POLITECNICO DI TORINO

Master's Degree in Computer Engineering
Software

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech

Master in Innovation and Research in Informatics
Data Science
Facultat d'Informàtica de Barcelona (FIB)

Master's Degree Thesis

A network algorithm for exact tests for Hardy-Weinberg equilibrium with X-chromosomal variants

Supervisors

Prof. Jan Graffelman

Prof. Mauro Gasparini

Candidate

Leonardo Ortoleva

July 2020

Abstract

In statistical genetics, the *exact test* is of fundamental importance to assess whether, in a population, genotypes are distributed according to the Hardy-Weinberg law. Genetic variables are usually filtered on the basis of exact test results to avoid genotyping errors, which can negatively affect subsequent analyses in genetic epidemiology, ecology, forensics among others. It is used to test for statistical independence of the different alleles within the same locus. Given this importance, it is useful to understand which are the current tools to calculate it and how efficient they are. For a bi-allelic context it is possible to use asymptotic tests, such as the chi-square test. Widening the point of view and taking into account more alleles the data are more scattered within the genotype matrix and the exact test is the most reliable. This involves using the probability of the genotype counts you are observing, and then extracting the final p-value from which you can deduce whether or not there is evidence for deviation from equilibrium. All this comes to a very high precision which is, however, at the expense of performance. A further step is to consider the X chromosome, where the different conformation of the two sexes causes considerable computational difficulties. In this master thesis, the network algorithm available for the autosomes has been extended for the X chromosome, achieving spectacular improvements in computation time with respect to state-of-the art enumeration algorithms, avoiding the repetition of calculations that can be stored and taking advantage of the recursive technique. We analysed complete chromosomes of the 1,000 Genomes Project in order to evaluate exact algorithms and results for autosomal and X chromosomal variants with two or more alleles. Its functioning has been verified first of all in terms of final results with the few tools currently available, with excellent outcomes. Subsequently a performance analysis has highlighted the great usefulness of the result achieved, reducing by several orders of magnitude the computation time to precise values. In some cases, an analysis lasting more than 6 hours reduced to a few seconds to obtain the same output values. This is an excellent step forward that delivers an interesting new tool enabling efficient exact Hardy-Weinberg testing for X chromosomal variants up to at least five alleles.

Table of Contents

List of Tables

List of Figures

Acronyms

DNA
Deoxyribonucleic acid

RNA
Ribonucleic acid

HWE
Hardy-Weinberg equilibrium

HWP
Hardy-Weinberg proportions

SNP
Single Nucleotide Polymorphisms

STR
Short Tandem Repeats

QTL
Quantitative trait loci

CRP
C-reactive protein

LR
Likelihood Ratio

Chapter 1

Introduction

1.1 Context

"Life as we know it is specified by the genomes of the myriad of organisms with which we share the planet. Every organism possesses a genome that contains the biological information needed to construct and maintain a living example of that organism. Most genomes, including the human genome and those of all other cellular life forms, are made of DNA (*deoxyribonucleic acid*) but a few viruses have RNA (*ribonucleic acid*) genomes. DNA and RNA are polymeric molecules made up of chains of monomeric sub-units called nucleotides".
With these words Brown (2002) begins his book, *The Human Genome*, in which he introduces several concepts that are of fundamental importance for understanding what will be explained later. DNA is a fundamental junction point in a hierarchy (see Figure 1.1) which, starting from the human being in general, reaches down to the individual genes of different forms. Only having some concepts clear, which will now be exposed, it will then be possible to better understand the field of statistical genetics and reach useful results from different points of view.
DNA appears in a fairly stable and particular structure called *double helix* and is organized into *chromosomes*. On the upper level there is the *chromosome* which is composed of proteins associated with a single DNA molecule. The DNA or RNA traits make up the *genes*, or the hereditary molecules that transfer the traits to the offspring. Of fundamental importance is the difference between DNA and genes: while DNA is a chemical structure that stores genetic traits, on the other hand genes are small pieces of DNA from which a specific trait of the individual is obtained who owns it (Panawala, 2017).

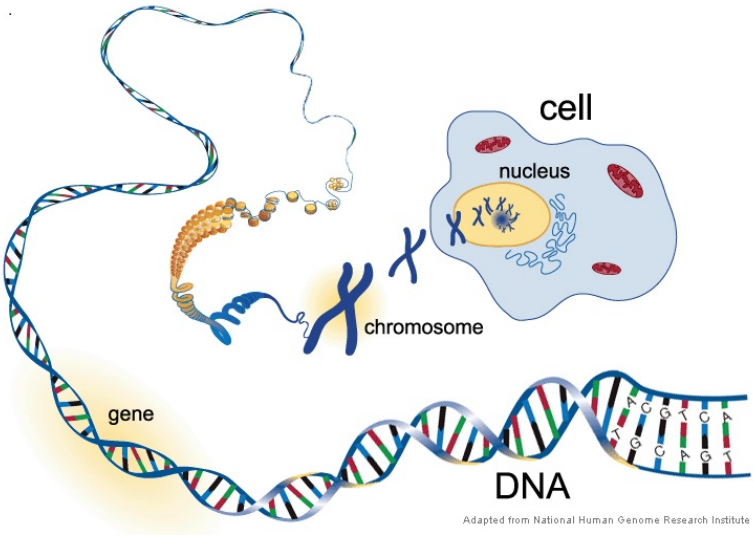


Figure 1.1: Hierarchical structure DNA (from National Human Genome Research Institute)

DNA

Deoxyribonucleic acid (DNA) is a nucleic acid which contains the genetic information necessary for the biosynthesis of RNA and proteins, essential molecules for the development and correct functioning of most living organisms and for their reproduction. From a chemical point of view, DNA is an organic polymer made up of monomers called *nucleotides*. All nucleotides are made up of three basic components: a phosphate group, deoxyribose, a pentose sugar and a nitrogen base. The latter binds to deoxyribose with an N-glycosidic bond. There are four nitrogenous bases that can be used in the formation of nucleotides to be incorporated in the DNA molecule: adenine (A), guanine (G), cytosine (C) and thymine (T), while in RNA, instead of thymine, uracil (U) is present. It is possible to define DNA as an antiparallel oriented and complementary polynucleotide chain. These bases are arranged in different orders in order to store the genetic information. The order of the nucleotide sequence determines the characteristics of the genes that form part of it. Two polynucleotide chains are joined together by hydrogen bonds between complementary base pairs. This process is called complement base pairing and it produces a doublestranded DNA molecule where each strand is complementary. Doublestranded DNA is further coiled to form a **double-helix** structure (see Figure 1.2). The two strands of a double-helix run into opposite directions, making them antiparallel. The asymmetric ends of the strand are called $3'$ and $5'$ ends. An organism's complete set of DNA is called a genome (Panawala, 2017).

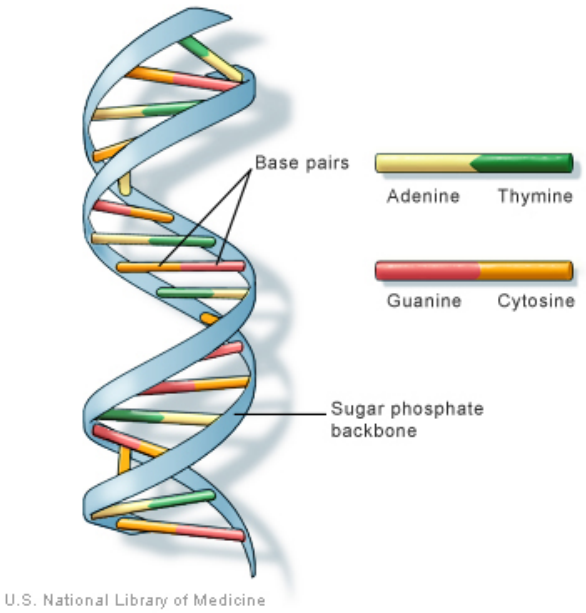


Figure 1.2: DNA is a double helix formed by base pairs attached to a sugar-phosphate backbone (from U.S. National Library of Medicine)

The human genome consists of two different parts, which together create the typical element of the genomes of all multicellular animals:

- the *nuclear genome*, comprise about 3.2 million DNA nucleotides, divided into 24 linear molecules.
- the *mitochondrial genome* is a circular DNA molecule of more than 16 thousands nucleotides, multiple copies of which are found in mitochondria, which are small organs involved in cellular respiration (Brown, 2002).

Gene

Going further down the hierarchy, we find the **gene** which is a region (*locus*) or a specific nucleotide sequence on the DNA strand. Genes encode a *amino acid sequence* of a specific protein.
Genetic instructions are transferred to offspring with reproduction thanks to genes, which are considered as hereditary molecular units. Genes are expressed by means of mRNA (messenger RNA), which encodes and transports information during transcription from DNA. This is certainly the cornerstone of molecular biology. The pioneer of these studies was, in 1860, Gregor Mendel, who managed to express

the concept of gene and to give a model of inheritance through many famous experiments. The whole theory developed by Mendel is very interesting to get to the bottom of the biological concepts from which we will start in this study. Despite this, the theory is very detailed and, for this reason, it was decided not to deal with it since it is not the central objective of the thesis that we want to pursue (Panawala, 2017).

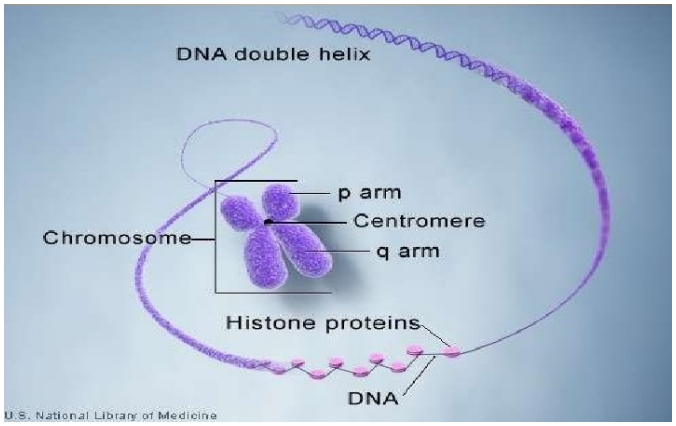


Figure 1.3: Genes are made up of DNA, each chromosome contains many genes (from U.S. National Library of Medicine)

Allele

In nature, some genes occur in a variety of different forms, located in the same position, or genetic locus, on a chromosome. Each variant form of a gene is called an **allele**. Organisms that have two alleles are called diploids; these include humans with one allele inherited from the father and one from the mother. Each pair of alleles represents the genotype of an individual for a specific gene. The genotypes are described as *homozygous* if they consist of two identical alleles in a given locus (AA or aa) and as *heterozygous* if the two alleles differ (Aa). One of the fundamental elements to which alleles contribute is the phenotype, that is the external aspect, of the organism.
Alleles can be dominant or recessive. When an organism is *heterozygous* at a specific locus and carries a dominant allele and a recessive allele, the organism will express the dominant phenotype (from Learn Science at Scitable).

Autosomes and sex chromosomes

The human being has a total of 46 chromosomes, 23 pairs, which can be divided into two categories: one pair of sex chromosomes (XX or XY) and 22 pairs of

autosomes.

The non-sexual chromosomes present in organisms are called **autosomes**. In humans there are 22 series of autosomes; they are labeled or referenced numerically from 1 to 22 (see Figure 1.4), depending on some properties such as shape and size. These deal with the transfer of genetic information from parents to their offspring, in fact they regulate all the inheritance of the characteristics of an organism. The chromosomes in each pair have the same length and the *centromere*, the region that unites each half of the chromosome, is also placed in the same position. During mitosis (process of reproduction of eukaryotic cells thanks to which 2 daughter cells genetically identical to the progenitor and between them are formed from a single cell) the chromosomes double and then transfer to the daughter cells. Therefore, the new daughter cells will receive a complete chromosome copy, containing the genetic information of their mother cell. Moreover, there is the second category of chromosomes: the ones that determine gender, male or female.
Such chromosomes that play a vital role in determining the gender or sex of humans or other animal species are known as **sex chromosome** or *allosomes*, but are generally called "X" and "Y" .
Among the pairs of chromosomes present in humans, the "XX" pair of chromosomes is the female while the individual who has the "XY" pair of the chromosome is the male.
Each chromosome has a structure with two arms, called the "p" arm (the shorter one) and the "q" arm (the longer one). In the case of sex chromosomes, the "Y" chromosome, which is much smaller than the "X" chromosome, consists of a very long "q" arm and a short "p" arm. In the case of the "X" chromosomes, it has a long and a short arm (see Figures 1.3 and 1.4) (Rachna, 2019).

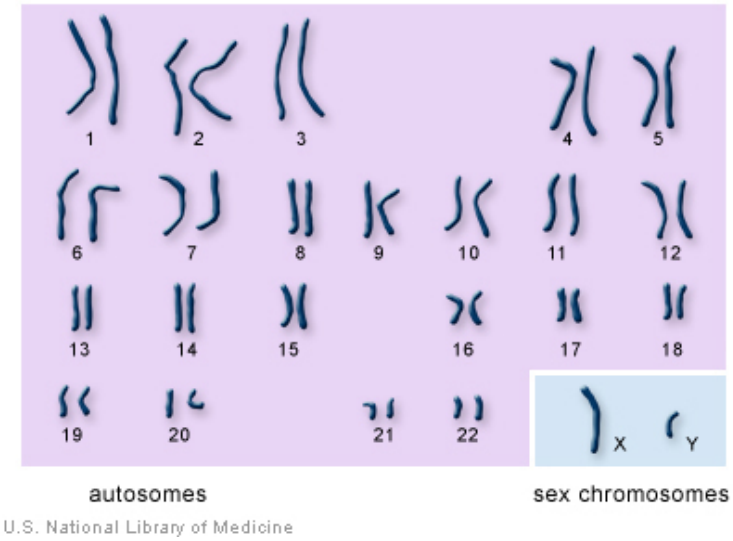


Figure 1.4: Human chromosomes (from U.S. National Library of Medicine)

1.2 Problem statement and goal

From a mathematical point of view, "*Statistical genetics* is a branch of statistics that deals with the analysis of inherited traits and genetic data" (Laird and Lange, 2010).
One of the most important topics covered by this discipline is that of Hardy-Weingberg equilibrium. Godfrey Hardy and Wilhelm Weinberg managed in 1908 to extract a formula which, starting from the frequency distribution of the alleles in the parents, derived the distribution of the genotypes of the offspring.
This law is an excellent junction point between mathematics (in particular statistics) and the medical and biological sphere. Its study is of great importance in the area of bioinformatics and biostatistics. Considering Hardy-Weinberg equilibrium (HWE) in a very simple way, to describe it we start with the concepts of genotype and alleles presented previously. For an autosomal biallelic marker, with only two alleles, the heterozygous (Aa) and the homozygous (AA or aa) forms will appear with the respective frequencies: $2pq$, p^2, q^2, where p is the frequency of the allele of 'A' and $q = 1 - p$ the frequency of the allele 'a' (see Table 1.1).
If there are no particular problems of detection or genotyping errors, which is the main reason for studying this phenomenon, then this proportion will be reached in one generation of mating and will remain so for all subsequent ones. In this case we can speak of Hardy-Weinberg equilibrium (Graffelman and Weir, 2016).

Various statistical tests will be introduced, such as the χ^2 test or the exact test, which can be used to test the existence of equilibrium in empirical genetic data

	A	a
A	p^2	pq
a	pq	q^2

Table 1.1: Genotype frequencies obtained from allele frequencies under HWE

sets.
The tests can be useful to detect two conditions, namely a deficit or an excess of heterozygotes. The first case, corresponding to a homozygote increase, may be due to consanguinity or stratification of the population. The second case can lead to "problems in genotyping due to the existence of highly homologous regions of the genome" (Wigginton et al., 2005).

Over the years, several authors have studied these types of problems, trying to shed light on the important aspects and carrying out tests for HWE from different points of view. The fundamental issues that must be clear in order to understand the work done when talking about HWE are two:

- consider only autosomes or including allosomes in the study
- consider the number of alleles that may be involved

Starting from this last point, techniques that consider both two and more alleles will be described. Obviously the problem increases its complexity by increasing the number of factors. The first point, however, is the key element, to which one wants to propose an innovative and efficient solution. As explained above, there is a difference between autosomes and sex chromosomes. While women have two X chromosomes, men are known to have only one, which is inherited from their mother. The authors Graffelman and Weir (2016) explain very well that the approach used to date has always been to test HWE only for females, thus simplifying a problem that is complex in itself. The aforementioned article also shows several very clear examples that highlight why using this approach leads to evaluation errors and, therefore, it is not correct. Currently, there are algorithms for exact HWE tests that take into account the X-chromosome for males, but only in the case of two alleles, or at most three alleles (since the problem is already computationally very heavy).
Without going into detail, the reasons for treating the X chromosome are also described in articles such as the one written by Broman et al. (2006). Here the authors talk about the mapping of *quantitative trait loci* (QTL), which represents a region of DNA associated with a quantitative trait. From the Figure 1.5, taken from the same article, you can see how, going down between generations, the influence of sex chromosomes is more and more relevant and cannot be overlooked.

The challenge to be faced in the next chapters is to show a solution of a network

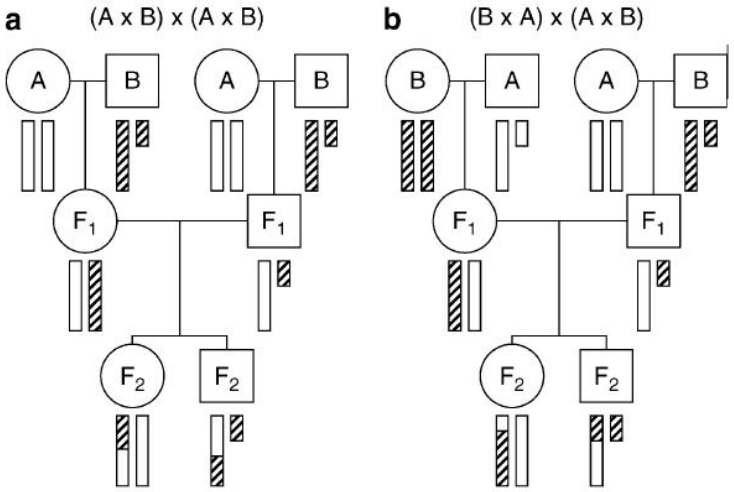


Figure 1.5: "The behavior of the X chromosome in an inter-cross. Circles and squares correspond to females and males, respectively. Open and hatched bars correspond to DNA from strains A and B, respectively. The small bar is the Y chromosome" (Broman et al., 2006).

algorithm that, starting from the one developed by Engels (2009), described in *Exact Tests for Hardy–Weinberg Proportions*, extends the possibility of testing for HWE with more than two alleles, always taking into account the male X-chromosome correctly. This problem was solved for the first time and with very encouraging and satisfactory timing and performance.

1.3 State of the art

This paragraph will briefly summarize the problem analyzed throughout the project, what the starting points were and the tools that are available today to solve them. We here link the introductory chapter and the theoretical explanation of the key statistical tools that form the foundations of the objective set above.
As will be explained later, during the last century there have been really many tools used or invented to solve the problem of the Hardy-Weinberg equilibrium, since the early 1900s, after precisely the study of the two authors after which the equilibrium itself is named. An important turning point certainly occurred in 1949, with Levene being among the precursors of the distribution of the exact test for an arbitrary number of alleles. Among the many successors it is possible to list Louis and Dempster (1987) or even Guo and Thompson (1992), who have worked to deepen the same topic and taking further steps forward. Finally, coming to nowadays, the exact test is analyzed by Wigginton et al. (2005), Engels (2009) and

Graffelman in many published articles. Obviously these names are just some of all those who have worked on this topic and who are also mentioned in all the above articles.
For biallelic variants, the exact test for HWE has become the standard statistical procedure, and it can be performed on a genome-wide scale. From a computational point of view, the problem is feasible and there are no major problems with the calculations even for large sample sizes, as will be seen shortly. The `R` packages, described in the next paragraph, give a big hand in carrying out results that were previously only expressed in theory and that required an excessively long time to resolve.
For k-allelics ($k > 2$), exact tests are computationally expensive, but have been improved a lot and, thanks to the algorithmic optimization that has been achieved, it is possible for large samples of autosomal variants. This is not a trivial thing: you will see how much you can go down recursively in calculations, for example of factorials, for large amounts of data. Without a study that, starting from genetics and statistics, fully exploits the algorithm in all its forms and possibilities to improve over the years, it would be impossible today to solve problems that can then have utility on a very large number of fields, such as medical or forensic science.
In the second half of the last century, some authors (Crow and Kimura, 1970; Hartl, 1980; Lange, 2002) highlighted an aspect hitherto underestimated: "the implications of X-chromosomal inheritance for HWE" (Graffelman and Weir, 2016). In recent years, therefore, efforts have been made to give importance to this situation also in statistical genetics, trying to reach here also exact test distributions that took into consideration the male hemizygous individuals, that is, with only one X chromosome inherited from the parents. This created a lot of confusion, as an already complex problem has further complicated itself. Regarding the X chromosome with only two alleles, it was quite easy to adapt the previous problem, making *small* theoretical and practical changes. So exact tests can be performed for the whole X chromosome, regardless of the amount of data.
Taking a final step, we get to consider the same problem for more than two alleles. Graffelman and Weir (2018) solved the case $k = 3$ with a complete enumeration of the possible cases which, even if with small data sets, explodes in size. Therefore, for the X chromosome a HWE exact test with k alleles ($k > 3$) is even more computationally demanding and is currently not feasible, except by sampling from the possible outcomes using a permutation test.
From here it is easier understood what the final **goal** of the thesis is, that is to improve the efficiency of the current exact procedures to test HWE on X with $k > 3$. On the one hand there is a problem partially solved but whose resolution is very expensive and it needs to be improved, on the other there is a problem that has not yet been analyzed previously and with which, therefore, there are no

possible comparisons in results and efficiency.

1.4 Materials and Methods

All calculations of this project were carried out on a Dell Inspiron 15 7000 with Intel Core i7-8550U processor at 1.80 GHz and 8 GB of RAM. This must be taken into consideration as statistics will be shown regarding the timing of the resolution of the statistical tests according to the algorithms, but obviously the computational power of the processor must be taken into consideration, both positively and negatively. The programming tools used are:

- RStudio 3.6.1 to use the programming language `R`
- Code::Blocks 17.12 for programming in `C`
- Visual Studio 2017 for programming in `C++`

As for the statistical context being analyzed, `R` is the best programming language that allows you to quickly make calculations and extrapolate significant graphs that summarize the theoretical aspects. The `HardyWeinberg` package, described by Graffelman (2015), contains most of the functions useful for carrying out the examples that will be shown, while the `HWxtest` package, owned by Engels (2009) and no longer located on the CRAN (The Comprehensive R Archive Network), has allowed to take a cue from some functions written in `C` for the most complicated calculations. In particular, the latter package calls functions written in `C` directly on `R` and, by passing parameters, is able to receive values backwards. The principle adopted here is similar and, using the `Rcpp` library (François and Eddelbuettel, 2011), it is quite simple to compile `C++` code in `R`.
The reason why we try to use more than a single language is that, while `R` is more useful, as mentioned, to show graphs and results of calculations that use vectors and matrices and that can be brought in parallel, the languages `C` and `C++` allow you to write better recursive algorithms that, if efficient, speed up your work.

