(!missing(allelefreq)) {
    remove <-c(2,3,5)
    lambda[remove]<-allelefreq
```

```
  }
  hfs <- cumu2hfs(lambda)
  #simulate datasets
  seqs <- c(0:(2^N-1))
  index <-seqs %*% rmultinom(nsize, 1, hfs)
  snp.sim.data <-t(sapply(index,function(x) {ord2bin (x,N)}))
  return(snp.sim.data)
}
```

B) For the *approach 2*

```
sim.dataset <-function(N,nsize,catalog,allelefreq){
  # calculate the std. cumulants
  # we dont delete the alleles frequencies.
  # the catalogue should have two entries
  cat.cumulants <- lapply(catalog,function(x)hfs2std.cumu(x) )
  # cumulants of one catalogue entry
  lambda <- (cat.cumulants[[2]])
  if(!missing(allelefreq)) {
    remove <-c(2,3,5)
    lambda[remove]<-allelefreq
  }
  hfs <- cumu2hfs(lambda)
  #simulate datasets
  seqs <- c(0:(2^N-1))
  index <-seqs %*% rmultinom(nsize, 1, hfs)
  snp.sim.data <-t(sapply(index,function(x) {ord2bin (x,N)}))
  return(snp.sim.data)
}
```

HamMap analysis

For the HapMap analysis we select 7 SNPs, where each SNP dataset consists of 120 haplotypes.

```
chr21 = readPhasedData('genotypes_chr21_CEU_r22_nr.b36_fwd')

# SNP 1
snp.name <- chr21$legend[21000,1]
snps <- SelectSNPsOrder(as.character(snp.name), chr21, marginNeg = 1,
marginPos = 1, maf = .05,buffer =5)$hts

#SNP 2
snp.name <- chr21$legend[100,1]
snps <- SelectSNPsOrder(as.character(snp.name), chr21, marginNeg = 1,
marginPos = 1, maf = .05,buffer =5)$hts

#SNP 3
```

```
snps = SelectSNPsOrder('rs2829806', chr21, marginNeg = 1, marginPos =
1, maf = .05,buffer =5)$hts

#SNP 4

snps = SelectSNPsOrder('rs1057885', chr21, marginNeg = 1, marginPos =
1, maf = .05,buffer =5)$hts

#SNP 5
snps = SelectSNPsOrder('rs11088561', chr21, marginNeg = 1, marginPos =
1, maf = .05,buffer =5)$hts

# SNP 6
snps = SelectSNPsOrder('rs1534', chr21, marginNeg = 1, marginPos = 1,
maf = .05,buffer =5)$hts

# SNP 7

snp.name <- chr21$legend[31363,1]
snps = SelectSNPsOrder(as.character(snp.name), chr21, marginNeg = 1,
marginPos = 1, maf = .05,buffer =5)$hts
```

Using the the R codes that have been describe above, we can simulate dataset using the haplotypes frequencies of each SNP and we can test the three hypotheses.

References

B Alberts. Molecular biology of the cell. 5th edition, 2007. 1, 2, 4, 5, 7

David J Balding. A tutorial on statistical methods for population association studies. *Nat Rev Genet*, 7(10):781–91, Oct 2006. doi: 10.1038/nrg1916. 23

B Balliu, Houwing-Duistermaat J J, and Boehringer S. Powerful testing via hierarchical linkage disequilibrium in haplotype association studies. *manuscript submitted*. 1, 27

The International HapMap Consortium. The international hapmap project. *Nature*, 2003. 2, 6, 28

L Duret. The null hypothesis of molecular evolution. *Nature*, 2008. 8

Root Gorelick and Manfred D Laubichler. Decomposing multilocus linkage disequilibrium. *Genetics*, 166(3):1581–3, Mar 2004. 1, 25

R C Griffiths and P Marjoram. Ancestral inference from samples of dna sequences with recombination. *J Comput Biol*, 3(4):479–502, 1996. 13

M Kimura. Evolutionary rate at the molecular level. *Nature*, 217(5129):624–6, Feb 1968. 1, 8

R C Lewontin. On measures of gametic disequilibrium. *Genetics*, 120(3):849–52, Nov 1988. 1, 23

M S McPeek and A Strahs. Assessment of linkage disequilibrium by the decay of haplotype sharing, with application to fine-scale genetic mapping. *Am J Hum Genet*, 65(3):858–75, Sep 1999. doi: 10.1086/302537. 1

Jakob C Mueller. Linkage disequilibrium for different scales and applications. *Brief Bioinform*, 5(4):355–64, Dec 2004. 25

Casey E Romanoski, Christopher K Glass, Hendrik G Stunnenberg, Laurence Wilson, and Genevieve Almouzni. Epigenomics: Roadmap for regulation. *Nature*, 518(7539): 314–6, Feb 2015. doi: 10.1038/518314a. 6

Yun S Song and Jotun Hein. Constructing minimal ancestral recombination graphs. *J Comput Biol*, 12(2):147–69, Mar 2005. doi: 10.1089/cmb.2005.12.147. 11

Richard S Spielman, Laurel A Bastone, Joshua T Burdick, Michael Morley, Warren J Ewens, and Vivian G Cheung. Common genetic variants account for differences in gene expression among ethnic groups. *Nat Genet*, 39(2):226–31, Feb 2007. doi: 10.1038/ng1955. 2

J D Watson and F H C Crick. A structure for deoxyribose nucleic acid. 1953. *Nature*, 421(6921):397–8; discussion 396, Jan 1953. 4

Bruce S. Weir. *Genetic Data Analysis II*. 1996. 26