Chapter 2

Statistical methods

Being aware of the importance of HWE, it is important to understand which methods have been developed and used over the years to test genetic variants for equilibrium. In fact, rather than testing *equilibrium* as a biological concept, *Statistical genetics* has taken care of testing the Hardy-Weinberg proportions (HWP). Generally speaking, it is possible to divide the tests that we have to date into two subgroups.
On the one hand, methods that work well with very large samples and lead to asymptotically acceptable results. Among them is the Pearson χ^2 test or the likelihood ratio test; the latter will not be dealt with.
On the other hand, when some frequencies in the contingency table are small, the Exact test (Levene, 1949; Haldane, 1954; Chapco, 1976) is used, which, in addition to working well with these quantities is also more accurate (Guo and Thompson, 1992). A limitation of the latter is certainly the high computational effort required to calculate it. In fact, if the genotype counts become larger, a complete enumeration of the contingency tables is avoided (useful, as you will see, for the exact test), going instead to use the permutation test that exploits sampling of the exact distribution.

This chapter will shed light on the methods currently known and used for HWP; in particular, the analysis takes into account statistical tests from different points of view. The first is the **number of alleles** you consider: it is quite easy to think about how the complexity of the problem can change depending on the number of factors you consider. The second is the type of **chromosome**: autosomes are the most common and those for which the literature is wider, having treated all aspects and all chromosomes, in particular the X chromosome, the latter often being bypassed, as explained above.

2.1 Two alleles

In this section we start with the **bi-allelic** case, which is the basic one to fully understand the statistical study for HWE. Being the simplest and most common case, it is studied in depth in the literature in the two subcases which are discussed below: the first concerns the autosomal context, while the second considers all the chromosomes.

2.1.1 Autosomes

The study of autosomes is relatively simple, as there is no specific case to take into account depending on the sex, as happens for the X chromosome. Therefore there are several tests to verify a deviation from the HWE that have been analyzed in detail in the years. We limit ourselves to seeing only a few of them, which are among those most commonly used and which are then recalled later to describe the heart of this project. The tests explained below are: *chi-square test*, *permutation test* and *exact test*.

Chi-square test

The *chi-square* test includes several possible statistical tests and is based on the fact that the test statistic follows the chi-square distribution under the **null hypothesis**. The test we are talking about here is that of Pearson, which is aimed at verifying whether there is a substantial difference between the frequencies observed and those expected in the contingency table of genotype counts. The purpose of the test is to assess the goodness-of-fit of the multinomial distribution with a given vector of probabilities.

In the case of HWE, the goal is to verify that the observed genotype frequencies are close to what is expected under the Hardy-Weinberg assumption. Using the chi square test, and crossing the χ^2 statistic with the degrees of freedom, it's possible to obtain "the probability that the observed numbers would deviate from the expected numbers as much or more by chance (see Figure 2.1). The degrees of freedom used to determine the significance of χ^2 value are equal to the number of phenotypic classes, k, minus one, and then minus the numbers of parameters estimated from the data" (Hedrick, 2011).

In general, if there are k genotypes and O_i and E_i are respectively the observed and expected frequencies of the i-th genotype, the chi-square statistic is:

$$X^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \tag{2.1}$$

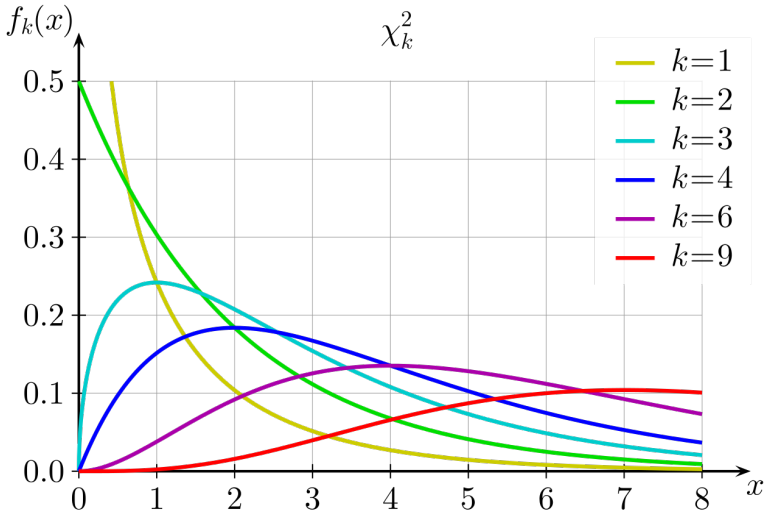


Figure 2.1: Chi-square distribution with different degrees of freedom (from Wikipedia)

More precisely, considering autosomes with two alleles A and a, there are on one side the genotype counts n_{AA}, n_{Aa} and n_{aa} (see Table 2.1), while on the other those expected in the context of HWE, which are e_{AA}, e_{Aa} and e_{aa}.
If there are n individuals, then there are $2n$ alleles and the frequency of allele A is supposed to be p, while that of allele a is q ($p + q = 1$) and the expected counts are: $e_{AA} = np^2$, $e_{Aa} = 2npq$ and $e_{aa} = nq^2$.

	A	a	
A	n_{AA}	$\frac{1}{2}n_{Aa}$	$\frac{1}{2}n_A$
a	$\frac{1}{2}n_{Aa}$	n_{aa}	$\frac{1}{2}n_a$
	$\frac{1}{2}n_A$	$\frac{1}{2}n_a$	n

Table 2.1: The three genotype counts

So, the chi-square statistic, to be compared with a χ^2_1 distribution (with 1 degree of freedom) is:

$$X^2 = \frac{(n_{AA} - e_{AA})^2}{e_{AA}} + \frac{(n_{Aa} - e_{Aa})^2}{e_{Aa}} + \frac{(n_{aa} - e_{aa})^2}{e_{aa}} \tag{2.2}$$

Unfortunately, only with this kind of test it's impossible to know if the reason for

rejecting an HWE is for an excess or a lack of heterozygotes. In this sense there is a further way to calculate the chi-square statistic which is the following:

$$X^2 = \frac{D^2}{p^2(1-p)^2 n} \tag{2.3}$$

where $D = \frac{1}{2}(n_{Aa} - e_{Aa})$ and indicates the deviation from independence for the heterozygote (Graffelman, 2015). The sign of D reveals whether the sample is characterised by an excess or a lack of heterozygotes.
Since we are trying to approximate a discrete distribution with a continuous one, when the expected count e_i is small, it is possible to modify this test by adding the so-called *continuity correction* c, which is usually equal to 0.5. In this way, if we have k genotypes, the new chi-square test result will be given by

$$X_c^2 = \sum_{i=1}^{k} \frac{(|n_i - e_i| - c)^2}{e_i} \tag{2.4}$$

Permutation test

Often, relying on an asymptotic solution, such as the chi-square test, can lead to unreliable results. For this reason, a new method is now being introduced, the *permutation test*, that was first proposed by Guo and Thompson (1992), which allows you to get better results, although with a very large computational effort. As explained above, if n individuals are considered in the analysis, there will be $2n$ alleles. First we calculate a test statistic, as can be the the X^2 or X_c^2, for the observed data and that will be used as a threshold. After that, the $2n$ alleles are listed sequentially (e.g. AaAaaaAAa) which will be permuted a certain number N of times. For each permutation, the genotypes will be considered as pairs of consecutive alleles in the sequence and a statistic will be calculated also here. This process is repeated N times and the number of statistics is counted which is greater than the statistic first calculated for the observed data. From the division between this number C and the number of permutations N, the p-value of the permutation test is obtained as C/N (Graffelman, 2015).

Exact test

Levene et al. (1949) described the conditional distribution of the number of heterozygotes of a sample taken from a population in equilibrium of Hardy-Weinberg, for an arbitrary number of alleles. The distribution depends on the number of each allele in the sample, and it is very similar to the fixed margins in Fisher's exact 2 x 2 distribution. This is the distribution used here to develop an *exact Hardy-Weinberg test* (Louis and Dempster, 1987). Subsequently, several authors have given a hand in improving this study, both from the point of view of the

efficiency of calculation and to achieve reliable results, and from the point of view of the complexity of the problem, which is presented here in the biallelic case. This method is the central point of all the work done in this thesis, is the starting point from which it is taken cue to extract a quick and reliable algorithm to solve situations neglected until today. The conservative approach that is used starts from considering n_{AA}, n_{Aa} and n_{aa} the numbers of the individuals AA, Aa and aa in the sample, respectively, and p_A and p_a the population frequencies of the alleles A and a. Hence, the multinomial probability of obtaining the sample (n_{AA}, n_{Aa}, n_{aa}) from a population in HardyWeinberg equilibrium is

$$P(N_{AA} = n_{AA}, N_{Aa} = n_{Aa}, N_{aa} = n_{aa}) = \frac{n!}{n_{AA}!n_{Aa}!n_{aa}!}(p_A^2)^{n_{AA}}(2p_Ap_a)^{n_{Aa}}(p_a^2)^{n_{aa}} \tag{2.5}$$

Considering now $n_A = 2n_{AA}+n_{Aa}$ and $n_a = 2n_{aa}+n_{Aa}$, the probability of obtaining the sample (n_A, n_a) is given by the binomial distribution

$$P(N_A = n_A, N_a = n_a) = \frac{2n!}{n_A!n_a!}(p_A)^{n_A}(p_a)^{n_a} \tag{2.6}$$

Finally, it's possible to obtain the real probability we're interested in: the conditional distribution of the number of heterozygotes (N_{Aa}) given the minor allele count (N_A)

$$\begin{aligned} P(N_{AA}, N_{Aa}, N_{aa}|n_A, n_a) = & \\ &= \frac{n_A!n_a!n!2^{n_{Aa}}}{\frac{1}{2}(n_A - n_{Aa})!n_{Aa}!\frac{1}{2}(n_a - n_{Aa})!(2n)!} = \\ &= \frac{n_A!n_a!n!2^{n_{Aa}}}{n_{AA}!n_{Aa}!n_{aa}!(2n)!} \end{aligned} \tag{2.7}$$

"The standard way to compute the p-value of an exact test is to sum probabilities according to Equation (2.7) for all samples that are as likely or less likely than the observed sample. This way to compute the p-value has been termed the SELOME p-value (select equally likely or more extreme samples)" (Graffelman, 2015).
The difference between the exact test for HWE and that of Fisher mentioned above, concerns only the frequencies that can be observed. Fisher's test sets the extremes in the contingency table, so as to have any freedom of values in the allele counts, in order to respect the margins. With this type of test, however, considering A the allele with the lowest count, then n_{Aa} will only take even values if n_A is even; they will be odd values in the opposite case.
As explained in Wigginton et al. (2005) paper, the higher computational effort

depends on the value of n_{Aa} and the factorial calculation that derives from it. The authors summarize a small algorithm, taking up what Guo and Thompson (1992) started, with which the number of multiplications and divisions is fixed and depends on the total number of alleles and not on the heterozygotes. This type of approach, called *naive*, helps a lot the software that must use these formulas for statistical tests to give faster answers and to avoid too high recursion levels, which can fault the calculators.

2.1.2 X chromosome

The approach that is usually used to test HWE for X-chromosomal marker sees the exclusion of males and the consideration of females only. This is because the males, inheriting only one X chromosome from the mother, are *hemizygous*. Considering them too involves a much higher computational cost which is often avoided. This leads to fundamental errors in the HWP test: not considering the males, half of the samples are lost; moreover, in doing so, it is assumed that the allelic frequency of the two sexes is equivalent, which might, a priori, not be true (Graffelman and Weir, 2016). Therefore, the same tests considered for autosomes will now be listed, correctly adapted to the X chromosomal context for the biallelic case.

Chi-square test

The basic principle of the chi-square test is the same as in the case of the autosomes. This time an additional factor comes into play, θ, which indicates the fraction of males in the entire population considered. Going in order, if there are two alleles A and a, p_A is the frequency of the first and (1-p_A) that of the second. The number of males with the alleles A, m_A, and a, m_a and the number of females having the three possible genotypes f_{AA}, f_{Aa} and f_{aa} must be taken into account. Now, if n_m and n_f are the total number of males and females in the population, $n = n_m + n_f$ is the total number of the sample, while $n_t = 2n_f + n_m$ is the total number of alleles present. Having this notational framework available, the previously defined θ factor will be equal to

$$\theta = \frac{n_m}{n}$$

Furthermore, the expected allele frequency of A (indicated with $\hat{p}_A$) will be the number of A allele on the total number of them

$$\hat{p}_A = \frac{n_A}{n_t}$$

In the Table 2.2 it's possible to see the different observed and expected counts for the two male genotypes and the three female genotypes. In the first line there is

the actual probability of the five categories in this analysis, while in the other two lines we give expected and observed counts for a chi-square statistic.

Genotype	Males		Females		
	A	a	AA	Aa	aa
Probability	θp_A	$\theta(1-p_A)$	$(1-\theta)p_A^2$	$2(1-\theta)p_A(1-p_A)$	$(1-\theta)(1-p_A)^2$
Observed	m_A	m_a	f_{AA}	f_{Aa}	f_{aa}
Expected	$n\theta\hat{p}_A$	$n\theta(1-\hat{p}_A)$	$n(1-\theta)\hat{p}_A^2$	$2n(1-\theta)\hat{p}_A(1-\hat{p}_A)$	$n(1-\theta)(1-\hat{p}_A)^2$

Table 2.2: Observed and expected genotype counts for an X-chromosomal marker under HWE

In particular, following the (2.1), if e_i is the expected count of genotype i, the statistic for goodness of fit will be

$$X^2 = \frac{(m_A - e_A)^2}{e_A} + \frac{(m_a - e_a)^2}{e_a} + \frac{(f_{AA} - e_{AA})^2}{e_{AA}} + \frac{(f_{Aa} - e_{Aa})^2}{e_{Aa}} + \frac{(f_{aa} - e_{aa})^2}{e_{aa}} \quad (2.8)$$

Normally the proportion of males, θ, is not known, therefore the distribution following the statistics is that of a chi-square statistic with 2 degrees of freedom. This type of test starts from considering two factors as null hypothesis: the homogeneity of allele frequency between the two sexes and that the females follow the HWP. When one or both of them fails, the test result will lead to the rejection of the null hypothesis (Graffelman and Weir, 2016). This concept is very well explained in the Figure 2.2, which is taken from the paper *Testing for Hardy–Weinberg equilibrium at biallelic genetic markers on the X chromosome* of Graffelman and Weir (2016). It shows a ternary diagram (also called **de Finetti diagram** thanks to the Italian statistician Bruno de Finetti) for four different situations with male and female genotype and allele frequencies for an X-linked marker in HWE:

a) females in HWE and equal allele frequencies for males and females

b) females not in HWE, but the same allele frequencies for the two sexes

c) females in HWP, but males and females with different allele frequencies

d) neither condition is verified

Permutation test

The *modus operandi* of the permutation test between autosomes and X chromosome is the same. This time, after choosing a statistical test, the $n_t = n_m + 2n_f$ alleles present are placed in sequence and mixed several times. The first n_m elements are

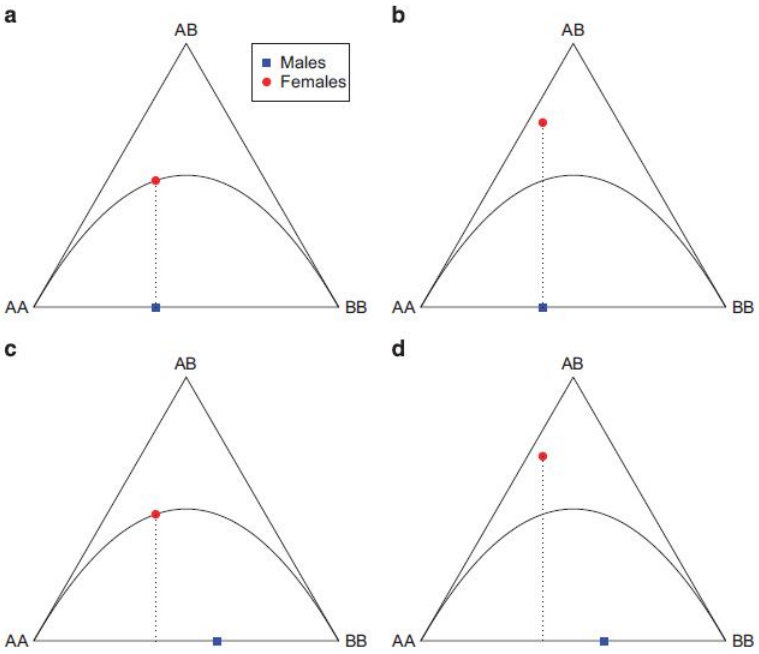


Figure 2.2: Hardy-Weinberg disequilibrium for a biallelic marker on X chromosome

considered males, while the remaining $2n_f$ are females and seen in pairs, in order to calculate the genotype counts. For each shuffle the statistical test is calculated and, finally, the p-value is the division between the number of times the result was greater than or equal to the threshold and the total number of permutations. This permutation test conditions on the observed allele frequency and also on the observed gender ratio (Graffelman and Weir, 2016).

Exact test

The equation (2.7) is just a special case of the real formula of the exact test, in which it's considered also the sex of the individuals. Following the same notation of the chi-square test explained above, in which the single X chromosome is considered in males and the double in females, we can express the probability of the intersection of distribution of the allele A in males and of heterozygosity in females, starting from having already fixed the sample size, the number of alleles A (n_A) and the number of males in the population (n_m).

$$P(M_A = m_A \cap F_{Aa} = f_{Aa} | n, n_A, n_m) = \frac{n_A! n_a! n_m! n_f!}{m_A! m_a! f_{AA}! f_{Aa}! f_{aa}! n_t!} 2^{f_{Aa}} \tag{2.9}$$

If males are excluded from the calculation, then $m_A = 0$, $m_a = 0$ and $n_m = 0$, the equation (2.9) is reduced to (2.7). The resulting p-value comes out, also in this case, adding all the results less than or equal to that of the observed sample (Graffelman and Weir, 2016). This type of test, applied to the X chromosome, is the latest type available in the various `R` packages, in particular in `HardyWeinberg`, and which can therefore be performed by a computer. All the calculations behind this test take a long time to be processed with the algorithms that are available today. The bi-allelic case is still feasible following the formula just explained; when more alleles are considered, as will be seen from the next paragraph, the exact test can only be calculated in practice for autosomes, although the theory underlying the sex chromosome is also available and clear.

2.2 *k* alleles

This paragraph is set up in a slightly different way from the previous one, despite the natural continuation. The reason is that for a large number of alleles some tests are not used in one or the other context and it is therefore useless to give a detailed description as done before. The only test worth making a clear separation is the exact test that will be explained last.

2.2.1 Chi-square test

In general, the chi-square test is not used in the case of multiple alleles, since in this situation most likely some alleles are rare. This involves **low counts** which, as mentioned, are not suitable for the chi square test and, in general, with asymptotic statistics. For this reason, the exact test and the permutation test are more suitable in this context. Nonetheless, we want to give a brief description of how the chi-square test can still be used in the multi-allelic case, taking into account the reliability of the results that come out according to what just said. The situation with two alleles has been described in paragraph 2.1: if the autosomes are considered, the distribution of the chi-square to test the HWE has only *one* degree of freedom (χ^2_1); on the other hand, if the X chromosome is taken into account, the chi-square test has *two* degrees of freedom (χ^2_2).

For the multi-allelic case, the chi-square statistics follow the same *modus operandi*. In particular, if there are k alleles, there will be a number of genotypes equal to

$$\frac{1}{2}k(k+1)$$

Considering $a_1, ...a_k$ the k alleles, we represent by n_{ij} the count of genotype $a_i a_j$ observed, whereas e_{ij} represents the expected count. Therefore, the resulting statistic is

$$X^2 = \sum_{i \geq j} \frac{(n_{ij} - e_{ij})^2}{e_{ij}} \tag{2.10}$$

This time, the degrees of freedom for this test are $\frac{1}{2}k(k-1)$. Also for this typology it is possible to add the correction factor c, which was introduced by Yates in 1935 (Emigh, 1980), and the statistic becomes

$$X^2_c = \sum_{i \geq j} \frac{(|n_{ij} - e_{ij}| - c)^2}{e_{ij}} \tag{2.11}$$

where, usually, $c = 0.5$.

2.2.2 Exact test

As previously introduced, the distribution for an arbitrary number of alleles into a population in HWE was introduced by Levene et al. (1949). In case the number of alleles exceeds 2, both for the autosomes and for the X chromosome, we apply the exact test that is treated in this project and for which an efficient resolution algorithm is given. The theory and the figures that will be used now are inspired by the paper of Graffelman and Weir (2018), which summarizes the main concepts of this topic in a concise but very clear way.

Autosomes

In the same way of the previous section, instead of considering as two standard alleles "A" and "a", the k alleles are considered as $a_1, a_2, ..., a_k$ and n_i is the total count of the i-th allele ($i = 1, 2, ..., k$). In the context of autosomes, we have n_{ij}, with $i \geq j$, which indicates the total number of heterozygous $a_i a_j$ or n_{ii} which indicates the number of homozygous $a_i a_i$, including both male and female sex. Furthermore, n represents the total number of individuals and $n_t = 2n$ the total number of alleles. The classic way of representing this situation is the lower triangular matrix in the Table 2.3.

a_1	n_{11}			
a_2	n_{21}	n_{22}		
$\vdots$	$\vdots$	$\vdots$	$\ddots$	
a_k	n_{k1}	n_{k2}	$\cdots$	n_{kk}
	a_1	a_2	$\cdots$	a_k

Table 2.3: Lower triangular matrix for autosomal genotype counts

Exact inference for autosomal variants with multiple alleles is based on the conditional distribution of the genotype counts, considering all observed allele counts as given and is obtainable by the following formula, which takes over the previous one (2.7)

$$P(N_{ij} = n_{ij} | n_1, ..., n_k) = \frac{n! 2^h \prod_{i=1}^{k} n_i!}{(2n)! \prod_{i \geq j} n_{ij}!} \tag{2.12}$$

where $h = \sum_{i>j} n_{ij}$ is the total heterozygote frequency.

X-Chromosome

For the X chromosome, gender must also be taken into account. Male individuals are hemizygous and therefore they have only one allele, so n_{mi} is the number of

males with the genotype a_i, while for females n_{fij} will indicate the number of female genotypes $a_i a_j$. If n_m and n_f are the number of males and females respectively, the total sample size will obviously be $n = n_m + n_f$ and the total number of alleles is $n_t = n_m + 2n_f$. All of this can be summarized in the Table 2.4.

a_1	n_{m1}
a_2	n_{m2}
$\vdots$	$\vdots$
a_k	n_{mk}

a_1	n_{f11}			
a_2	n_{f21}	n_{f22}		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
a_k	n_{fk1}	n_{fk2}	$\ddots$	n_{fkk}
	a_1	a_2	$\cdots$	a_k

Table 2.4: Lower triangular matrix for female X-chromosomal genotype counts and column vector with male genotype counts

Under the HWE hypothesis for the females and equality of allele frequencies in the sexes, i.e. the same ones seen for the bi-allelic case, the distribution of the genotype counts is given by the following equation

$$P(N_{fij} = n_{fij} \cap N_{mi} = n_{mi}) = \frac{n_m! n_f! 2^h \prod_{i=1}^{k} p_i^{n_i}}{\prod_{i=1}^{k} n_{mi}! \prod_{i \geq j} n_{fij}!} \tag{2.13}$$

where $h = \sum_{i>j} n_{fij}$ is the total female heterozygote count and p_i is the frequency of the i-th allele. Under the same conditions, the N_i allele counts follow the multinomial distribution

$$P(N_i = n_i) = \frac{n_t!}{\prod_{i=1}^{k} n_i!} \prod_{i=1}^{k} p_i^{n_i} \tag{2.14}$$

The conditional distribution of the genotype counts given the allele counts is obtained by dividing the previous two equations.

$$P(N_{fij} = n_{fij} \cap N_{mi} = n_{mi} | n_1, ..., n_k) = \frac{n_m! n_f! 2^h \prod_{i=1}^{k} n_i!}{n_t! \prod_{i=1}^{k} n_{mi}! \prod_{i \geq j} n_{fij}!} \tag{2.15}$$

In both situations, the goal is to extract the p-value of the test, that is important to understand if the considered sample is significant or not under HWE hypothesis. Clearly, the tables 2.3 (autosomes) and 2.4 (X chromosome), refer only to the observed sample. Based on the counting vector of the total alleles, the number of obtainable tables of the same type can grow dramatically. Theoretically, to have the final p-value, the probability must be calculated for each table and, finally, all the probabilities less or equal to the statistical test of the initial table must be added, as described in the previous chapter. Just to give an idea of the quantities we are

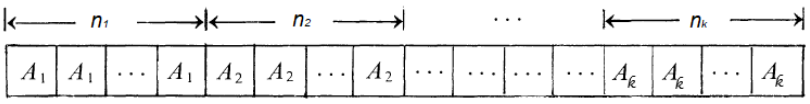
talking about, it's possible to take inspiration from an example of the paper written by Engels (2009), which will be very useful in a few chapters. If the observed sample, with 4 alleles, has the following matrix

$$\mathbf{a} = \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

it is possible to indicate the vector of the count allele with $\mathbf{m} = [\ 2 \quad 2 \quad 2 \quad 2\ ]$. From these data 17 different tables can be built, including the one above, and for each table you have to calculate the corresponding probabilities; then it's necessary to perform the calculations explained above multiple times, and this will take a lot of computer time, given all the factorials present in the formulas.
The goal of this project is to calculate Equation (2.15) in the most efficient way, in particular for large samples for which there are many outcomes. Ideally, we want to be able to carry out an X-chromosomal exact test for HWE for any sample size, any number of alleles, and any observed allele and genotype counts.

2.2.3 Permutation test (Monte Carlo Method)

The explanation of the permutation test at the bottom is not accidental. The reference statistic that was previously calculated with one of the existing tests of choice, which could be the chi square test, this time must be carried out with the exact test just explained, following the formula (2.12) or (2.15).
The procedure with which this test is carried out generally follows the previous cases with two alleles, only this time there are larger groupings before extracting the pairs. We will follow the explanation already given by Guo and Thompson (1992) which describes the different steps to follow quite well in the third section. For convenience, this time the k alleles will be indicated with a capital letter A_i. Hence, if we have a population of n individuals, who have n_i A_i type alleles $(i = 1, ...k)$, which can be arranged in the following way.



After a random permutation is applied, the n successive pairs of alleles can be considered as a sample of size n drawn from a population under HWP, with n_i alleles of type A_i $(i = 1, ..., k)$. If we repeat another permutation and pair the alleles again into n pairs, then the n pairs can be considered as another random sample

from the same population. The algorithm can be summarized in the following points:

1. Compute the probability `P` following (2.12); set counter $C = 0$
2. Construct index vector s, with $s_1 = s_2 = \cdots = s_{n_1} = 1, \cdots,$ $s_{n_1+n_2+n_{k-1}+1} = \cdots = s_{n_k} = k$
3. For specified simulation size N do the following
   a) Reset s to a random permutation of the previous one
   b) Obtain sample $\mathbf{g}$ by cutting s into n consecutive pairs and letting $\mathbf{g} = \{A_{s_{2c-1}}A_{s_{2c}},\, c = 1, ..., n\}$
   c) If the probability of $\mathbf{g}$ is lower or equal than `P`, add 1 to C
4. The final p-value is equal to $\frac{C}{N}$

By changing some small things, for instance by using (2.15) instead of (2.12) it's possible to obtain a result for the case of the X-chromosome. This test can be carried out in `R` and it is, so far, the only means of obtaining results for samples with a large number of alleles and the X chromosome with currently available computing power.

Chapter 3

Examples with a single polymorphism

Chapter three will allow you to better understand the different theoretical concepts explained in the previous one. Real case studies will be taken in order to carry out practical and, at the same time, meaningful numerical examples. This is of fundamental importance to understand that the work carried out in this project and the one already available, which concern HWE, are not only descriptive words but have a significant practical implication in various fields. For each of the four macro-categories that are being considered, i.e. autosomes and X chromosome, each with 2 or with more alleles, one or more *polymorphisms*, Single Nucleotide Polymorphisms (SNPs) or microsatellites, also known as Short Tandem Repeats (STRs) will be taken, and some of the previously explained tests will be applied and it will be seen what results will come and what this means. To better understand what the data we will see means, let's give a couple of more specific definitions.
The main concepts behind everything are trait and marker. Some *traits*, as can be the state of a disease, depend on genetic variables. A *marker* is one of these, which shows a variation over several individuals. The causes that can influence traits are analyzed thanks to the interaction between these two concepts. If a marker contains only homozygotes then it is called *monomorphic*. Otherwise, "the term *polymorphism* is defined simply as a genetic variant at a single location within a gene. Technically, a variation must be present in at least 1% of a population to be classified as a polymorphism" (Foulkes, 2009). In general the terms polymorphism, variant or marker will be used as synonyms, although the latter is a superset that contains SNPs (Single Nucleotide Polymorphism), STRs (Short Tandem Repeat), RFLPs (Restriction Fragment Length Polymorphism), indels (insertion / deletion polymorphism), etc. In particular, with SNP we intend to observe the differences of *nucleotides* (which contain the nitrogen bases) between the different individuals of

a population. These can be factors of evolution over time, or they can be compared with diseases to make medical diagnoses, as seen in Figure 3.1.

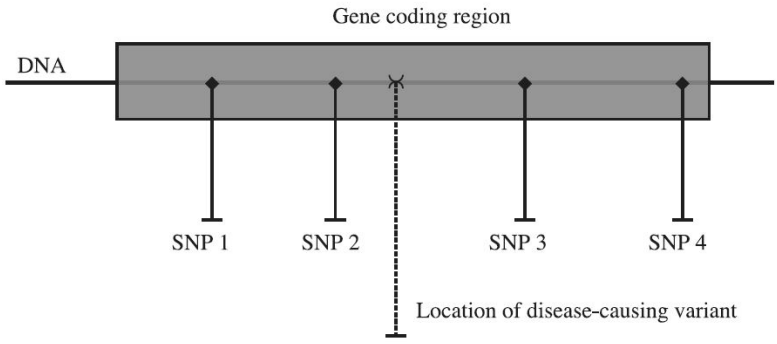


Figure 3.1: Marker SNPs (Foulkes, 2009)

DNA has, as already discussed, a double helix structure that requires that each chromosome has complementary nucleotide pairs in each position. This is clearly visible in the Figure 3.2, where there are two non-identical chromosomes in that they differ in a pair (homologous chromosomes). Obviously, as already mentioned in the first chapter, there are standards for reading DNA fragments in an unambiguous way (following the 5′ and 3′). SNPs are of enormous importance in modern genetic mapping, especially in relation to the low financial cost of genotyping in different places. They occur about once every 300 pairs for about 10 million SNPs in the human genome (Laird and Lange, 2010).

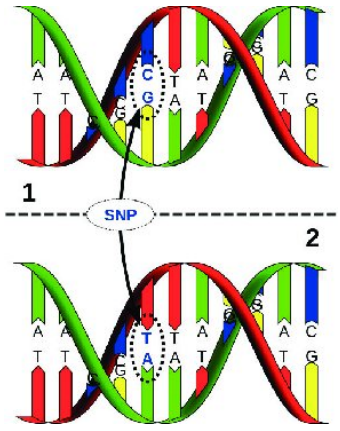


Figure 3.2: Single nucleotide polymorphism (SNP) with alleles C and T (David Hall, License: Creative Commons)

Therefore, within the following examples some polymorphisms belonging to

real databases will be chosen which aim to study different types of diseases by analyzing the SNPs of a certain different number of individuals. For the autosomes no distinction will be made between the sexes, it is a crucial element however, for the analysis of X chromosomal polymorphisms. The tools named in the Chapter 2 will be used. For some of them, in particular in the multi allelic analysis of the X chromosome, there will be a detailed explanation of the operation of the algorithm behind it in the following chapters.

3.1 Autosomes

Two alleles

In the context of the autosomes with **two** alleles, the chosen example to pursue concerns a database that was drawn up to study Alzheimer's disease and that was taken from the paper written by Takei et al. (2009). This data-set has also been included in the `HardyWeinberg` package (Graffelman and Camarena, 2008), so that it can be used as an example of how some functions work for statistical tests and the graphs that can be extracted from them. The study of HWE is preliminary to that of diseases, such as Alzheimer. The article just cited explains in detail how the study of the disease is carried out in a population of 1262 Japanese individuals. "The $\epsilon 4$ allele of APOE is a well-characterized genetic risk factor for late-onset Alzheimer disease (LOAD)": these are the first words of the abstract, in which it is clearly understood that the study of different *polymorphisms* helps to detect allelic mutations and, possibly, to have results to deepen the study of the disease or not. All this, however, must be preceded by the verification of the HWP in the treated sample. Confirmation of this is given by the final p-value which can give us an idea of the rejection or not of the null hypothesis. Let's see the example step by step. First of all, the `HardyWeinberg` package can be installed and loaded by using:

```
install.packages("HardyWeinberg")
library("HardyWeinberg")
vignette("HardyWeinberg")
```

This will make available a lot of functions, like `HWChisq, HWExact, HWPerm`, that are useful for our goals. The document describing the package can be consulted from inside `R` by typing the last command in the previous script. Then, we choose the polymorphism 'rs446037' to study its behavior. The alleles chosen in this example, for uniformity with the data source, will be named with 'M' and 'm'. Then, after taking the data from the database (with the `data` command) and showing which

row the chosen SNP corresponds to, the genotype counts are inserted in an `x` vector, which will be given as input to the various tests. Note that the set of polymorphisms is divided into *cases* and *controls*, but only the latter are used in the following commands.

```
data("Alzheimer")

Alzheimer[60,]

MM <- Alzheimer[60, 1]
Mm <- Alzheimer[60, 2]
mm <- Alzheimer[60, 3]

x <- c(MM, Mm, mm)
```

The output of the command `Alzheimer[60,]`, that is used just to print, is

```
                     MM Mm mm Group
 rs446037.controls 674 34  1     0
```

and it confirms the SNP we're analyzing (group 0 is for the controls).
Now one performs two chi-square test with the following functions: the first one is with the continuity correction factor, while in the second one, with the option `cc=0`, there is a normal test.

```
HWTest <- HWChisq(x, verbose = TRUE)

HWTest <- HWChisq(x, cc = 0, verbose = TRUE)
```

The two outputs

```
Chi-square test with continuity correction for Hardy-Weinberg
equilibrium (autosomal)
Chi2 =  0.01384079 DF =  1 p-value =  0.9063474 D =  -0.5430183
f =  0.03095353

Chi-square test for Hardy-Weinberg equilibrium (autosomal)
Chi2 =  0.6793078 DF =  1 p-value =  0.4098252 D =  -0.5430183
```

```
f =  0.03095353
```

show that there is a small statistic, 0.0138 in the first case and 0.6793 in the second one, with two p-values (0.906 and 0.4098) much larger than the usual value $\alpha = 0.05$ or, even better, $\alpha = 0.01$ as indicated in the paper where data come from. In order to have a complete view, `HWTest` is a list object containing the all the results of the test (statistic, p-value, half the deviation from HWE (`D`) for the heterozygote and the inbreeding coefficient `f`). The first function (with continuity correction) is not recommended for low minor allele frequencies, so the second one is more reliable. It is useful to also mention the minor allele frequency (MAF). It is used in statistical genetics studies because it provides an indication in order to differentiate between common and rare variants in the population. When we have two alleles, the MAF corresponds to the frequency of the least present one. If p_M and p_m are the frequencies of the alleles M and m respectively, then $MAF = min(p_M, p_m)$, which can be obtained as follows.

```
M <- 2*MM + Mm    # 1382
m <- Mm + 2*mm    # 36
total <- 2*sum(x)

MAF <- m/total
```

```
MAF = 0.02538787
```

The closer this value is to 0, the closer the marker is to being a monomorphism and the less frequent allele is rare.
Another possibility is the permutation test which is activated by

```
set.seed(123)
HWPermutationtest <- HWPerm(x, verbose = TRUE)
```

```
Permutation test for Hardy-Weinberg equilibrium
Observed statistic: 0.6793078   17000 permutations.
p-value: 0.3655882
```

The number of permutations can be setted with `nperm` argument. By default the chi-square statistic will be used as the test statistic, otherwise, another statistics can be used with the `FUN` argument. The result of the p-value is 0.3655, so it is similar to the previous one. Even if the data-set is not so big, this kind of test is very slow due to its intrinsic functioning, so it must be used carefully and when necessary, especially in the analysis of multiple SNPs. Finally, in order to perform the exact test the function `HWExact` accept the genotype counts vector as input.

```
HWExactest <- HWExact(x, verbose = TRUE)
```

```
Haldane Exact test for Hardy-Weinberg equilibrium (autosomal)
using SELOME p-value
sample counts: n =  674 n =  34 n =  1
H0: HWE (D==0), H1: D <> 0
D =  -0.5430183 p-value =  0.3660253
```

The latter is certainly the most relevant test and that gives us the most truthful result. The p-value of 0.366 definitely confirms that we are in the acceptance zone, so it is not possible to reject the null hypothesis and the HWE is confirmed. The resulting statistics can be calculated in different ways (`two-sided`, `greater` and `less`), as well as the p-value (`dost`, `selome` and `midp`) based on the parameters that are passed to the routine. Here we will not go into detail, but only the default ones ('two-sided' and 'SELOME') are used. In order to give an overview of the results, you can use the function `HWAlltests`.

```
HWResults <- HWAlltests(x, verbose = TRUE, include.
   permutation.test = TRUE)
```

```
                                          Statistic   p-value
Chi-square test:                         0.67930778 0.4098252
Chi-square test with continuity correction: 0.01384079 0.9063474
Likelihood-ratio test:                   0.51459407 0.4731569
Exact test with selome p-value:                  NA 0.3660253
Exact test with dost p-value:                    NA 0.7320506
Exact test with mid p-value:                     NA 0.2178784
Permutation test:                        0.67930778 0.3655882
```

k alleles

Another meaningful example that can be dealt with is the one taken from the article of Mölkänen et al. (2010). This time the context is the **tri-allelic** one, although autosomes are always being considered. We wanted to choose a relatively low k since, in doing so, the same package `R` of the previous example is still exploitable. If k becomes too large, other tools must be used to achieve results within a reasonable time, which will be explained later. The chosen SNP is taken from a study created to verify sepsis or organ dysfunction thanks to the C-reactive protein (CRP). In particular, to verify the results of the variation of the protein in different SNPs, a comparison was made with patients with *Staphylococcus aureus bacteremia*. The scientific details of the medical studies can be read in the paper cited above and are beyond the scope of the thesis, which focuses on statistical analysis. To ensure that the biological data on which an analysis is made are correct, so as not to lead eventually to altered results, the SNPs are subjected to statistical tests to test the HWE. Given and considered the complexity of the problem, as already discussed in Chapter 2, the tests that can be used and that can be trusted are less than in the previous case. Therefore, more in detail, the polymorphism 'rs3091244' will be used, the distribution of which has been studied in 116 patients. The Table 2 of the paper written by Mölkänen et al. (2010) shows the available data in detail. Here in Table 3.1 only what is necessary for the following analysis is shown.

SNP	Genotype distribution	n
rs3091244	Additive	
	TT	21
	TC	55
	TA	4
	CC	28
	CA	8
	A-minor allele recessive	

Table 3.1: Distribution of SNP rs3091244 genotypes in 116 patients

First of all, the `HardyWeinberg` library is included, so that you can use the available functions, in particular `HWTriExact` and `HWPerm.mult`. Subsequently, the data in the table are manually entered into the corresponding variables.

```
1 library(HardyWeinberg)
2
3 TT <- 21
4 TC <- 55
5 TA <- 4
```

```
CC <- 28
CA <- 8

x <- c(AA=0,AB=TA,AC=CA,BB=TT,BC=TC,CC=CC)
```

The function that follows accepts standard labels for input alleles, that are A, B and C. For this reason, since the alleles A, C and T are present in the polymorphism being analyzed, it is decided to consider the allele T as B. Therefore the data are inserted into a vector and the exact test is activated.

```
out <- HWTriExact(x)
```

```
Tri-allelic Exact test for HWE (autosomal).
Allele counts: A = 12 B = 101 C = 119
sum probabilities all outcomes 1
probability of the sample 0.01583807
p-value =  0.6872099
```

In this way we can obtain the analysis using the exact distribution with three alleles, which can be done for X-chromosome too by using the same function with different parameters. Then, to have a comparison from the point of view of the results, the permutation test is launched with the following commands.

```
x3 <- toTriangular(x)
out <- HWPerm.mult(x3)
```

```
Permutation test for Hardy-Weinberg equilibrium (autosomal).
3 alleles detected.
Observed statistic: 0.01583807   17000 permutations.
p-value: 0.6821765
```

The function `toTriangular` is used to convert a vector of genotypes into a triangular matrix, like in the Table 2.3, while `HWPerm.mult` approximates exact test

probabilities for joint tests for HWE and equality of allele frequencies for variants with multiple alleles (from `help` of the `R` package). The final p-values of the two tests are very similar each other. They are larger than 0.05 and it can be confirmed that there is no deviation from HWE in the genotype distributions for this SNP in the CRP gene.

3.2 X chromosome

Two alleles

The examples of markers on the X chromosome, especially for the **bi-allelic** case, are more difficult to find in the form with which the others have been described. In particular, more than SNP we speak of STR (Short Tandem Repeat) in this context, as we will see later with k alleles. This is probably due to the difficulty of studying HWE with this type of problem and the consequent few practical examples available are a demonstration of this. For now we will show two examples, in order to see what functions exist in `R` and how to use them.
The first example is purely demonstrative and is taken from the paper written by Graffelman and Weir (2016); the used data are summarized in the Table 3.2 and they are taken from 20 patients, 10 males and 10 females.

SNP	Genotype distribution	n
	Females	
	AA	0
	AB	3
	BB	7
	Males	
	A	3
	B	7
	A-minor allele recessive	

Table 3.2: Distribution of sample in 20 patients

This time, as hemizygous males (with only one X chromosome) are also being considered, there will be a division between the sexes and their genotype counts. First, as done in the previous examples, we include the `HardyWeinberg` library and declare variables corresponding to the alleles to be inserted in a vector. The alleles, in both examples we will see, will be called 'A' and 'B' to remain consistent with the notation used in the sources.

```
library(HardyWeinberg)

# males
A <- 3
B <- 7

# females
AA <- 0
AB <- 3
BB <- 7

x <- c(A=A, B=B, AA=AA, AB=AB, BB=BB)
```

In order to test the HWP with the available data, the three different tests explained in Chapter 2 are used. The chi-square test, the permutation test and the exact test are activated in R using the following functions. The first one is called without continuity factor and with the parameter `x.linked=TRUE`, in order to specify the X chromosomal context, instead of autosomal one.

```
chisqResult <- HWChisq(x,cc=0,x.linked=TRUE,verbose=TRUE)
```

```
Chi-square test for Hardy-Weinberg equilibrium (X-chromosomal)
Chi2 =  1.09375 DF = 2 p-value =  0.5787556 D =  NA
f =  -0.1764706
```

In the same way, for the permutation test it's specified the X chromosome; moreover, here with `nperm` the number of permutations can be set.

```
permResult <- HWPerm(x,nperm=10000,x.linked=TRUE,verbose=
   TRUE)
```

```
Permutation test for Hardy-Weinberg equilibrium of an X-linked
marker
Observed statistic: 1.09375   10000 permutations. p-value: 0.741
```

Finally, the way to call the exact test is similar to the other ones.

```
exactResult <- HWExact(x, x.linked = TRUE, verbose = TRUE)
```

```
Graffelman-Weir exact test for Hardy-Weinberg equilibrium on the
X-chromosome using SELOME p-value
Sample probability 0.1940129 p-value =  0.7453581
```

All tests give very high p-values as output, certainly greater than the 0.05 threshold, so it can be said that there is no deviation from the Hardy-Weinberg equilibrium. From the same article containing this example, the summary of what has just been said is given in the Table 3.3, thanks to small numbers it is feasible 'by hand'.

	m_A	m_B	f_{AA}	f_{AB}	f_{BB}	$Prob$
1	0	10	3	0	7	0.0002
2	0	10	2	2	6	0.0085
3	0	10	1	4	5	0.0340
4	0	10	0	6	4	0.0226
5	1	9	2	1	7	0.0121
6	1	9	1	3	6	0.1132
7	1	9	0	5	5	0.1358
8	2	8	2	0	8	0.0034
9	2	8	1	2	7	0.1091
10	2	8	0	4	6	0.2546
11	3	7	1	1	8	0.0364
12	3	7	0	3	7	0.1940
13	4	6	1	0	9	0.0035
14	4	6	0	2	8	0.0637
15	5	5	0	1	9	0.0085
16	6	4	0	0	10	0.0004

Table 3.3: All possible samples for a set of 20 individuals (10 males and 10 females) with a total of 6 A alleles

In summary, another example is given, a little more significant since taken from a real SNP and with slightly higher numbers. SNP 'rs5968922' is taken from an experiment on 1255 individuals, including 604 males and the remaining females, which is performed by Wellek and Ziegler (2019). The authors in the paper want to analyze the dynamics and difficulties that face us in the same context of this work. Looking at table 5 of the cited article, the data that are taken directly from the GENEVA project are shown here in Table 3.4.

SNP	Genotype distribution	n
rs5968922	Females	
	AA	275
	AB	296
	BB	80
	Males	
	A	392
	B	212
	B-minor allele recessive	

Table 3.4: Distribution of sample 'rs5968922' in 1255 patients (from GENEVA project)

Again, the names of the alleles are used in a standard way but could have been others without any difference. Without going too far, we report the same calculations of the previous example with their outputs.

```
# males
A <- 392
B <- 212
# females
AA <- 275
AB <- 296
BB <- 80

x <- c(A=A, B=B, AA=AA, AB=AB, BB=BB)

chisqResult <- HWChisq(x,cc=0,x.linked=TRUE,verbose=TRUE)
```

```
Chi-square test for Hardy-Weinberg equilibrium (X-chromosomal)
Chi2 =  0.00170001 DF = 2 p-value =  0.9991504 D =  NA
f =  0.0009953963
```

```
permResult <- HWPerm(x,nperm=10000,x.linked=TRUE,verbose=
   TRUE)
```

```
Permutation test for Hardy-Weinberg equilibrium of an X-linked
marker
Observed statistic: 0.00170001   10000 permutations. p-value: 1
```

```
exactResult <- HWExact(x, x.linked = TRUE, verbose = TRUE)
```

```
Graffelman-Weir exact test for Hardy-Weinberg equilibrium on the
X-chromosome using SELOME p-value
Sample probability 0.002818315 p-value =  1
```

This time, due to the large numbers, the table with all possible combinations is not shown. Anyway, it is quite clear from the result of all the tests that the p-value is close, if not equal, to 1 and it is a sign of a perfect HWE for the SNP 'rs5968922'.

k alleles

The most complex and complete context that can be shown is that which includes the X chromosome for a number of k alleles greater than 2. From here on out we will enter an "unexplored" world of which it is not possible, in most of the cases, to have comparisons with other methods for the truthfulness of the results obtained. The chosen example to explain is taken from the paper written by Chen et al. (2018), in which 19 X-chromosomal STRs are taken from different Chinese populations with the aim of analyzing forensic characterizations and exploring relationships within them. Also in this case the study goes into medical details, as well as statistics, which will not be seen in detail here and for which we refer you to the direct reading of the article in case you want to deepen.
Very clear and simple is the definition that is given of STR, which, as anticipated in the previous paragraph, is another type of marker, such as SNPs, which can be analyzed in the context of statistical genetics. "Short tandem repeats (STRs), also

known as *microsatellites* and composed of repeating $2-6$ base pair motifs, are highly variable variants with the number of approximately 700,000 in the human genome, play a pivotal role in population genetics, anthropology, genetic genealogy and forensics" (Chen et al., 2018). If, for instance, there is a tri-nucleotide STR, AGG, that is repeated 3 times in an individual's DNA sequence, it will be encoded as 3 (3 times) or 9 (3 times tri-nucleotide) depending on the chosen encoding. Furthermore, it is always to be taken into account that male individuals will inherit one while female ones will inherit two alleles. The number of repeated alleles will be analyzed in order to see those shared by different individuals to analyze their HWE. With these premises, let's see the example practically.
At first the libraries are included and the data are read (also available at the link `http://www.plosone.org/article/fetchSingleRepresentation.action?uri=info:doi/10.1371/journal.pone.0204286.s001`) of which, for the moment, the STR 'DXS7423' is chosen for analysis, which, as we shall see, contains 4 alleles $(13, 14, 15$ and $16)$. From here the males and females are split into two different structures and, for the latter, a concatenation is carried out between the two columns, so as to have data of the type "13/14", like before there was "A/B" or "AB". The Table 3.5 shows the first rows of the reference database, while the output of the two table statements gives us a count of the elements present. Overall, the experiment included 206 individuals including 102 males and 104 females.

	Sample identifier	DXS7423...4	DXS7423...5
1	ZunyiHan001	14	15
2	ZunyiHan002	14	14
3	ZunyiHan003	15	15
4	ZunyiHan004	15	15
5	ZunyiHan005	15	15
6	ZunyiHan006	14	14
7	ZunyiHan007	15	15
8	ZunyiHan008	15	15
9	ZunyiHan009	14	15
10	ZunyiHan010	15	15

Table 3.5: Distribution of sample 'DXS7423' in 206 patients

```
1 library(readxl)
2 library(HardyWeinberg)
3
```

```
dataSTR <- read_excel("../journal.pone.0204286.s001.xlsx",
   range = "A2:E208")
dataSTR <- dataSTR[,-(2:3)]    # removing unuseful STR

#splitting males and females
females <- dataSTR[1:104,2:3]
males <- dataSTR[105:206,2]

#concatenating father and mother X chromosomes
newFemales <- paste(females$DXS7423...5, females$DXS7423
   ...4, sep = "/")

table(newFemales)
table(males)
```

```
newFemales
14/13 14/14 15/14 15/15 16/14 16/15
    2     8    39    46     1     8

males
13 14 15 16
 1 33 64  4
```

After this, some functions are included which are useful in the calculations, such as the probability of the sample considered according to the formula (2.15), or some auxiliary functions that are used inside it. In particular, the `observedProb`, `matrix.to.vec`, `vec.to.matrix`, `alleleCounts`, `fillUpper` functions are declared and have been modified slightly compared to the Engels namesakes that are part of its `HWxtest` package.

```
fillUpper <-
  function(gmat){
    if(!(is.matrix(gmat) && (nrow(gmat)==ncol(gmat)))) stop(
   "Must be square matrix at least 2x2")
    k <- nrow(gmat);
    if(k<2) return(gmat)
    for(j in 2:k) {gmat[1:(j-1), j] <- gmat[j,1:(j-1)]};
    gmat
```

```
  }


alleleCounts <-
  function(gmat) {
    if(class(gmat)!="matrix") gmat <- vec.to.matrix(gmat)
    t <- fillUpper(gmat);
    k <- dim(t)[1];
    m <- integer(k);
    for(i in 1:k) {m[i] <- sum(t[i,]) + t[i,i]};
    if(!is.null(rownames(gmat))) names(m) <- rownames(gmat)
    m
  }


vec.to.matrix <-
  function(gvec, alleleNames=""){
    if(!(is.vector(gvec) && is.numeric(gvec))) stop("\nMust
   be a vector")
    nGenotypes <- length(gvec)
    nAlleles <- as.integer((sqrt(8*nGenotypes + 1) - 1)/2)
    if(nGenotypes != nAlleles*(nAlleles + 1)/2) stop("\
   nWrong number of genotype counts")
    t <- matrix(NA, nAlleles, nAlleles)
    for(i in 1:nAlleles){t[i, 1:i] <- gvec[(i*(i-1)/2 + 1):(
   i*(i+1)/2)]}
    if(length(alleleNames)>=nAlleles) {
      rownames(t) <- alleleNames;
      colnames(t) <- alleleNames;
    }
    t
  }

matrix.to.vec <-
  function(gmat){
    if(!(is.matrix(gmat) && (nrow(gmat)==ncol(gmat)))) stop(
   "Must be square matrix")
    v <- c();
    k <- nrow(gmat)
    for(i in 1:k){v <- append(v,gmat[i,1:i])}
    names(v) <- NULL;
    v
  }

observedProb <-
```

```
49   function(c, males, mf){
50     c <- fillUpper(c);
51     m <- alleleCounts(c);
52     d <- sum(diag(c));
53     n <- sum(m)/2;
54     nt <- mf[1] + 2*mf[2]
55
56     k <- n * log(2) + lgamma(mf[1]+1) + lgamma(mf[2] + 1) -
    lgamma(nt+1) + sum(lgamma(m+males+1));
57     a <- matrix.to.vec(c);  # only females
58     p <- exp(k - sum(lgamma(a+1)) - sum(lgamma(males+1)) - d
    *log(2));
59   }
```

A further step is done by completing the vector of the genotype count with the values at 0, so that we can use the functions listed above and calculate the probability of the considered STR.

```
1  vecFemale <- table(newFemales)
2  vecMale <- table(males)
3  mf <- c(sum(vecMale), sum(vecFemale))
4  vecFemale["13/13"] = 0
5  vecFemale["16/16"] = 0
6  vecFemale["15/13"] = 0
7  vecFemale["16/13"] = 0
8
9  x <- vecFemale[order(names(vecFemale))]
10 x
11
12 vecFemale <- as.vector(vecFemale[order(names(vecFemale))])
13 vecFemale <- vec.to.matrix(vecFemale)
14
15 nAlleles <- 4
16
17 tot13 <- 2*x["13/13"] + x["14/13"] + x["15/13"] + x["16/13"]
       + vecMale["13"]
18 tot14 <- x["14/13"] + 2*x["14/14"] + x["15/14"] + x["16/14"]
       + vecMale["14"]
19 tot15 <- x["15/13"] + x["15/14"] + 2*x["15/15"] + x["16/15"]
       + vecMale["15"]
20 tot16 <- x["16/13"] + x["16/14"] + x["16/15"] + 2*x["16/16"]
       + vecMale["16"]
```

```
21
22 prob <- observedProb(vecFemale, vecMale, mf)
23 prob
```

```
 13/13 14/13 14/14 15/13 15/14 15/15 16/13 16/14 16/15 16/16
     0     2     8     0    39    46     0     1     8     0

 Prob 1.751051e-05
```

The final step is to build a data structure, here called *counts*, which contains the complete set of individual allele counts and the observed probability. This structure is printed on a file that will be taken as input by an executable, `HWx.exe`, which was generated by the algorithm which will be discussed in Chapter 5. Then we build the string with the other input parameters that the executable needs, such as the number of alleles, the number of males and females and the probability considered and outputs the exact test p-value.

```
1  counts <- as.numeric(sort(c(tot13, tot14, tot15, tot16),
       decreasing = T))
2  counts <- c(counts, prob)
3
4  write.table(counts, file="counts.txt", row.names=FALSE, col.
       names=FALSE)
5  nInput = 1
6  str <- paste("HWx.exe", nAlleles, nInput, mf[1], mf[2], "
       counts.txt", "outpvalues.txt")
7
8  # call Program with Algorithm
9  system(str)
10
11 pvalsAlg <- read.table("outpvalues.txt")
```

```
 P-value
 0.509918
```

The final result reached is a p-value of 0.509, which is far greater than 0.05 and therefore the HWE is confirmed for this STR, confirming the premise for developing

the study in the article. The big difference with what has already been discussed by the authors is that they carry out the calculations to verify the HWE **only on the female individuals** that are available, reaching a p-value of 0.2654 for the same marker just studied, as is possible read in the paper or also in the summary table 2 at the bottom of it (`http://www.plosone.org/article/fetchSingleRepresentation.action?uri=info:doi/10.1371/journal.pone.0204286.s002`). Both results lead to the same conclusion that he does not reject the null hypothesis for Hardy-Weinberg proportions, but as it is clear, the numerical values are vastly **different**. So, given that it can sometimes happen that the conclusions change according to the consideration or not of the male individuals, the new algorithm for this type of verification is appropriate and very useful.

As previously mentioned, the more you go forward in the number of alleles considered, the more difficult it is to have, at least currently, tools to verify the results obtained. Up to 5 alleles a good index of truthfulness can be obtained with the permutation test of the `HardyWeinberg` package, for which the operation is illustrated. Since the function requires that alleles have names, we call the 4 alleles with letters from A to D and use the same numbers seen before.

```
x.m <- c(A=1, B=33, C=64, D=4)

x.f <- matrix(c(AA=0,AB=0,AC=0,AD=0,BA=2,BB=8,BC=0,BD=0,CA
   =0,CB=39,CC=46,CD=0,DA=0,DB=1,DC=8,DD=0), nrow = 4, ncol
   =4, byrow=T)

out <- HWPerm.mult(x.m,x.f)
```

```
Permutation test for Hardy-Weinberg equilibrium and equality of
allele frequencies (X-chromosomal).
4 alleles detected.
Observed statistic: 1.751051e-05   17000 permutations.
p-value: 0.5061176
```

The result (0.506) is very very similar to that previously obtained (0.509), so it is possible to confirm what has been said both numerically and as a resulting conclusion. The important thing that changes between the two methodologies is certainly the *computation time*, which is much higher in the permutation test and which will be well explained and described in the next chapters, in order to give feedback on the work done.

Chapter 4

Algorithm for the bi-allelic exact test

The test of the hypothesis that a population is in Hardy-Weinberg equilibrium has been studied, deepened and improved throughout the last century up to the present day. As already mentioned, deviations from HWE can indicate inbreeding, population stratification, and even problems in genotyping (Wigginton et al., 2005). For this reason, having useful and efficient tools has been increasingly necessary over time. All the methods discussed in theory have had several implementations in a practical context to achieve results and draw conclusions quickly, but in any case comparable to the original ones in terms of consistency and correctness. A summary of the historical path has already been seen in the introductory chapter. Here we now want to focus exclusively on the **exact test** and its implementation in the different algorithms available that can be used. Most of the topics that will be explained here are an in-depth analysis of what has been seen, only superficially, up to now with the examples. What's inside those functions that lead us to such clear results? Are they simple or complex? How long does it take to reach the output values? These are some of the questions we will try to answer. In particular, only the **bi-allelic** context is exposed here, dividing the algorithms for the *autosomes* and for the *X-chromosome*. Finally, an analysis will be made in terms of speed of the different solutions available, so as to have a focus on the main reasons for applying one solution rather than the other.

4.1 Autosomes

In both types of chromosomes, the possible algorithmic implementations for the exact test will be the same. The *complete enumeration*, *sampling* (permutation test) and finally a *network* algorithm will be exposed, of which the last will be the

absolute novelty in the next chapter for k alleles.

4.1.1 Complete enumeration

The complete enumeration algorithm is among the first and simplest developed so far. The moment you are faced with this statistical problem, this is the most intuitive method that comes to mind. The summary is as follows: list all the possible tables that can be generated, keeping the margins fixed, calculate their probabilities and add all those smaller or equal to the sample we are dealing with. Let's take a closer look at each step. The explanation will follow in practice the code of the `HWExact` function, inside the `HardyWeinberg` package of `R`; for a more theoretical treatment, instead, we take inspiration from the paper written by Louis and Dempster (1987) which clarifies the operation step by step, both for two and for more alleles. Suppose we have a sample of diploid individuals with a number of AA, AB and BB genotypes whose HWE hypothesis we want to test. By keeping the number of alleles A and B fixed, the n fixed samples can be considered, having the Fisher margins fixed and constant. This means that, if n_A and n_B are the number of alleles A and B, represented in Table 4.1 , the sum along the rows and columns of all the possible tables generated with these data will remain the same. It should be noted that, also from the theoretical point of view from now on, to use the same notation of the package, alleles 'A' and 'B' will be used instead of 'A' and 'a' seen in the previous theoretical explanation.

	A	B	
A	$2n_{AA}$	n_{AB}	n_A
B	n_{AB}	$2n_{BB}$	n_B
	n_A	n_B	$2n$

Table 4.1: Number of alleles and genotypes

From the code point of view, the first operation is to store the data and check its structure, in particular if a three-element vector has been introduced and if they are all positive integers. The homozygous and heterozygous genotypes are then divided into two vectors.

```
1 if (length(X) != 3 | any(X < 0))
2     stop("HWExact: X is not a 3 by 1 non-negative count
  vector")
3 if (any(!is.wholenumber(X))) {
4     warning("Genotype counts are not integers, counts will
  be rounded.")
```

```
    X <- round(X, digits = 0)
}
n <- sum(X)
Xhom <- X[homozyg(X)]
Xhet <- X[heterozyg(X)]
nA <- 2 * Xhom[1] + Xhet
nB <- 2 * n - nA
```

If n_{AA}, n_{AB} and n_{BB} are the numbers of individuals with the genotypes AA, AB and BB, respectively, and p_A and p_B are the population frequencies of the alleles A and B, it is possible to express the multinomial probability of obtaining the sample (n_{AA}, n_{AB}, n_{BB}) if there is Hardy-Weinberg equilibrium, such as

$$P(N_{AA} = n_{AA}, N_{AB} = n_{AB}, N_{BB} = n_{BB}) = \frac{n!(2p_Ap_B)^{n_{AB}}}{n_{AA}!n_{AB}!n_{BB}!}(p_A^2)^{n_{AA}}(p_B^2)^{n_{BB}} \quad (4.1)$$

Considering now $n_A = 2n_{AA} + n_{AB}$ and $n_B = 2n_{BB} + n_{AB}$, it's possible to obtain the real probability we're interested in. The conditional distribution of the number of heterozygotes (N_{AB}) given the minor allele count was derived by Levene et al. (1949) and Haldane (1954) and it's given by

$$P(N_{AA}, N_{AB}, N_{BB}|n_A, n_B) = \frac{n_A!n_B!n!2^{n_{AB}}}{n_{AA}!n_{AB}!n_{BB}!(2n)!} \quad (4.2)$$

To obtain the **final statistics** we need, let's calculate the cumulative probability, that is obtained by summing all probabilities less than or equal to the probability of the observed sample ("Select Equally Likely Or More Extreme" or SELOME). The crucial point is to be able to obtain all the probabilities and, therefore, to build the different samples. Considering n_A and n_B the fixed number of alleles, the table can be made from the different sets of 3 elements, let's call them (s_{AA}, s_{AB}, s_{BB}). We start with $(0, n_A, (n_B - n_A)/2)$ if $n_A \leq n_B$ or $((n_A - n_B)/2, n_B, 0)$ in the opposite case. Going forward, subtract 2 from the middle element (heterozygous) and add 1 to each of the two extremes (homozygous) s_{AA} and s_{BB}. Continue until s_{AB} is 1 or 0. In this way you create a table like the 4.2 of which you can calculate the probabilities of each row by following the formula (4.2). Finally, adding all the minor ones than the one concerned, the desired p-value comes out.
Returning to the code, what has just been said is not implemented in exactly the same way within the `HWExact` function. This happens because of the high computational effort in listing the triples in this way and making the necessary calculations. As introduced in Chapter 2, a *naive* approach is introduced in the paper by Wigginton et al. (2005), which facilitates the recursive calculations that

	AA	AB	BB	$P(n_{AB}\|n_A)$	$p-$ value
1	86	14	0	0.6136	1.0000
2	87	12	1	0.3209	0.3864
3	88	10	2	0.0602	0.0654
4	89	8	3	0.0051	0.0053
5	90	6	4	0.0002	0.0002
6	91	4	5	0.0000	0.0000
7	92	2	6	0.0000	0.0000
8	93	0	7	0.0000	0.0000

Table 4.2: Example with all possible samples of 100 individuals and $n_B = 14$

must be carried out, managing to obtain a fixed number of operations regardless of the size of the dataset present. In particular, with a sample size of size N, the authors reduce the calculations of the formula (4.2) to only four multiplications and one division, regardless of N, using the following equations.

$$\begin{aligned} &P(N_{\mathrm{AB}} = n_{\mathrm{AB}} + 2|N, n_{\mathrm{A}}) \\ &= P(N_{\mathrm{AB}} = n_{\mathrm{AB}}|N, n_{\mathrm{A}}) \frac{4n_{\mathrm{AA}}n_{\mathrm{BB}}}{(n_{\mathrm{AB}}+2)(n_{\mathrm{AB}}+1)}, \text{ and} \\ &P(N_{\mathrm{AB}} = n_{\mathrm{AB}} - 2|N, n_{\mathrm{A}}) \\ &= P(N_{\mathrm{AB}} = n_{\mathrm{AB}}|N, n_{\mathrm{A}}) \frac{n_{\mathrm{AB}}(n_{\mathrm{AB}}-1)}{4(n_{\mathrm{AA}}+1)(n_{\mathrm{BB}}+1)} \end{aligned} \tag{4.3}$$

The author of the package chooses to follow this path. So, after having placed the index of the sample in `ind`, it calls two functions (`CompProbUp` and `CompProbDown`) which help to fill two vectors with different probabilities, following the equations just introduced.

```
MaxHet <- min(nA, nB)
if (MaxHet < 2) {
    pval <- 1
    prob <- 1
    pofthesample <- 1
    ind <- 1
}
else {
    ind <- match(Xhet, seq(MaxHet%%2, MaxHet, 2))
    # 1,3,5..nA (if odd) or 0,2,4..nA (if even)
    enAB <- nA * nB/(2 * n - 1)
    enAB <- round(enAB, digits = 0)
    if ((enAB%%2) != (MaxHet%%2))
      enAB <- enAB + 1
```

```
    nAA <- (nA - enAB)/2
    nBB <- (nB - enAB)/2
    initialprob <- 1
    AboveExp <- NULL
    BelowExp <- NULL
    if (enAB < MaxHet)
      AboveExp <- CompProbUp(nAA, nBB, enAB, initialprob,
                             MaxHet)  # recursive formula
    BelowExp <- CompProbDown(nAA, nBB, enAB, initialprob)
    prob <- c(rev(BelowExp), initialprob, AboveExp)
    prob <- prob/sum(prob)
}
```

At this point, `prob` will contain all the probabilities of the different samples obtainable starting from the fixed alleles. From now on, three vectors are declared and filled, `Plow`, `Phigh` and `Phwe`. Given the theoretical part described so far, the last one is the only one that interests us, as the default parameters passed to the function are "selome" and "two.sided", as already discussed. This vector will contain at each index the probability value of the sample at that index.

```
if (MaxHet%%2 == 0)
    names(prob) <- seq(0, MaxHet, 2)
if (MaxHet%%2 == 1)
    names(prob) <- seq(1, MaxHet, 2)
Plow <- cumsum(prob)
Phigh <- 1 - c(0, Plow)
Phigh <- Phigh[-length(Phigh)]
Phwe <- pmin(1, 2 * Phigh, 2 * Plow)

pofthesample <- prob[ind]
pval <- switch(alternative, # [...]
    two.sided = switch(pvaluetype, dost = Phwe[ind],
    selome = sum(prob[prob <= pofthesample]), midp = sum(
   prob[prob < pofthesample]) + 0.5 * pofthesample, stop("
   invalid value for parameter pvaluetype")), stop("invalid
   value for parameter alternative"))
```

Considering the `ind` value that was stored before, thanks to the `selome` parameters the final p-value is obtained. The `switch` instructions are useful in order to have a flexible way to adapt the function according to the inputs. In this code only the

part we're interested in is reported. Finally, `HWExact` returns 3 values:

- `pval`: p-value of the exact test
- `prob`: probabilities of all possible samples with the same sample size and minor allele count
- `pofthesample`: probability of the observed sample

4.1.2 Sampling (Permutation test)

The permutation test uses a *sampling* algorithm to obtain a statistical value of the deviation or otherwise of a variant from the HWE. This is a type of test that uses the tests already described (such as the chi-square or the exact test) to calculate the simple probabilities of the marker observed. Despite being the one with the highest computational effort, this test is still very useful and, in some ways, the only one that can be used in certain cases involving a high number of alleles. In the Chapter 2 it has already been introduced at a theoretical level, especially in the multi-allelic context in which the algorithm by Guo and Thompson (1992) has been used, describing it point by point. Here we will go into a little more detail, going to observe how this is applied in practice and, in particular, in the `HWPerm` function of the `HardyWeinberg` package in `R`. The article written by Li et al. (2013), from which we will take inspiration for some considerations, analyzes the statistical power of this test by demonstrating superiority over the simple chi-square, especially on a large scale. Starting from the autosomal context with only two alleles, A and B, and with n individuals, this test consists in: the calculation of the statistics of the observed sample; subsequently, the $2n$ alleles are arranged one after the other and considered in pairs, on which the statistics are calculated according to a chosen test (the same for the example observed); finally, after N permutations, the p-value is calculated in the way we will see shortly. This can be summarized schematically in Figure 4.1, taken from the article cited above.
Also in this case we will accompany the theoretical explanation with the practical one, highlighting the key aspects of the `HWPerm` function of the same package as the previous one. If there are two alleles in the population, there will be 3 possible genotypes, two homozygous (AA and BB) and one heterozygous (AB). By putting these three elements into a vector `x`, we pass to the `HWPerm` function which, as the first step, counts alleles. One of the cardinal elements of this test concerns the type of *internal* test that you want to apply to the observed sample and obtain the probability. This can be decided by the user through the `FUN` parameter, which can be specified whether to perform a chi-square test (default), a likelihood ratio or an exact test. So in `stat.obs` we will have the test statistic of the sample we are considering.

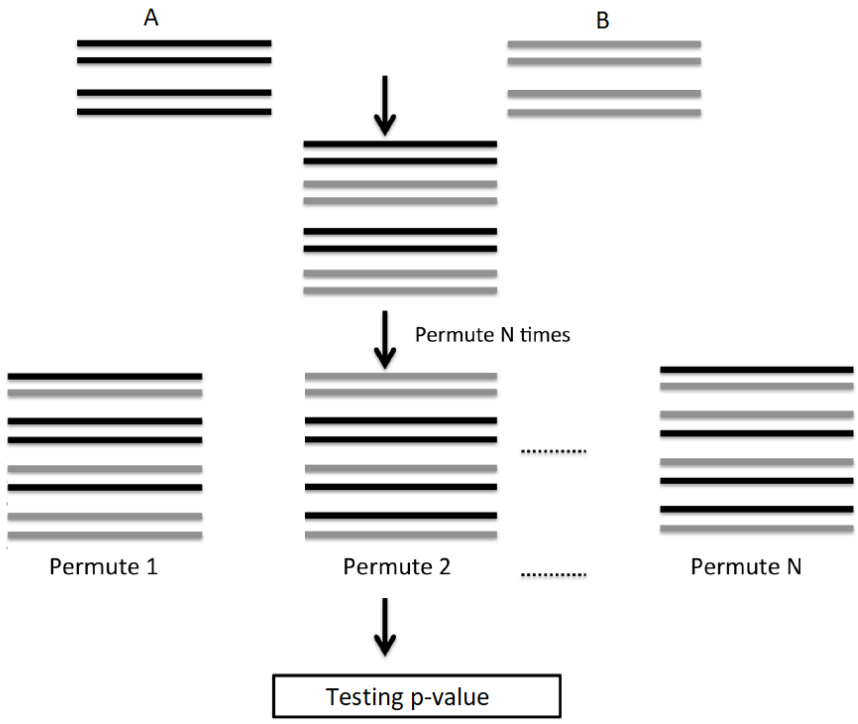


Figure 4.1: Permutation test with Monte Carlo method (Li et al., 2013)

```
n <- sum(x)
nA <- 2 * x[1] + x[2]
nB <- 2 * n - nA
stat.obs <- FUN(x)
```

The next step is to decide on the number of permutations to be made. The higher the number of random samples, the more truthful the test will be, at the expense of calculation times. The paper cited above explains mathematically that, for a threshold $\alpha = 0.01$, although it is more common to consider 0.05, and a confidence level of 99%, the ideal is about 17000 permutations, which is also the default of the function that we are analyzing . So it is appropriate to consider a parameter `nperm` that is greater than or equal to this value, also depending on the available computing power. This parameter is useful to cycle a correct number of times in a loop, within which the following operations are carried out.

1. to build a random sample with the given number of A and B alleles

2. the latter are placed in pairs and the genotypes are counted

3. to extract a vector `y`, similar to the one given at the beginning

4. to calculate the probability of `y` in the same way as `x`

```
pseudodist <- numeric(nperm)
i1 <- seq(1, 2 * n, 2)
i2 <- seq(2, 2 * n, 2)
for (i in 1:nperm) {
  xx <- sample(c(rep("A", nA), rep("B", nB)))
  A1 <- xx[i1]
  A2 <- xx[i2]
  Geno <- paste(A1, A2, sep = "")
  Geno[Geno == "BA"] <- "AB"
  nAA <- sum(Geno == "AA")
  nAB <- sum(Geno == "AB")
  nBB <- sum(Geno == "BB")
  y <- c(AA = nAA, AB = nAB, BB = nBB)
  stat.pseudo <- FUN(y)
  pseudodist[i] <- stat.pseudo
}
```

The final step is to calculate the test p-value. To do this, among all the values just calculated in the `for` loop, only those greater or equal to the value for the observed sample (`stat.obs`) are added and, finally, we divide this value by the number of permutations.

```
nlarger <- sum(pseudodist >= stat.obs)
pval <- nlarger/nperm
```

The high number of iterations is certainly a determining factor in the performance and speed of this test, confirming that the operations within the cycle are not trivial. We will see shortly how much this will affect in terms of CPU.

4.1.3 Network algorithm

As you can easily guess, the algorithms just seen are very useful and often precise to solve the initial problem. The flaw is, however, the effort necessary to do it.
In the last decades of the past century, several authors have tried to improve existing algorithms with the aim of speeding up implementation. The first authors to try this were Mehta and Patel (1983), who described a **network algorithm** that allows to obtain the Fisher's exact test more quickly. This study is then taken up again by Engels (2009) who adapts it to the case that is useful to us too and which is the starting point for the study of this thesis. In particular, assuming the fixed number of alleles to be studied, it is possible to construct an acyclic graph in which each *path* from the root to the tail represents one of the possible contingency tables. The representation and the consequent calculation algorithm varies slightly between the two papers mentioned. Having the Engels's source code available was a fundamental aid that helped a lot in the realization of what will follow. The strength and usefulness of this algorithm comes out, in particular, for a number of alleles larger than two, whether we consider all chromosomes or not. In fact, the algorithm that we will see here for the autosomes with two alleles is only the final part of the algorithm needed for $k > 2$ alleles. So you will not notice huge improvements both with regard to the procedure and the performances compared to the previous case. The most innovative and interesting part will be illustrated in the next chapter.

For the simpler case described here, a real network will not yet be seen, since with just the two alleles we require a specific but simpler treatment than what will be described below. The Engels (2009) algorithm, adapted to our needs, starts from taking into consideration an overall vector of the allele counts together with the contingency table of the specific marker to be analyzed. We consider n individuals with two alleles, also called A and B here. If with n_A and n_B we indicate the total count of A and B respectively, while with n_{AA}, n_{AB} and n_{BB} those of the corresponding genotypes, the exact test distribution is as follows .

$$P(N_{AA}, N_{AB}, N_{BB}|n_A, n_B) = \frac{n!2^{n-d}n_A!n_B!}{(2n)!n_{AA}!n_{AB}!n_{BB}!} \tag{4.4}$$

The difference with the formula (4.2) seen in the previous section is in the exponent of 2 which stands for numerator: to indicate the total number of heterozygotes, instead of doing it directly with $2^{n_{AB}}$, subtract from the total number of individuals n the factor $d = n_{AA} + n_{BB}$ which is the number of homozygotes. This is a simplification that helps the author to perform the necessary calculations, especially when very large tables come into play. If carefully observed, the formula (4.4) contains a part that always remains the same, regardless of the tables, and one that varies as the marker achievable with the available alleles changes. You can

clearly see it by rewriting it as follows.

$$P(N_{AA}, N_{AB}, N_{BB}|n_A, n_B) = \underbrace{\frac{n!2^n n_A! n_B!}{(2n)!}}_{\exp K_p} \frac{1}{2^d n_{AA}! n_{AB}! n_{BB}!} \tag{4.5}$$

Here, the first multiplicative factor is the constant one, while the second is the one that changes. The last premise to be made concerns a further simplification in the calculations. Since there are large factorials in the distributions that we want to calculate, it is good practice to use a less complex way that concerns the correlation between logarithms and exponentials. Being that one is the inverse function of the other and the logarithms greatly reduce the orders of magnitude, it is decided to carry out all the accounts by tracing the data back to the logarithm, to then perform only one exponential calculation at the end. For this reason, instead of considering the probability of having a contingency table, given the count of alleles, the logarithm is considered in this way

$$\ln P(N|n) = K_p - \sum_{i \geq j} f(n_{ij}) - d \ln 2 \tag{4.6}$$

where K_p is the constant factor just explained, n_{ij} is the count of the different genotypes, while f is a function defined as $f(i) = \ln(i!)$. The final probability is easily obtained by making the natural exponential of the result obtained. Now, the algorithm is written in `C` and called in `R` via the `.C()` function. The required inputs are: a vector of allele counts (`rm`), the number of alleles involved (`rk`, here is 2), a vector with different types of statistics (such as χ^2, LLR or simple probability), of which we consider only the probability element useful for the test we are carrying out and, finally, a vector for the return parameters. The other parameters are irrelevant for our situation as they are useful for extracting the different histograms that we have not dealt with in this project. The `R hwx.test()` function of Engels (2009) internally calls the one in `C xtest()`. It should be noted that the following code will cut out several parts of the original source code which, although present in the original one, have not been treated here and are not useful for explanation. The initial part of the function involves saving the input values in some global variables, such as `nAlleles`, and creating some auxiliary attributes, such as `Rarray`, which we will need shortly. Note the `pPr` parameter which will finally contain the p-value to be returned to the caller. Subsequently, the constant K_p, called `constProbTerm`, is built, already on a logarithmic scale, which contains the values described before that do not change between the different tables. Finally, `maxlPr` will contain the observed probability, with a certain tolerance rate, which will serve as a threshold.

```
void xtest (int * rm,
            int * rk,
            double * robservedVals, // observed stats
            double * rPvals, // computed P values
            [...]
            )
{
nAlleles = *rk;
pPr=0;
Rbytes = *rk * sizeof(COUNTTYPE);
Rarray = Calloc(*rk * *rk * (*rk-1)/2, COUNTTYPE);
for (int i = 0; i < nAlleles; i++) Rarray[i] = rm[i];
mi = rm-1; // 1-based list of allele counts
lnFact = Calloc(rm[0] + 1, double);
// Make lookup tables
lnFact[0] = 0;
double lni;
for (int i = 1; i <= rm[0]; i++) {
    lni = log(i);
    lnFact[i] = lnFact[i-1] + lni;
}
int  nGenes = 0;
for(int i = 0; i < nAlleles; i++) nGenes += rm[i];
ntotal = nGenes/2;

// Get constant terms for Prob
constProbTerm =  0;
for (int i = 0; i < nAlleles; i++) {
    constProbTerm +=  lgammafn(rm[i] + 1); //lnFact[rm[i]];
}
constProbTerm += log(2)*ntotal + lgammafn(ntotal+1) -
   lgammafn(nGenes +1);

// Get cutoffs for the four test statistics
double oneMinus = 0.9999999; // Guards against floating-
   point-equality-test errors
if(robservedVals[0] > 0.000000000001) robservedVals[0] = 0;
   // positive values are rounding errors

maxlPr = log(robservedVals[1]) * oneMinus;

if (nAlleles == 2) {
    twoAlleleSpecialCase();
```

```
}else {
    //...
}
}
```

The last step is to call the `twoAlleleSpecialCase()` function, which is the heart of the algorithm. This is very simple and includes the already introduced concepts of the allele counts vector and triangular genotype array. The first, which had been passed as the first parameter, has a form of the type $[\ n_A \quad n_B\ ]$. The second concept can be schematized in the following two ways,

$$\begin{bmatrix} n_{AA} & \\ n_{AB} & n_{BB} \end{bmatrix} = \begin{bmatrix} a_{11} & \\ a_{12} & a_{22} \end{bmatrix}$$

the first according to our usual notation, while the other more similar to the code. Here a kind of *complete enumeration* is made, in which in a `for` loop the possible tables are listed, with the parameter n_{AA} that goes from 0 up to $n_{AA}/2$. For every possible triad (n_{AA}, n_{AB}, n_{BB}), the probability is calculated using the formula (4.6) and exponentiating the result. Finally, the latter is added to `pPr` if it is less than the previously defined threshold.

```
static void twoAlleleSpecialCase() {
    unsigned a11, a21, a22, res1, res2;
    double problT, prob;
    unsigned dT;
    int hdex;
    res1 = mi[2]; // because they come ordered largest to
   smallest
    res2 = mi[1];
    tableCount = res1/2 + 1;
    for(a11 = 0; a11 <= res1/2; a11++) {
        a21 = res1-a11*2; // integer arithmetic rounds down
        a22 = (res2-a21)/2;
        problT =  lnFact[a11] + lnFact[a21] + lnFact[a22];
        dT =  a11 + a22;

        // Here come the actual probability and LLR values
        problT = constProbTerm - problT -dT * M_LN2;
        prob = exp(problT);

        //Now process the new values of prob and stat
```

```
20          probSum += prob;
21          if(problT <= maxlPr) pPr += prob;
22      } // for a11
23  }
```

The `oneMinus` variable in the caller function definitely deserves some extra attention. This is used by Engels (2009), and then inherited in the creation of the new algorithm, as **guards against floating-point equality test errors** or **problem of ties**. Due to the small numbers you are in contact with when talking about p-values and probabilities, particularly in these contexts, it is essential to have an adequate calculation **accuracy**. When calculating the probability of a sample, especially with the X chromosome, as we will see, the individual values can also be of the order of 10^{-35}. Obviously, the sum of all the values smaller than the one considered goes to create a p-value with orders of magnitude more considerable and useful to evaluate the significance of the HWE. For this reason it is important that, when comparing for example two probabilities, it emerges which one is actually smaller or larger than the other. If the precision is not sufficient, the risk is to skip in the final sum some probabilities that then substantially change the value of the final result. The variable `oneMinus` serves to give a certain downward tolerance to the probability threshold value of our marker, in order to facilitate comparisons with the others. What we have noticed, at the level of reliability of the result in comparison with the already existing complete enumeration, is that the more this value is precise and the more the two algorithms will have a better fit of the results. When, however, this value exceeds a certain threshold of precision, then it causes errors in the opposite way, eventually producing a result that is not true at all. In the complete enumeration this precision can be modified from the outside, for example in the case with 3 alleles you can add a small *eps* parameter at will that changes the final result. Returning to the explanation the first thing that can come to mind is why use these complications to end up in a situation where all possible tables are listed, when this could be done from the start as seen in the previous sections. In fact it may seem true for this context with few alleles, but it will be seen that this is only the basis of a large lattice built to have a work as efficient as it is precise.

4.2 X Chromosome

Taking into account the X chromosome in the analysis that you want to carry out, leads to a deepening with the modifications, more or less heavy, of the algorithms just seen. As explained in Chapter 2, there are some changes in the final formulas to arrive at the statistics of interest between the autosomes and the X chromosome. Obviously, this reflects on the methodologies and calculations to be done to solve problems in the most efficient way possible.

4.2.1 Complete enumeration

The complete enumeration is, also in this case, the easiest way to test the HWE according to the exact test, even if not the most efficient. Due to the still *no depth* study on this topic, there have not been many improvements in the literature to develop these calculations more quickly. There are many papers that historically describe the development of the exact test for autosomes and several authors who have implemented new techniques. Searching for a well-done article that explains these methods and the algorithms that underlie them in the context of sex chromosomes is a difficult challenge. However, we will try to give it as clear a hue as possible, always using the `HWExact` function of the `HardyWeinberg` package. The paper by Graffelman and Weir (2016) (the first one is the same author of the `R` package) is among the few able to give an explanation of how much we are interested in this work, albeit keeping in a fairly theoretical line.

Being still in a bi-allelic context, it is also necessary here to take into account, as a starting point, that the number of alleles A and B will remain unchanged and fixed in all the calculations that will be carried out. The substantial difference compared to the previous case concerns the division into sexes, males and females, and that the former possess only one X chromosome, therefore just one allele per genotype. The steps to follow, in broad terms, are always the same: building the tables, calculating the different probabilities and adding those smaller or equal to the sample observed to obtain the exact test p-value. Considering a set of n individuals, consisting of n_m males and n_f females, we will indicate, following a notation similar to the code we will see, with n_A and n_B the (fixed) number of alleles A and B. Furthermore, by precisely making a division by sex , the number of males with the alleles A and B will be indicated respectively with m_A and m_B, while f_{AA}, f_{AB} and f_{BB} identify the genotypes in the females. Finally, $n_t = 2n_f + n_m$ is the total number of alleles. In this perspective, it is certainly useful to recall the formula, already seen previously, of the calculation of the probability of the

distribution to which we are interested.

$$P(M_A = m_A \cap F_{AB} = f_{AB} | n, n_A, n_m) = \frac{n_A! n_B! n_m! n_f!}{m_A! m_B! f_{AA}! f_{AB}! f_{BB}! n_t!} 2^{f_{AB}} \tag{4.7}$$

In the `HWExact` function, it is important for the analysis with the X chromosome to set the parameter `x.linked` to TRUE. Subsequently, a vector with 5 values must be input, 2 for the male allele counts (A and B) and 3 for the female genotype counts (AA, AB and BB). This will be the sample considered for which we want to test the HWE. We show how, internally, the parameter structures are first tested and, subsequently, stored in variables that will be useful shortly for the calculations.

```
1  if (length(X) != 5 | any(X < 0))
2    stop("HWExact: X is not a 5 by 1 non-negative count
     vector for an x-linked marker")
3  if (any(!is.wholenumber(X))) {
4    warning("Genotype counts are not integers, counts will
     be rounded.")
5    X <- round(X, digits = 0)
6  }
7  lab <- names(X)
8  if (!all(lab %in% c("A", "AA", "AB", "B", "BB")))
9    stop("Unknown genotypes occurred. Supply counts as a
     named vector c(A,AA,AB,B,BB)")
10 n <- sum(X)
11 nfAA <- X[lab == "AA"]
12 nfAB <- X[lab == "AB"]
13 nfBB <- X[lab == "BB"]
14 nmA <- X[lab == "A"]
15 nmB <- X[lab == "B"]
16 nAf <- 2 * nfAA + nfAB
17 nBf <- 2 * nfBB + nfAB
18 nm <- nmA + nmB
19 nf <- n - nm
20 X <- c(nmA, nmB, nfAA, nfAB, nfBB)
21 nA <- nmA + 2 * nfAA + nfAB
22 nB <- nmB + 2 * nfBB + nfAB
23 nt <- nA + nB
24 pA <- nA/nt
25 if (nA < nB) {
26   X <- c(nmA, nmB, nfAA, nfAB, nfBB)
27 }
```

```
  else {
    X <- c(nmB, nmA, nfBB, nfAB, nfAA)
  }
  nfAA <- X[3]
  nfAB <- X[4]
  nfBB <- X[5]
  nmA <- X[1]
  nmB <- X[2]
  nAf <- 2 * nfAA + nfAB
  nBf <- 2 * nfBB + nfAB
  nA <- nmA + 2 * nfAA + nfAB
  nB <- nmB + 2 * nfBB + nfAB
  pA <- nA/nt
```

After arranging all the data received, the function internally calls `auxiliartable` which helps to make the **complete enumeration** of the possible cases. This time we start by considering the minor allele frequency at 0 in males, while giving the rest to the other allele. With the number of remaining alleles it is possible to see how many combinations of the female genotypes are achievable, following the algorithm with the sets of three elements seen for the autosomes. Subsequently, the minor allele is increased by 1 and the major allele decreases by 1, again in males, obtaining a certain number of alleles still to be distributed for females. From here the algorithm of before is re-launched and all possible cases are listed. This is repeated until the minor allele in males reaches its entirety.
Recalling an example already seen in Chapter 3, if there are 20 individuals, divided into 10 for each sex, and the minor allele is A which is present 6 times, then it can be completely enumerated having as a result that of Table 4.3. Each row of that table is only a subset of possible other enumerations within the female gender, in fact f_A is the total number of the minor allele count in females. By using this values, it's possible to build all the triples (f_{AA}, f_{AB}, f_{BB}). The complete result was already shown in Table 3.3.

```
  Z <- auxiliartable(X)
  prob <- numeric(nrow(Z))
  pofthesample <- sample.prob.last(n, nm, nmA, nA, nfAB)
  for (i in 1:nrow(Z)) {
    prob[i] <- subsamples.prob(nA, nB, nm, nf, nt, Z[i,1],
    Z[i, 2], Z[i, 3], pofthesample)
  }
  if (pvaluetype == "selome")
```

	m_A	m_B	f_A
1	0	10	6
2	1	9	5
3	2	8	4
4	3	7	3
5	4	6	2
6	5	5	1
7	6	4	0

Table 4.3: Complete enumeration 20 individuals with 6 A alleles

```
9       pval <- sum(prob)
10    if (pvaluetype == "midp") {
11      pval <- sum(prob) - 0.5 * pofthesample
12    }
```

Then, after the table, the probability of the observed sample is calculated in `pofthesample`, using the formula (4.7); subsequently, a vector `prob` is constructed thanks to the `subsamples.prob` function and it contains only the values of the probabilities less than that observed, after having analysed all the enumeration in the table. Again, the only parameter we are considering is "selome" and, therefore, the final p-value will simply be the sum of all the elements of the `prob` vector.

4.2.2 Sampling (Permutation test)

If the X chromosome with two alleles is taken into consideration, the permutation test remains a valid tool to verify HWE in a polymorphism, without complicating the reasoning with respect to the previous case with autosomes only. The theory behind this test is broadly explained in the article by Graffelman and Weir (2016). The first author, in the `HardyWeinberg` package, extends the `HWPerm` function also to this context, adapting the calculations to the distinction between males and females as we will see. The central heart of the permutation test remains the repeated *sampling* of the alleles that are arranged one after the other and the subsequent calculation of the probabilities to then obtain the p-value. The explanation seen in the Section 4.1.2 remains practically unchanged in the input parameters needed for this algorithm, except for the vector `x` of the genotype counts. Suppose that the two alleles involved in our marker are A and B. Hemizygous males will have only one of them, while females inherit two. Therefore, this time a vector of 5 elements is passed to the function, 2 for men and 3 for women (the latter in the same format as the autosomes).

The first step, after the error checks for the parameters received, is that of the allele counts and the calculation of the probability, following the test that has been chosen to use (also here it can be chi-square, likelihood ratio or exact test) thanks to the `FUN` attribute.

```
if (length(x) != 5 | any(x < 0))
  stop("HWPerm: x is not a 5 by 1 non-negative count vector
   for an x-linked marker")
if (any(!is.wholenumber(x))) {
  warning("Genotype counts are not integers, counts will be
   rounded.")
  x <- round(x, digits = 0)
}
lab <- names(x)
if (!all(lab %in% c("A", "AA", "AB", "B", "BB")))
  stop("Unknown genotypes occurred. Supply counts as a named
    vector like c(A,AA,AB,B,BB)")
n <- sum(x)
nfAA <- x[lab == "AA"]
nfAB <- x[lab == "AB"]
nfBB <- x[lab == "BB"]
nmA <- x[lab == "A"]
nmB <- x[lab == "B"]
nm <- nmA + nmB
nf <- n - nm
x <- c(nmA, nmB, nfAA, nfAB, nfBB)
nA <- nmA + 2 * nfAA + nfAB
nB <- nmB + 2 * nfBB + nfAB
nt <- nA + nB
stat.obs <- FUN(x)
```

The next step is to arrange all the alleles in sequence one after the other and permute them a certain number of times `nperm`, which by default is 17000, as explained in the article by Li et al. (2013). If there are n_m males and n_f females, the total number of alleles we will have will be $n_m + 2n_f$. From this long string, which after each sample operation will be different, the first n_m elements are considered as male alleles and the subsequent $2n_f$ female ones, taken in pairs. By counting the different genotypes obtained it is possible to calculate a new probability.
This step is repeated as many times as the number of permutations chosen and the values obtained are loaded into a `pseudodist` vector.

```
pseudodist <- numeric(nperm)
for (i in 1:nperm) {
  xx <- sample(c(rep("A", nA), rep("B", nB)))
  males <- xx[1:nm]
  nmAsim <- sum(males == "A")
  nmBsim <- sum(males == "B")
  females <- xx[(nm + 1):nt]
  i1 <- seq(1, 2 * nf, 2)
  i2 <- seq(2, 2 * nf, 2)
  A1 <- females[i1]
  A2 <- females[i2]
  Geno <- paste(A1, A2, sep = "")
  Geno[Geno == "BA"] <- "AB"
  nfAAsim <- sum(Geno == "AA")
  nfABsim <- sum(Geno == "AB")
  nfBBsim <- sum(Geno == "BB")
  y <- c(A = nmAsim, B = nmBsim, AA = nfAAsim, AB = nfABsim,
         BB = nfBBsim)
  stat.pseudo <- FUN(y)
  pseudodist[i] <- stat.pseudo
}
```

The last step consists of adding, among all the values obtained, those greater than or equal to the value of `stat.obs`, in order to obtain the final p-value.

```
nlarger <- sum(pseudodist >= stat.obs)
pval <- nlarger/nperm
```

4.2.3 Network algorithm

From now on, what will be illustrated is, in large part, a novelty compared to what can be found in the literature. Until now, the network algorithm developed by Engels (2009) has never been adapted to all the chromosomes present in humans, including the X. The work done from here on was only achievable thanks to what the author just mentioned left in *open source*; in fact the 'small' changes to his work, with targeted additions, made it possible to create this algorithm without excessive effort. Also in this case the impact and the real innovation will be more evident with $k > 2$ alleles which will be explained in Chapter 5, but here we begin

to see the reasoning made to set the work in the bi-allelic situation. Without going too far in defining a problem already seen in theory, we see the elements useful for understanding the **network algorithm** for the X chromosome.
If we have a number of individuals n with two alleles, A and B, divided into males and females, we know that the former inherit one while the females two, going to build the genotype. We indicate with n_m and n_f the number of male and female individuals respectively, therefore the total count of the alelles will be $n_t = n_m + 2n_f$. To define the exact test distribution for the HWE we call the formula (2.9), seen a few chapters earlier, with a slightly different notation. It can be rewritten in this way.

$$P(M_A = m_A \cap F_{AB} = f_{AB}|n, n_A, n_m) = \frac{n_A!n_B!n_m!n_f!}{n_t!m_A!m_B!f_{AA}!f_{AB}!f_{BB}!}2^{n_f-d} \tag{4.8}$$

If it is easy to understand what denominator terms are, in which the specific count for males (m) and females (f) appears, the same may not happen for the exponent of the term 2. As already seen for the same algorithm with the autosomes, this notation then helps in writing the code. In particular $d = f_{AA} + f_{BB}$ is the total number of homozygotes. This time, unlike before, we only consider one gender and this will be reflected in the code that we will see. Let's make some premises that help us understand better. Taking advantage of the same reasoning made with autosomes and considering that to calculate the final p-value you have to build all the possible tables, in the formula above you can see 3 different factors, one of which is constant for all the tables (K_p) and the others that change in different points depending on the sample considered.

$$P(M_A = m_A \cap F_{AB} = f_{AB}|n, n_A, n_m) = \underbrace{\frac{n_A!n_B!n_m!n_f!}{n_t!}}_{\exp K_p}\underbrace{\frac{2^{n_f}}{m_A!m_B!}}_{\exp \frac{1}{K_{males}}}\frac{1}{f_{AA}!f_{AB}!f_{BB}!2^d} \tag{4.9}$$

As you can see, the constants are indicated with exp. This happens because all multiplications become sums if we move on to logarithms, simplifying complexity by far. Therefore the probability of each marker will be on a logarithmic scale, according to the following formula, to be transformed at the end.

$$\ln P(M_A \cap F_{AB}|n, n_A, n_m) = K_p - K_{males} - \sum_{i \geq j} f(f_{ij}) - d \ln 2 \tag{4.10}$$

where f is a function defined as $f(i) = \ln(i!)$ and f_{ij} is the female genotype count for the alleles i and j .

Now let's see how the actual algorithm behaves. Since the code by Engels (2009) was written in `C`, in this case too we proceed following the same path. Externally, this function can be imported into `R`, through the `Rcpp` package (François and Eddelbuettel, 2011) and after being rewritten in order to adapt the `C` language to `C++`. The only added parameter with respect to the function with autosomes is a two-element vector (`mf`) containing the number of males and females involved in the analysis. The procedure can be summarized by points as follows.

1. Useful constants, such as K_p, are calculated and data structures are set
2. A recursive function is launched to check the possible combinations of males, females and number of alleles for each one
3. Probabilistic calculations are carried out for each specific sample
4. The probabilities less or equal than the observed marker are added

The main function is called `xChrom()` and is similar to `xtest()` seen before, so we highlight the fundamental differences. In addition to the `Rarray` vector, another one, `alleleVect`, is added as a global variable. Both are only two elements (equal to the number of alleles): the first will contain those intended for females while the second will be for males. In this way the final calculations with the female genotypes will remain as similar as possible to the case of the autosomes already described. The other fundamental step is that of calculating K_p, here called `constProbTerm`, which contains the values seen in the formula (4.9) and whose logarithm is made. Finally, `maxlPr` will contain the probability of the observed sample which will act as a threshold for the calculation of the p-value, also in this case with a tolerance that, probably, is even more useful than in the previous case given the high weight of the denominator that causes infinitesimal numerical values..

```
1  void xChrom (int * rm,
2               int * mf,
3               int * rk,
4               double * robservedVals, // observed stats
5               double * rPvals, // computed P values
6               [...]
7               )
8  {
9      // Set up global variables used during recursion
10     nAlleles = *rk;
11     male = mf[0];
12     female = mf[1];
13     pPr = 0;
```

```
14      //[...]
15      Rarray = Calloc(*rk * *rk * (*rk-1)/2, COUNTTYPE);
16      alleleVect = Calloc(*rk * *rk * (*rk-1)/2, COUNTTYPE);
17      for (int i = 0; i < nAlleles; i++) {
18              Rarray[i] = rm[i];
19              alleleVect[i] = rm[i]+1;
20      }
21      //[...]
22      // Make lookup tables
23      lnFact = Calloc(rm[0] + 1, double);
24      lnFact[0] = 0;
25      double lni;
26      for (int i = 1; i <= rm[0]; i++) {
27          lni = log(i);
28          lnFact[i] = lnFact[i-1] + lni;
29      }
30
31      int  nGenes = 0;
32      for(int i = 0; i < nAlleles; i++) nGenes += rm[i];
33      ntotal = nGenes/2;
34
35      // Get constant terms for LLR and Prob
36      constProbTerm =  0;
37      for (int i = 0; i < nAlleles; i++) {
38          constProbTerm +=  lgammafn(rm[i] + 1);
39      }
40       int nt = male + 2*female;
41      constProbTerm +=  lgammafn(male+1) + lgammafn(female+1)
    - lgammafn(nt+1);
42
43
44      // Get cutoffs for the four test statistics
45      double oneMinus = 0.9999999;
46
47      if(robservedVals[0] > 0.000000000001)
48          robservedVals[0] = 0; // positive values are
    rounding errors
49      maxlPr = log(robservedVals[1]) * oneMinus;
50
51      recursiveEnumeration(1);
52
53      //[...]
54  }
```

Although the code is similar to the autosomal context, some subtle differences have already been underlined. The fundamental part is shown at the bottom with the call to `recursiveEnumeration()` and the disappearance of the switch by number of alleles. This function, which has an integer as its only parameter, launches a recursion that scrolls all the allele counts available and serves to delineate the possible combinations of alleles distributed in the males, as long as there remains an amount that coincides with the number of female individuals. For each of the male tables found, the `twoAlleleSpecialCaseX()` function is called, which is the correspondent of `twoAlleleSpecialCase()` for autosomes. The basic data structures, at this point in the program, will be composed by `alleleVect`, which is filled with the number of alleles for males, and `Rarray` with those to be distributed to females. Apart from some calculations related to K_{males}, called `constMales` here, which includes what is collected in the second multiplicative factor of the formula (4.9), the next procedure will be the same already seen with the autosomes, given the available triangular matrix with two alleles.

```
1  static void twoAlleleSpecialCaseX() {
2      unsigned a11, a21, a22, res1, res2;
3      double problT, prob, constMales=0.0;
4      unsigned dT;
5      int  nGenes = 0, nMales=0, n;
6
7      res1 = mi[2]-alleleVect[1]; // because they come ordered
       largest to smallest
8      res2 = mi[1]-alleleVect[0];
9      tableCount = res1/2 + 1;
10
11     for(int i=0;i<nAlleles;i++){
12         constMales += lgammafn(alleleVect[i]+1);
13         nMales += alleleVect[i];
14         nGenes += Rarray[i];
15     }
16     n = (nGenes-nMales)/2;
17     constMales -= log(2)*n;     // add 2^n in the constant
     males
18
19     for(a11 = 0; a11 <= res1/2; a11++) {
20         a21 = res1-a11*2; // integer arithmetic rounds down
21         a22 = (res2-a21)/2;
22
23         problT =  lnFact[a11] + lnFact[a21] + lnFact[a22];
24         dT =  a11 + a22;    // 2^h    (h=n-d) only females
```

```
25          problT = constProbTerm - problT - dT * M_LN2 -
    constMales;    prob = exp(problT);
26
27          probSum += prob;
28          if(problT <= maxlPr) pPr += prob;
29      } // for a11
30  }
```

The final part includes the calculation of the p-value, adding up the probabilities obtained if they are less than `maxlPr`. Throughout the algorithm, only the most useful parts have been shown, avoiding the reporting of auxiliary functions that allow you to give a cut when you cannot proceed with the initial recursion and speed up the calculations. Let's now see how the described algorithms affect different examples in terms of performance, in order to understand how influential the choice is when there are only two alleles and the alternatives are manifold.

4.3 Performance in terms of CPU

To verify how efficient the 3 different algorithms just illustrated are, it was decided to use one of the samples from the *1000 Genomes Project*. In particular, it was decided to analyze a population from Tuscany, Italy, which has 107 unrelated individuals, including 53 males and 54 females (The 1000 Genomes Project Consortium, 2015). All SNPs of chromosome 1 and X chromosome were extracted from this database, called TSI. From the first case we have drawn up an analysis for *autosomes*, without considering gender; the opposite was done in the second case for the *X chromosome*. The three algorithms (complete enumeration, permutation and network) were launched with an increasing number of polymorphisms and their output timings were detected. To take the time needed for the different functions, in `R` it was enough to detect these with the `Sys.time()` function at the beginning and at the end of the algorithm under analysis and subtract the two values obtained to have the final time. For the first two, the `HWExactStats()` and `HWPerm()` functions were used in `R`: they extract the p-values for exact test in the most efficient way possible with many SNPs in input. These functions are part of the `HardyWeinberg` package in `R` (Graffelman and Camarena, 2008), therefore their call has **no particular delays** if not inherent in the functions themselves. As for the network algorithm, however, both for the code by Engels (2009) and for the modification in the case of the X chromosome, an executable was extracted, as seen in the examples of Chapter 3, called in `R` with `system()`. To obtain the results, the executable **prints** all the p-values in a text file, which are then read in `R`. Printing output and reading input of the data passed as parameters are fundamental since, at the moment, we

did not manage to find a clean and functional integration of the code between the C/C++ and R language. We are confident that this work will be carried out in the future, but for now we must take into account that the **network algorithm is partially affected by the performance of these external factors**.
Regarding the **autosomes**, the database without missing data and without considering monomorphic variants contains a number of markers equal to 1,155,326. After arranging the data so as to conform to what the functions want as input, it was decided to proceed step by step by analyzing the SNPs in powers of 10 $(1, 10, 100, ...)$ up to covering the entire data set. Since the permutation test is much slower than the other two, this has only been tested for up to 100 variants. As can be seen in Figure 4.2, the calculation time explodes for sampling, while it remains close to 0 for enumeration and the network. In Figure 4.3 a zoom of the previous case is shown.

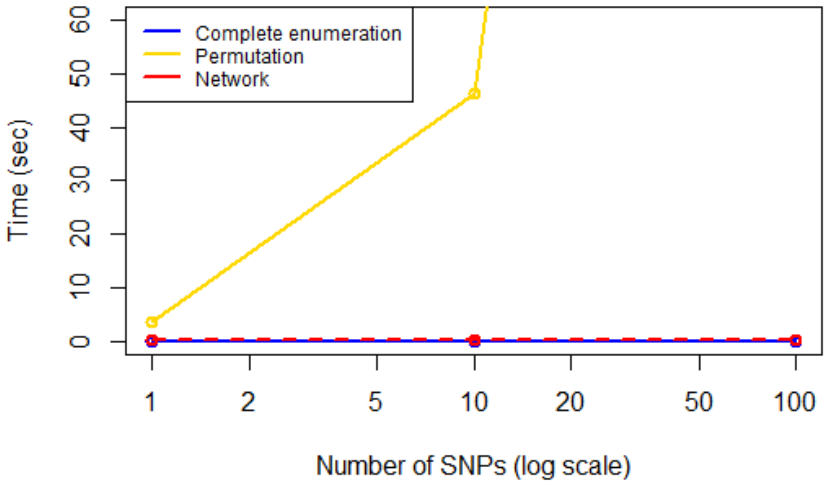


Figure 4.2: Performance plot for autosomes with 2 alleles (up to 100 SNPs)

Continuing to study the entire data set, we notice a lower time of the network algorithm than the enumeration for a number of markers of 10^5, and then go back up when the database is complete up to about 11.5 seconds, against the 5.8 of the other (see Figure 4.4).
These data make us understand how the complete enumeration function for 2 alleles is well structured and how the network is not fully exploited. Moreover, this small difference can be explained considering the overhead mentioned before, for which the network algorithm works externally from R and needs some I/O operations

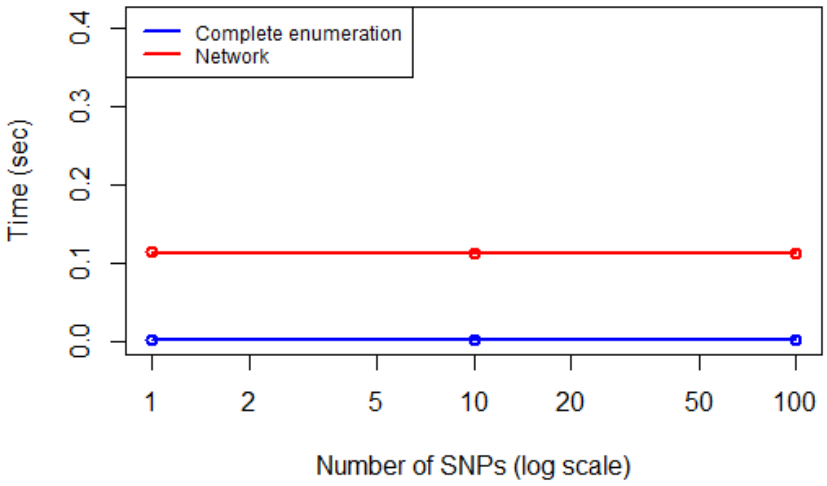


Figure 4.3: Zoom only on complete enumeration and network algorithm

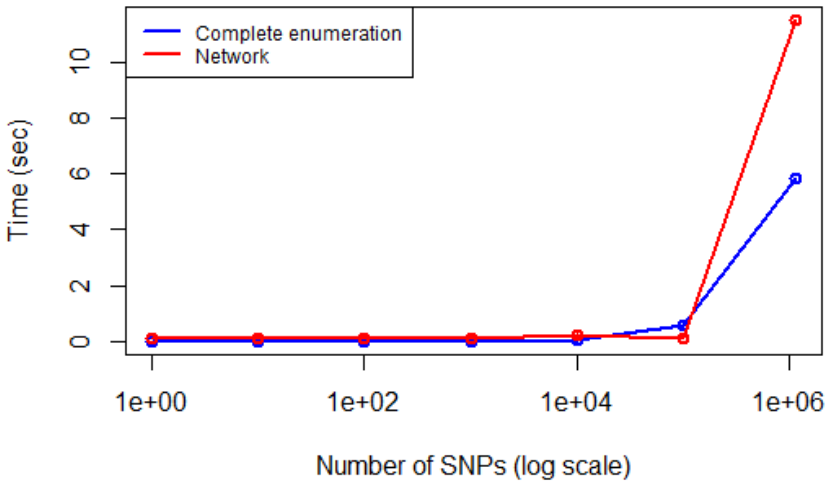


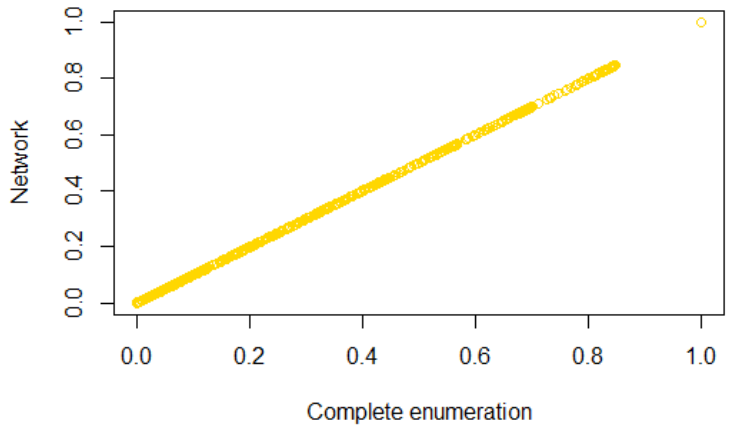
Figure 4.4: Performance plot for autosomes with 2 alleles (whole dataset)

that in such a short time make the difference. These concepts had already been mentioned in the previous theoretical explanation, but here we see the result in

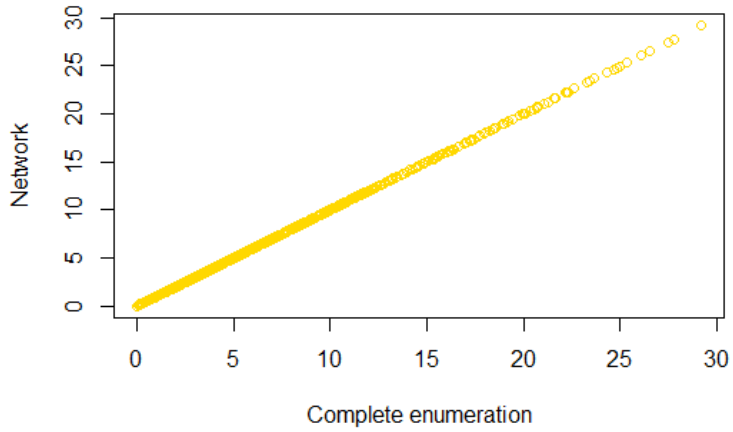
practice. The Table 4.4 shows the exact overall data, in which the difference is only a few seconds between the two algorithms, if you consider the almost 5 minutes taken by the permutation test with only 100 markers. The fundamental point is that of the comparison between the two sets of final p-values. Figure 4.5 indicates how in general the two algorithms are equivalent from the point of view of the truthfulness of the results, with a maximum difference between outputs of $4.997736e-08$. In the first plot, the natural p-values are put in comparison, while the second one uses the $-log_{10}$ scale that is often used in genomics studies in order to emphasise the smaller p-values, i.e. the most significant ones.

	Complete	Permutation	Network
1	0.002038002	3.558928	0.1130481
10	0.001992941	46.391947	0.1125062
10^2	0.002020121	470.234341	0.1129351
10^3	0.007975101	-	0.1122561
10^4	0.052883863	-	0.2157109
10^5	0.582729101	-	0.1107490
All	5.826529026	-	11.4904130

Table 4.4: Computation times (in seconds) for autosomal variants with 2 alleles



(a) normal scale



(b) -log10 scale

Figure 4.5: Complete enumeration and network outputs comparison for autosomes with 2 alleles

A similar analysis was carried out for the **X chromosome**. This time the total number of SNPs for the Tuscan population is 391,152. The fundamental differences concern the formulas for calculating the probabilities, taking into account the distinction between males and females, and the use of the new network algorithm that is being tested for the first time. Also in this case it was decided to go

gradually, for powers of 10, from 1 SNP to the whole set. The final results are very similar for trends compared to the autosomal context, with subtle differences in times between the complete enumeration and the network and with the explosion of the permutation test already with 10^2 markers. Figures 4.6 and 4.7 show times up to 100 SNPs, while Figure 4.8 includes all the data, excluding the clearly inefficient sampling algorithm. It is clear how the network algorithm, albeit slightly, is less efficient than the complete enumeration.

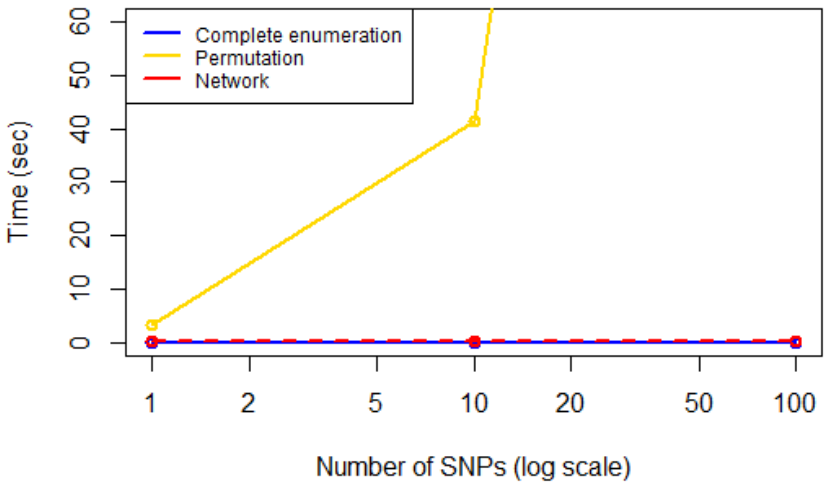


Figure 4.6: Performance plot for X chromosome with 2 alleles (up to 100 SNPs)

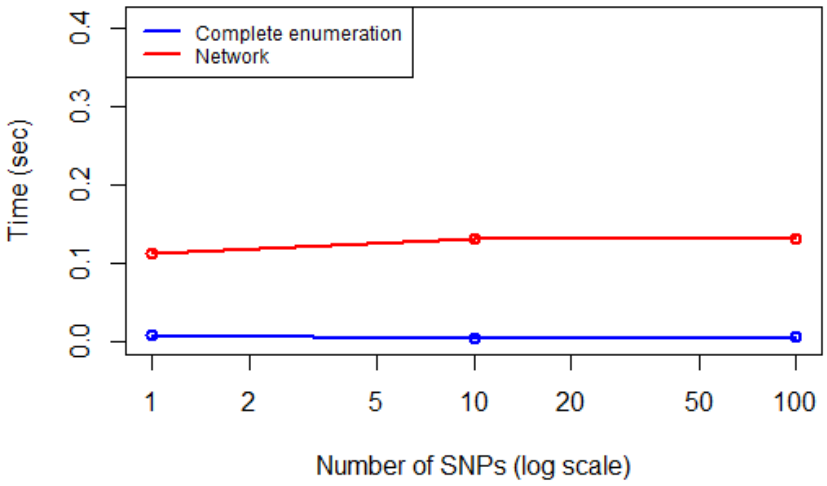


Figure 4.7: Zoom only on complete enumeration and network algorithm

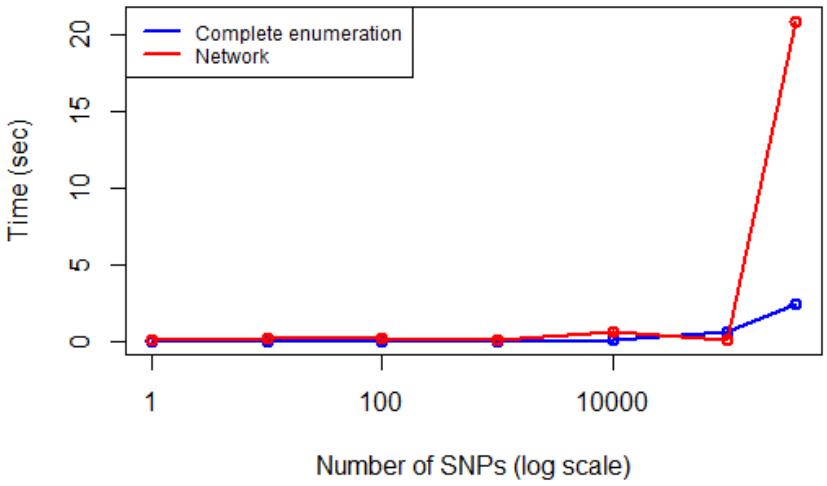


Figure 4.8: Performance plot for X chromosome with 2 alleles (whole dataset)

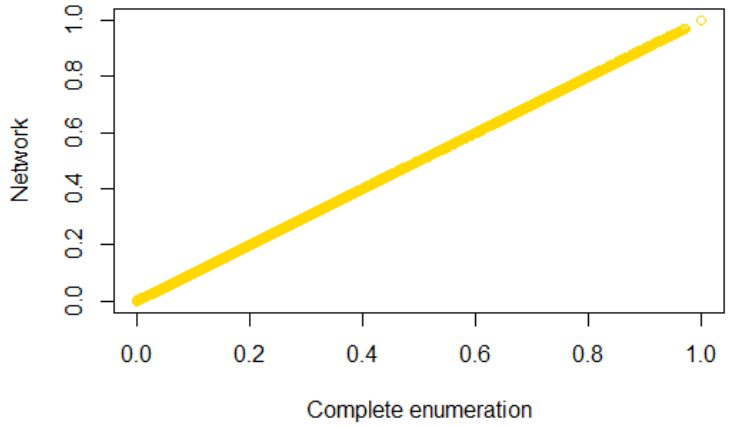
The Table 4.5 shows the precise data, from which one can start by asking why the complete enumeration is not exploited and that's all, without trying to find other

ways.

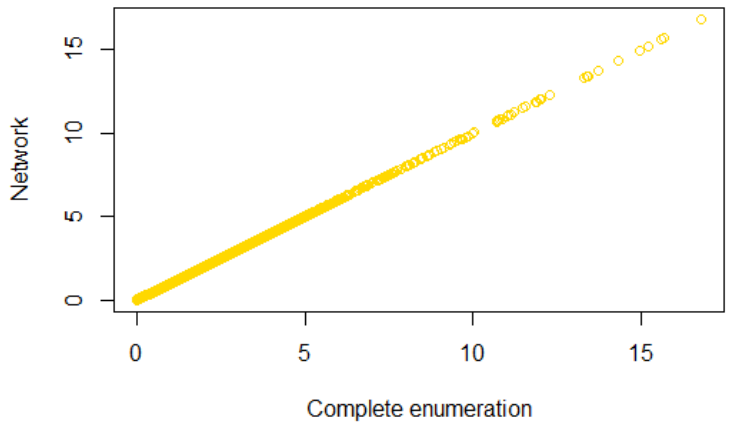
	Complete	Permutation	Network
1	0.0069820881	3.172202	0.1208451
10	0.0049870014	41.280097	0.1125350
10^2	0.0009970665	439.160565	0.1130559
10^3	0.0119681358	-	0.1116209
10^4	0.0728259087	-	0.6181090
10^5	0.6193430424	-	0.1128709
All	2.4702091217	-	20.9553330

Table 4.5: Computation time (in seconds) for X chromosomal variants with 2 alleles

The first point that gives us confidence is that shown in Figure 4.9, in which it is noted that the results between an algorithm that has been in existence for years and a completely new one are practically similar, with a maximum difference of $4.998902e-08$. Secondly, it has already been explained that with only 2 alleles one can not allow to build a large network and that it really gives us an advantage, without considering the external factors mentioned before because they partially slow down the times. All these considerations have spurred us to move forward with this approach also for a number of alleles $k > 2$ where, as we will see in the next Chapter, we begin to understand why it was really necessary and how it affects performance with large amounts of data.



(a) normal scale



(b) -log10 scale

Figure 4.9: Complete enumeration and network outputs comparison for X chromosome with 2 alleles

Chapter 5

Algorithm for the k-allelic exact test

The chapter that is being introduced now is the most important, but at the same time more complex, of the whole project we are carrying out. The statistical testing of Hardy-Weinberg proportions in a certain population with a large number of alleles has acquired fundamental importance over the years and, therefore, the same has happened for the parallel analysis of innovative procedures to carry out the analysis. The network algorithm introduced in the previous chapter seems to be good enough in terms of performance and reliability. For this reason, we will go only into detail of this algorithm, both for the autosomes and for the X chromosome, avoiding to recall the complete enumeration and the permutation test. Despite this, they will be briefly mentioned in the final part of the chapter to have, also in this context, feedback in terms of differences in computation power between the methods. For both of the last two tests mentioned, it is quite simple to algorithmically understand how generalization from 2 to k alleles takes place, given the mechanics with which they are developed. Precisely for this reason they are simple in construction but slow in terms of performance. The **network algorithm**, on the other hand, is quite innovative compared to the others and will therefore be analyzed in detail, in order to allow the reader to fully understand its operation that is by no means trivial. We will start by taking up the algorithm by Engels (2009), which has worked only with autosomes, trying to better explain some things taken for granted in his article, but neglecting other detailed ones that go beyond this thesis. Subsequently, the **new network algorithm** developed for the X chromosome will be introduced, which is strongly based on that by Engels but with differences upstream for the distinction of the two sexes.

5.1 Autosomes

In his paper, Engels (2009) deals with different aspects of the exact test with a number of alleles larger than 2. He talks about the possible probabilistic tools that can be used, making a comparison between the simple probability, which is the one seen so far and also the only one that we will continue to treat, and the likelihood ratio (LR). After that he explains two different algorithms that take advantage of the previous definitions, the network algorithm and the Monte Carlo method. Looking through the article quickly, you notice how in reality the network algorithm is called by Engels *full enumeration*. We will now see how, in reality, that explained by the author is a sort of hybrid between the two things, in that it is impossible to do without enumerating all the cases, in order to obtain a precise and reliable p-value of the exact test. However, this does not remove the possibility of doing it in the most optimal way possible, taking advantage of the concepts already introduced by Mehta and Patel (1983) and drawing up an intrigued and efficient algorithm.

Let's start with the formal definition of the problem to be addressed, using a notation similar to that of Engels both in the paper and in his `HWxtest` package, from which we will take some parts of code that implement in practice what is described in theory. If we have a sample of n individuals with k alleles, the diploid genotypes can be represented in a lower triangular matrix $k \times k$, where each element a_{ij} indicates the number of genotypes for the alleles i and j. This matrix, also called the **triangular genotype array**, presents homozygotes along the diagonal and heterozygotes below.

$$\begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & \vdots & \ddots & \\ a_{k1} & a_{k2} & \dots & a_{kk} \end{bmatrix}$$

From here we can extract the **vector of the alleles count** where each element $n_i = 2a_{ii} + \sum_{i>j} a_{ij}$.

$$\begin{bmatrix} n_1 & n_2 & \dots & n_k \end{bmatrix}$$

If a sample is obtained from a population in HWE, its probability given the count of alleles is given by Levene's (1949) result:

$$P(N_{ij}|n_i) = \frac{n!2^{n-d}\prod n_i!}{(2n)!\prod_{i\geq j} a_{ij}!} \tag{5.1}$$

that is a generalization of equation (4.4) for k alleles, in which $d = \sum a_{ii}$ is the total number of homozygotes.

The central point of our problem is that, given a vector of allele counts, the number of genotype arrays that can be constructed increases with the number of alleles and genotypes. To test HWE, the formula (5.1) must be used for each of these arrays. It is precisely here that the author introduces an innovation: to represent the set of tables with a network made of nodes and edges, in a similar way to what was done by Mehta and Patel (1983). Therefore, starting from a vector of fixed alleles, each path of this network, which must be traveled from left to right, represents one of the possible tables. If we have 4 alleles available, each one repeated 2 times, we will have the vector made as follows

$$\begin{bmatrix} 2 & 2 & 2 & 2 \end{bmatrix}$$

and from which 17 contingency tables can be extracted. These can be represented with the network in Figure 5.1.

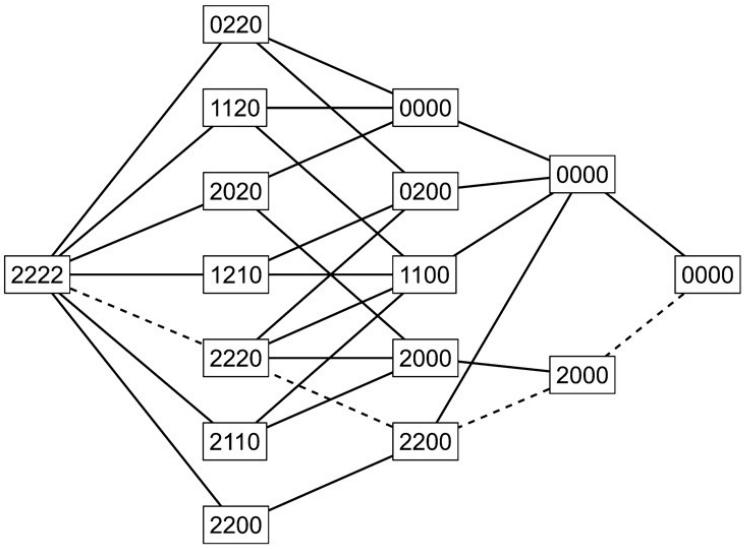


Figure 5.1: Example of network with 4 alleles (Engels, 2009)

In particular, the dotted path is that relating to the matrix

$$\begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Before explaining the network construction algorithm and its calculations to extract the final p-value, to test deviation from HWE, we better interpret the network just shown, trying to understand how it should be read and, especially, how to go from

a path to its table. "The four digits identifying each node are the residual allele counts, and each column of nodes represents the genotype assignments for one of the rows of the table, starting with the bottom" (Engels, 2009). This means that, for each edge that leaves one node and enters another, a row of the contingency table of that path is built. Furthermore, each column of the network (meaning by this the set of nodes appearing in the same *vertical section*), refers to one of the elements in the vector as homozygous. In particular, it starts from the last element to the right of a node and ends with the first on the left. The procedure used to build the table from a path is schematically illustrated in Figure 5.2. Reading from the bottom, each arc connecting two nodes corresponds to a row in the matrix where you can see how many genotypes are formed by an allele with all the others. Analyzing the two nodes term by term, the subtraction of each with the one above, reveals the genotype count corresponding to that index in the corresponding row. Finally, if the term we are analyzing is homozygous (circled here in red), it must be divided by two.

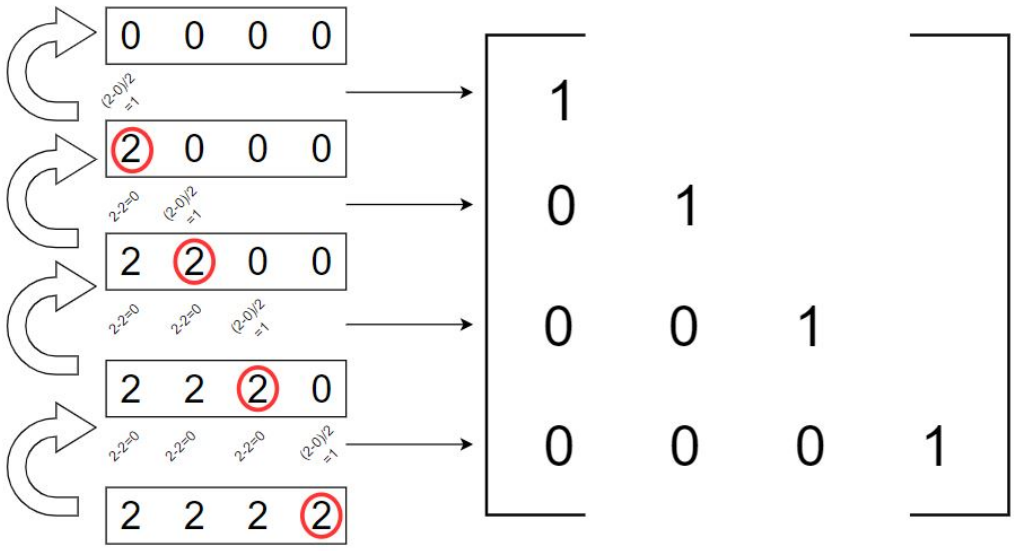


Figure 5.2: Explanation of table construction with 4 alleles following the path

The reason for the presence of some arrays with a certain allele count rather than another is a legitimate question at this point. It seems clear, now, how to interpret the various paths of a ready-made network, but how to understand which are the elements of the graph exactly? Why does the sequence of the dashed path have, in each node, exactly those arrays? These questions will be answered shortly, when

the algorithm will actually be presented. Let's already anticipate that passing from one *vertical section* to the next, it is as if we were going up the triangular matrix of the genotype count. For this reason, every time we move forward by one section, we find an extra 0 at the end of the node: the triangular matrix, by definition, presents at each line starting from the bottom and going upwards, always one element less than the previous one (also considered as 0).
Before proceeding, it is possible to notice that in this example a detail is missing, due to the fact that the matrix contains many zeros inside it. Proceeding with the member-to-member subtractions, if the result is not 0, this value must be subtracted in turn in the calculation of the homozygous term, before the division by two. Let's quickly analyze another network in Figure 5.3, taken from the same paper, in which there are 5 alleles and a network with 139 paths. The analysis made before to go back to the table is the same, in which it is possible this time to note that the residual alleles before the last term are not always equal to 0. Therefore, following the dotted path, the final result is indicated in Figure 5.4.

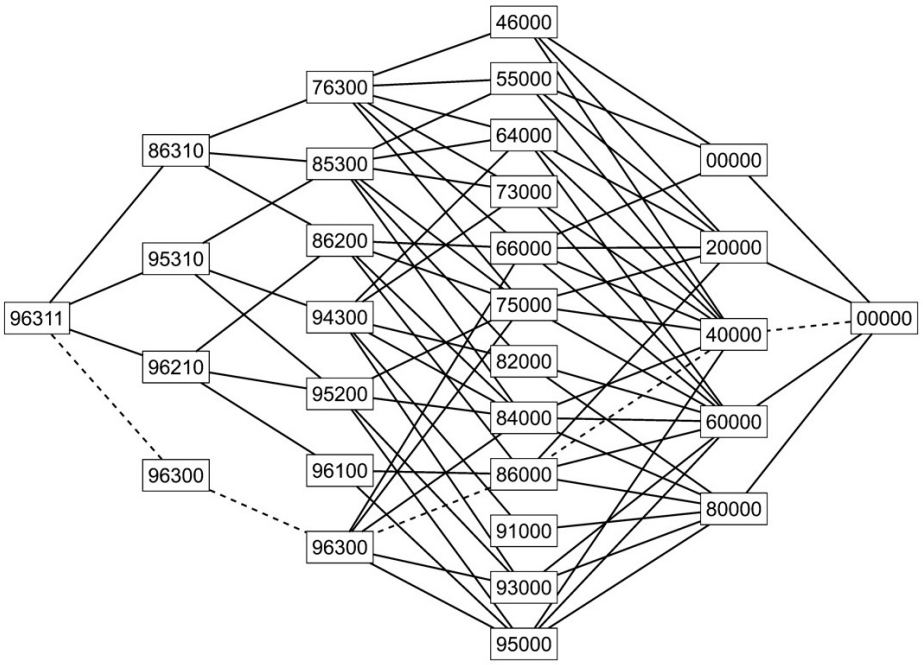


Figure 5.3: Example of network with 5 alleles (Engels, 2009)

Having, therefore, a little clearer the interpretation of the network with which we will work, we go to see the real algorithm. The probability calculations according to the equation (5.1) are distributed along the network, so as to computationally weigh as little as possible. As already seen for the bi-allelic case, this time too we will work in a logarithmic scale, so as to make all the recursive multiplications

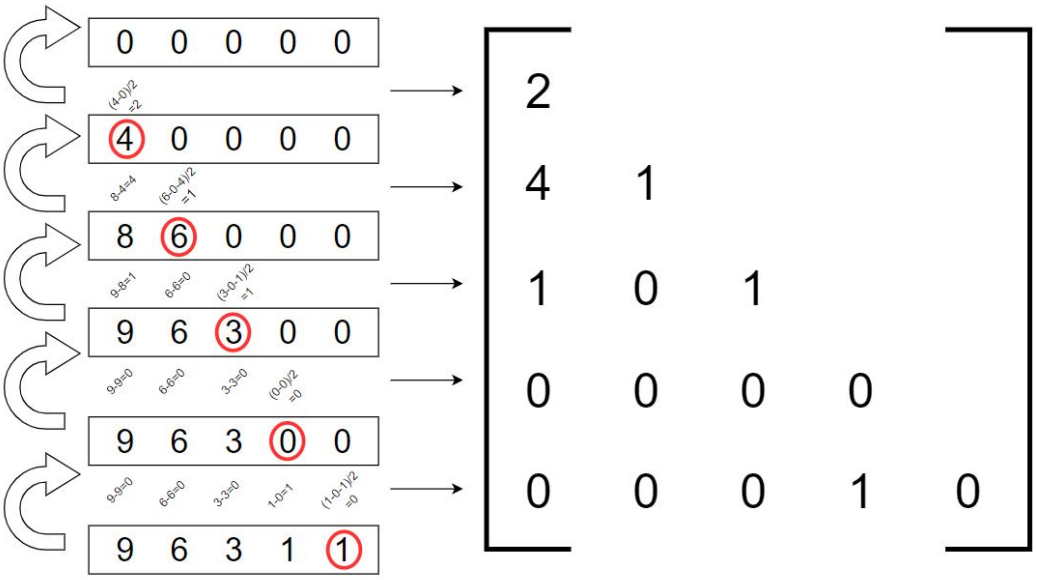


Figure 5.4: Explanation of table construction with 5 alleles following the path

present lighter. In light of this, it is possible to think of the formula just referred to as the set of two factors, one constant that we will call K_p, and the other variable as the contingency tables vary.

$$P(N_{ij}|n_i) = \underbrace{\frac{n!2^n \prod n_i!}{(2n)!}}_{\exp K_p} \frac{1}{2^d \prod_{i \geq j} a_{ij}!} \tag{5.2}$$

Hence, the logarithm of the probability of obtaining a certain genotype array will be given by

$$\ln P(N) = K_p - \sum_{i \geq j} f(a_{ij}) - d \ln 2 \tag{5.3}$$

where K_p is the constant factor, d is the total number of homozygotes, while f is a function such that $f(i) = \ln(i!)$.

The algorithm is basically based on 3 parts:

1. first part with calculation of the constant K_p
2. *homozygote()* function which works on each **node** of the network
3. *heterozygote()* function called for each **edge** of the network

From a code point of view, the call to the `xtest()` function denotes the beginning of the algorithm. We have already seen what it contains within the same context but with two alleles. We therefore underline only the fundamental aspects: the calculation of K_p and the first call to `homozygote()` function.

```
void xtest (int * rm,
            int * rk,
            double * robservedVals, // observed stats
            double * rPvals, // computed P values
            [...]
            )
{
    // [...]
    constProbTerm = constLLRterm = 0;
    for (int i = 0; i < nAlleles; i++) {
        constProbTerm +=  lgammafn(rm[i] + 1);
    }
    constProbTerm += log(2)*ntotal + lgammafn(ntotal+1) -
   lgammafn(nGenes +1);
    // [...]
    if (nAlleles == 2) {
        twoAlleleSpecialCase();
    } else {
        homozygote(nAlleles, 0, 0, 0, 0, Rarray);
    }
    // [...]
}
```

The arrays `rm` and `Rarray` contain the residual allele counts, initially set to the values contained in the *vector of the allele count*, $n_1, \ldots, n_k$, in descending order. The order does not change the final result but influences the performances. The constant K_p, therefore, contains what can be seen from the formula (5.2). The other useful parameter is `robservedVals`, in which there are the statistical values observed for the marker considered. In addition to the exact test that uses the definition of probability of the equation (5.1), the author also includes the computation of LR and the chi-square statistics in the program. The latter have not been analyzed and will not be explained, as the definition of probability for the exact test is the only one we need now and afterwards for the extension to the X chromosome. If the number of alleles is larger than 2, the function *homozygote()* is then called. The analysis starts from the first node, starting from the left, of the network and from

the last row of the contingency table. The function receives 3 parameters as input:

1. r: row of the table we are analyzing, initialized to k (the number of alleles)
2. f_p: represents the partial sum $\sum f(a_{ij}) + d \ln 2$ of the probability that we are calculating, initialized to 0
3. R: vector of the residual count allele, initialized with the one passed as input to the calling function

The other parameters refer to the other statistical values that we will not see.
The *homozygote()* function initially provides for the calculation of the upper and lower limit of the value of a_{rr} considered, according to the following formulas, taking into account how many alleles remain available.

$$\text{lower} \ = (R_r - \sum_{i=1}^{r-1} R_i)/2$$

$$\text{upper} \ = R_r/2$$

If *lower* is negative, it is set to 0. Furthermore, the sum is to be understood as integers, therefore rounded down. At this point, **for each value between the two extremes**, the call to the *heterozygote()* function is made, which receives one more input: the column c that is being analyzed in the table. In detail, the parameters passed are:

1. corresponding row r
2. column c, here equal to $r - 1$
3. update of f_p: $f_p + f(a_{rr}) + a_{rr} \ln 2$
4. updated allele array R'

The array R' is obtained by subtracting $2a_{rr}$ from the value of the same allele in R_r. Since a_{rr} is homozygous, it is part of the value d which provides for its total sum, as described by the formula (5.3). Column c moves one "cell" backwards, until it then traverses the entire corresponding row (see Figure 5.5). What we are doing, in a nutshell, is going to calculate the "length of the arcs" of each path, accumulating it in the parameter of probability, up to the last node.

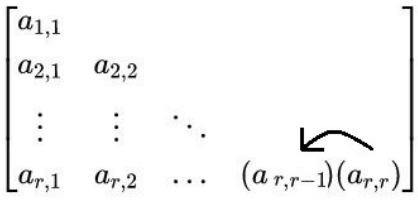


Figure 5.5: Scrolling the contingency table row

The code clearly shows what has been said.

```
static void homozygote (unsigned r, double probl, [...],
   COUNTTYPE * R)
{
    //[...]
    COUNTTYPE * Rnew = R + nAlleles;
    memcpy(Rnew, R, Rbytes);
    //Find upper and lower limits for arr.
    res = R-1;  // So res is a 1-based version of R
    resn = Rnew-1; // resn is 1 based for Rnew
    lower = res[r];
    for (i = 1; i <= r-1; i++) lower -= res[i];
    lower = lower < 2 ? 0 : lower/2;
    upper = res[r]/2;
    //For each possible value of arr, examine the
   heterozygote at r, r-1
    for(arr = lower; arr <= upper; arr++) {
        resn[r] = res[r] - 2*arr;    // subtracting the
   homozygous a_rr
        arrln2 = arr * M_LN2;
        heterozygote(r,
                     r-1,
                     probl + lnFact[arr] + arrln2,
                     [...],
                     Rnew);
    }
}
```

The subtraction of $2a_{rr}$ from the vector of the residual alleles immediately brings to mind the explanation made at the beginning of the paragraph concerning the interpretation and reading of the final network.

Let's now see the *heterozygote()* function, whose call is the last operation performed by the previous one. This is a **recursive** procedure, which behaves differently depending on the row (r) and column (c) parameters received. The first step is, also here, the calculation of the upper and lower limits of the value of the genotype a_{rc} that we are considering.

$$\text{lower} = (R_r - \sum_{i=1}^{c-1} R_i)/2$$

$$\text{upper} = min(R_r, R_c)$$

Now let's see in detail the three cases that can occur at this point of the algorithm.

a) $c > 2$

In this case we have not yet come to observe the whole row of our table. Then the *heterozygote()* function is called recursively with the parameters $[r, c-1, f_p+f(a_{rc}), R']$, where R' is constructed by subtracting a_{rc} from R_r and R_c. In this way the next call will analyze, in the same row, the possible values of $a_{r,c-1}$ with the new vector of the residual alleles updated.

```
static void heterozygote (unsigned r, unsigned c, double
   probl, [...], COUNTTYPE * R)
{
    lower = fmax(0, lower);
    upper = fmin(res[r], res[c]);
    if(c > 2) for (arc = lower; arc <= upper; arc++) {
        memcpy(Rnew, R, Rbytes);
        // decrement residuals for the current value of arc.
        resn[r] -= arc;
        resn[c] -= arc;
        heterozygote(r, c-1,
                     probl+lnFact[arc],
                     [...]
                     Rnew);
    } // for arc
    //[...]
```

b) $c = 2$ and $r > 3$

This time we have analyzed the whole row and, it remains only to understand how many different values the first two genotypes of the same row can assume. For each value of a_{r2} between the two extremes (lower and upper), the last value will be

$$a_{r1} = min(R_r - a_{r2}, R_1)$$

At this point the contribution of these two genotypes is added to the previous probability and it is passed to the line immediately higher, that is to the next node of the network, through the function *homozygote()* with parameters $[r-1, f_p + f(a_{r2}) + f(a_{r1}), R']$, in which R' is constructed by subtracting a_{r2} from R_r and R_2; moreover, a_{r1} will be subtracted from R_r and R_1.

```
if(c==2){
    if(r > 3)
        for (ar2= lower; ar2 <= upper; ar2++) {
         memcpy(Rnew, R, Rbytes);
        // decrement residuals for the current value of arc.
        resn[r] -= ar2;
        resn[c] -= ar2;
        // The value of ar1 is now fixed, so no need for any
    more calls to heterozygote in this row
        ar1 = fmin(resn[r], resn[1]);
        resn[1] -= ar1;
        resn[r] -= ar1;
        homozygote(r-1,
                   probl + lnFact[ar2] + lnFact[ar1],
                   [...],
                   Rnew);
        } // if r > 3
 //[...]
}
```

c) $c = 2$ and $r = 3$

We have therefore reached the third and last possibility, we have almost drawn the whole contingency matrix and, in the variable f_p, we have accumulated part of the probability we need. Row number 3, starting from the top, ends by analyzing, for each value of a_{32}, what remains for a_{31} and updating the variable f_p.

$$a_{31} = min(R_3 - a_{32}, R_1)$$

$$f' = f_p + f(a_{31}) + f(a_{32})$$

We are therefore now in a situation similar to that with 2 alleles, whose count was indicated with n_1 and n_2. The new values for the 2 remaining alleles will be $n'_1 = R_1 - a_{31}$ and $n'_2 = R_2 - a_{32}$. At this point it is easy to obtain the first 3 values of the matrix (a_{11}, a_{21}, a_{22}), making sure that they are positive. Therefore, the algorithm can end with the actual calculation of the probability corresponding to this table, therefore its path in the network, considering that: K_p was the initial constant, f' contains the probability deriving from the rest of the table and a_{11} and a_{22} are homozygous and must be added to the factor d.

$$\ln P = K_p - f' - f(a_{11}) - f(a_{21}) - f(a_{22}) - (a_{11} + a_{22}) \ln 2$$

Finally, the exponential of the obtained value is calculated and, if it is verified that it is lower than the probability of the observed marker, it is added to the final **p-value** to be returned in output.

```
if(r==3) // and c = 2
{
    for(a32 = lower; a32 <= upper; a32++) {
        a31 = fmin(res[1], res[3]-a32);
        probl3 = probl + lnFact[a32] + lnFact[a31];
        // get residual allele counts for two-allele case
        res1 = res[1] - a31;
        res2 = res[2] - a32;

        if(res1 > res2) {
            resTemp = res2;
            res2 = res1;
            res1 = resTemp;
        }
        // Now process two-allele case
        tableCount += res1/2 + 1;
        for(a11 = 0; a11 <= res1/2; a11++) {
            a21 = res1-a11*2;
            a22 = (res2-a21)/2;
            problT = probl3 + lnFact[a11] + lnFact[a21] +
  lnFact[a22];
            dT = a11 + a22;

            problT = constProbTerm - problT -dT * M_LN2;
            prob = exp(problT);

```

```
26              //Now process the new values of prob and stat
27              probSum += prob;
28              if(problT <= maxlPr) pPr += prob;
29              //[...]
30          } // for a11
31      } // for a32
32  } // if r == 3
33  }
```

The complete algorithm ends when the first *homozygote()* function returns, so as to be sure that it has covered all possible paths on the network. Certainly, the one of Engels (2009) is a very complex and articulated algorithm, which however manages to carry out the calculations little by little and not weigh down the overall complexity. The use of recursion is a fundamental aid in this, albeit limited to the size of the stack available. With some targeted changes, we managed to generalize what has just been seen in the specific case of the X chromosome.

5.2 X Chromosome

Starting from the work just discussed, which was the cornerstone of the study for this whole project, we tried to extend an algorithm studied in detail to an even wider context. We have repeatedly mentioned the fact that the X chromosome is different from all the others, since its quantity changes according to gender, something that has often been overlooked or bypassed in its analysis regarding HWE. Arriving almost at the end of the thesis, it is easy to understand how the exact test for the deviation from the proportions of Hardy-Weinberg sees this difficulty of calculation increase exponentially with the increase of the complexity of the problem. Up to the bi-allelic case, still relatively small and manageable, we have seen that there are different ways of approaching and solving the question. At a time when the number of alleles considered is larger than 2, it was preferred, for simplicity, to test only female individuals and take the final result for both sexes for good. Even in the analysis with only 2 alleles it turned out that this approach is very questionable and the results can often be different from the real ones (Graffelman and Weir, 2016). For this reason, we wanted to take a path that saw the solution of this problem, taking advantage of the very solid foundations by Engels (2009), through a network algorithm which, in addition to being functional in the results, is also very fast. Already in the previous chapter this was introduced, with only 2 alleles, bringing out the logic that we wanted to exploit to do it. Let's see a little more in detail how the **network algorithm for the X chromosome** can be generalized for k alleles.

The situation we are in now involves different upstream considerations depending on the *gender*. The X chromosome is present once in male individuals and twice in female individuals. Contrary to before, this time when analyzing genotypes this aspect must be taken into account. If we have a sample of n individuals, n_m males and n_f females, in a context with k alleles we will have a background that provides a **vector** for the first ones, in which for each allele $i = 1, \ldots, k$ how many individuals n_{mi} contain it, while a **matrix** for the latter, in which each cell n_{fij} shows the number of women with alleles i and j, both in the range $1, \ldots, k$. The matrix is similar to that which for autosomes includes all individuals together; also in this case, only the lower triangular part is taken into account.

$$\begin{bmatrix} n_{m1} \\ n_{m2} \\ \vdots \\ n_{mk} \end{bmatrix} \begin{bmatrix} n_{f11} & & & \\ n_{f21} & n_{f22} & & \\ \vdots & \vdots & \ddots & \\ n_{fk1} & n_{fk2} & \ldots & n_{fkk} \end{bmatrix}$$

The total number of alleles will be $n_t = n_m + 2n_f$, while every single quantity of the generic allele i will be indicated with n_i. The conditional distribution of the genotype counts given the allele counts, taking up the equation (2.15) seen in the first chapters, is

$$P(N_{fij} = n_{fij} \cap N_{mi} = n_{mi} | n_1, ..., n_k) = \frac{n_m! n_f! 2^{n_f - d} \prod_{i=1}^{k} n_i!}{n_t! \prod_{i=1}^{k} n_{mi}! \prod_{i \geq j} n_{fij}!} \tag{5.4}$$

where $d = \sum n_{fii}$ is the total number of homozygous females.
After having already theoretically discussed these concepts, it is easy to understand how the final p-value for the HWE must be calculated: with the number of alleles fixed, a lot of samples can be constructed in addition to the one being analyzed; therefore, in theory, for the exact statistical calculation it is necessary to list all the possible matrices and vectors. Once again we split the equation so as to make some constants clearly visible, which can be calculated initially and remain fixed, and other factors which instead vary with the variation of the tables considered.

$$P(N_{fij} \cap N_{mi} | n_1, ..., n_k) = \underbrace{\frac{n_m! n_f! \prod_{i=1}^{k} n_i!}{n_t!}}_{\exp K_p} \underbrace{\frac{2^{n_f}}{\prod_{i=1}^{k} n_{mi}!}}_{\exp \frac{1}{K_{males}}} \frac{1}{\prod_{i \geq j} n_{fij}! 2^d} \tag{5.5}$$

The reason why *exp* appears before each constant is because the calculations will be carried out, also in this case, on a logarithmic scale to reduce the load due to factorials and multiplications. The first term K_p is fixed and independent of the sample. The second term K_{males}, written as a relationship in the formula, is

clear and defined once we know what is the value of the elements in the vector of the male alleles $n_{m1}, \ldots, n_{mk}$. The last multiplicative element changes as the sample considered varies, depending on the female individuals. The logarithm of the probability of the observed marker will be equal to

$$\ln P(N) = K_p - K_{males} - \sum_{i \geq j} f(n_{fij}) - d \ln 2 \tag{5.6}$$

where f is a function such that $f(i) = \ln(i!)$, useful to extract the last factor of equation (5.5). Although what has been said may seem repetitive with respect to the previous explanations, it is of fundamental importance to note the details and the differences compared to the other cases. We are basing the study on the adaptation of what has already been done by Engels, making the female genotype matrix the analogue of the total contingency table for autosomes. The difference that arises from the male hemizygous is inserted in a subtle way in the algorithm described in the previous paragraph, thus managing to exploit its consolidated efficiency. After the premises, we describe in more detail how the whole procedure works, trying to avoid the repetitions of the functions that perform the same work both here and with the autosomes.

The algorithm starts from the assumption of receiving the following parameters as *input*:

- vector of total alleles count
- number of male and female individuals
- number of alleles
- test statistic for the observed sample, calculated using (5.4)

After this, in short it is possible to summarize the essential points on which we will focus and which are the main steps to follow in order to carry out the algorithm:

1. calculation of the constant K_p, which will remain fixed until the end, and initialization of variables R (female alleles vector) and R_{males} (male alleles vector) to the number of **total** alleles available
2. call *recursiveEnumeration()*, a function that allows you to list which values R_{males} can take (and for complementary also R) in each of the possible contingency tables
3. for each R_{males} the constant K_{males} is calculated and *homozygoteX()* is launched
4. considering the vector R as if it were the overall autosomal one, the algorithm proceeds as in the previous paragraph in which for each possible combination

of R_{males} a network for females is generated to be crossed in all possible paths, thanks to the recursive calls between the *homozygoteX()* and *heterozygoteX()* functions

5. as soon as the first call to *homozygoteX()* returns, proceed for another value of R_{males} by returning to point 3

6. when all possible males have been analyzed, *recursiveEnumeration()* returns and the algorithm ends

Considering that even here we have exploited the Engels code, adapting our modifications, we will see in `C` how this is reflected in the focal points, explaining in detail something more than the points just listed.
The main function is `xChrom()`, which receives the parameters described above as input and starts the algorithm. The points to note are the new parameters, such as `alleleVect` (corresponding to R_{males}) and `mf` (number of males and females), but above all the **new content** of the constant `constProbTerm` (K_p), which changes with respect to the autosomal context. Finally you can notice the `maxlPr` parameter will contain the probability value of the observed polymorphism, to be used as a threshold beyond which the values for calculating the p-value are excluded, and the call to `recursiveEnumeration()`. The `alleleVect` array is defined as the number of alleles +1 to favor the way in which it was decided to build the last function.

```
void xChrom (int * rm,
             int * mf,
             int * rk,
             double * robservedVals, // observed stats
             double * rPvals, // computed P values
             [...]
             )
{
    // Set up global variables used during recursion
    nAlleles = *rk;
    male = mf[0];
    female = mf[1];
    pPr = 0;
    //[...]
    Rarray = Calloc(*rk * *rk * (*rk-1)/2, COUNTTYPE);
    alleleVect = Calloc(*rk * *rk * (*rk-1)/2, COUNTTYPE);
    for (int i = 0; i < nAlleles; i++) {
            Rarray[i] = rm[i];
            alleleVect[i] = rm[i]+1;
```

```
20      }
21      //[...]
22      int  nGenes = 0;
23      for(int i = 0; i < nAlleles; i++) nGenes += rm[i];
24      ntotal = nGenes/2;
25      // Get constant term for Prob
26      constProbTerm =  0;
27      for (int i = 0; i < nAlleles; i++) {
28          constProbTerm +=  lgammafn(rm[i] + 1);
29      }
30      int nt = male + 2*female;
31      constProbTerm +=  lgammafn(male+1) + lgammafn(female+1)
    - lgammafn(nt+1);
32
33      double oneMinus = 0.9999999;
34      //[...]
35      maxlPr = log(robservedVals[1]) * oneMinus;
36
37      recursiveEnumeration(1);
38      //[...]
```

The adaptation of the algorithm to the situation with the X chromosome has seen us focus only on the exact test that follows the probability as defined in the equation (5.4). For the moment, references to histogram extraction or other tests have been neglected, which are instead taken into consideration by Engels. This may then be a starting point for future work for completeness.
However, let's now conceptually see how `recursiveEnumeration()` works. This can be thought of as a *depth visit to a tree*: starting from the root, every time you go down one level towards the leaves you are considering an extra allele; at each level all possible combinations of values will be present depending on the count present in the vector of total alleles. When it comes to the leaves, it is verified that the number of alleles in that leaf is equal to the number of males under analysis and that the remaining count is sufficient to cover the female genotypes. These are two simple and quick checks that allow you to avoid going through an entire network to test the contingency matrices if there are not the right numbers and therefore make the whole procedure fast. Let's see a small example to give a practical idea of what has been said. Suppose we have a sample with 6 individuals, 4 males and 2 females, and the number of alleles $k = 3$. In this type of algorithm it is a good rule, to speed up the calculations, to order the vector of alleles count in **descending order** always obtaining the same result as the original case. If we call the alleles

a_1, a_2 and a_3 and their count is

$$\begin{bmatrix} 4 & 3 & 1 \end{bmatrix}$$

then we will have a recursion tree like the one in Figure 5.6.

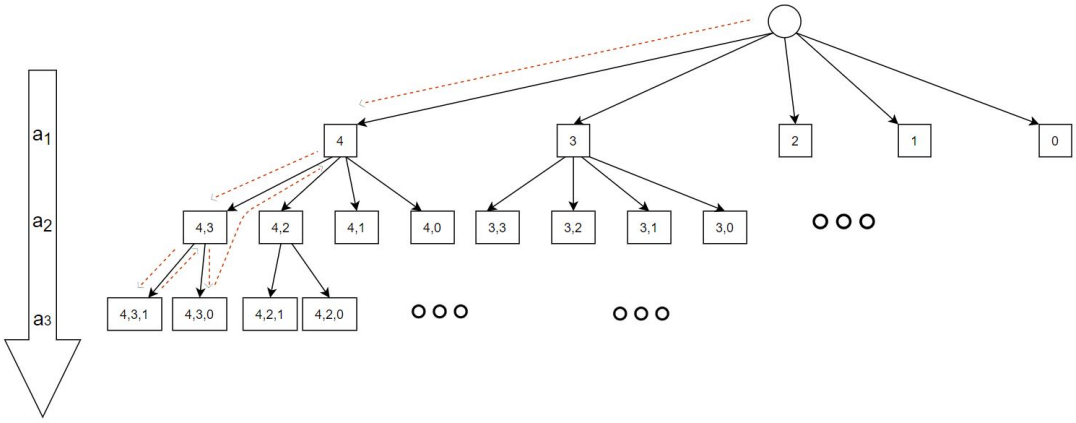


Figure 5.6: Depth visit for male alleles

The tree starts from the top root and considers the possible quantity for each allele a_i at level i ($i = 1,2,3$). Since a_1 is present at most 4 times, all possibilities from 4 to 0 are considered; the same happens with a_2 which is at most 3 and a_3 which is 1 or 0. The **red dotted arrows** show us the order in which the construction of the tree takes place in the first part. After which is the procedure continues in the same way for the other values of the alleles. In the `C` code, the function accepts as input the `index` parameter, corresponding to the *level* being analyzed in the tree. By the time it reaches the leaves (`index==nAlleles`), we check that we have **enough residual alleles** for the female individuals. If the control is passed, then the `constMales` constant is calculated, that is K_{males} in the formula (5.5), the elements of R (here `Rarray`), are set for the total alleles for the female genotypes and `homozygoteX()` is launched to start exploring the net.

```
1 static void recursiveEnumeration(int index){
2 int value = alleleVect[index-1];
3 int nGenes;
4 double constMales;
5 while(alleleVect[index-1]>0){
6     constMales = 0.0;
7     alleleVect[index-1]--;
8
9     if(alleleVect[index-1] < 0)
```

```
10          return;
11      else  {
12      if(index == nAlleles){  // last one
13          if(sum()!=male){
14              continue;
15          }
16          else {
17          if(enoughFemale()){
18              if (nAlleles == 2) {
19                  twoAlleleSpecialCaseX();
20              } else {
21                  nGenes = 0;
22                  for(int i=0;i<nAlleles;i++){
23                      constMales += lgammafn(alleleVect[i]+1);
24                      Rarray[i] -= alleleVect[i];
25                      nGenes += Rarray[i];
26                  }
27                  constMales -= log(2)*(nGenes/2);
28                  homozygoteX(nAlleles,0,constMales,Rarray);
29                  // Reset Rarray
30                  for(int i=0;i<nAlleles;i++){
31                      Rarray[i] += alleleVect[i];
32                  }
33              }
34          }
35          else {
36              continue;
37          }
38          }
39      }
40      else {
41          recursiveEnumeration(index+1);
42      }
43      }
44  }
45  if(index!=1){
46      alleleVect[index-1] = value;
47  }
48  return;
49  }
```

Clearly as the number of alleles k increases and the number of individuals increases, this tree will be deeper and/or wider. In this case it is possible to make further

changes to speed up the process, for example by avoiding reaching the leaves if it is already understood that it will be the wrong way, for the moment we have adopted a solution that is as intuitive as possible and that gives reliable results. In the previous example, only 8 of the leaves are actually combinations of possible alleles for the males and, from each of them, the network algorithm can start, which sees the subtraction of the male alleles as the starting node. Whenever the *homozygoteX*() function is launched after reaching the lowest level of the tree, the situation that presents itself is similar to that of Figure 5.7, in which we have avoided representing the various paths for each possible combination.

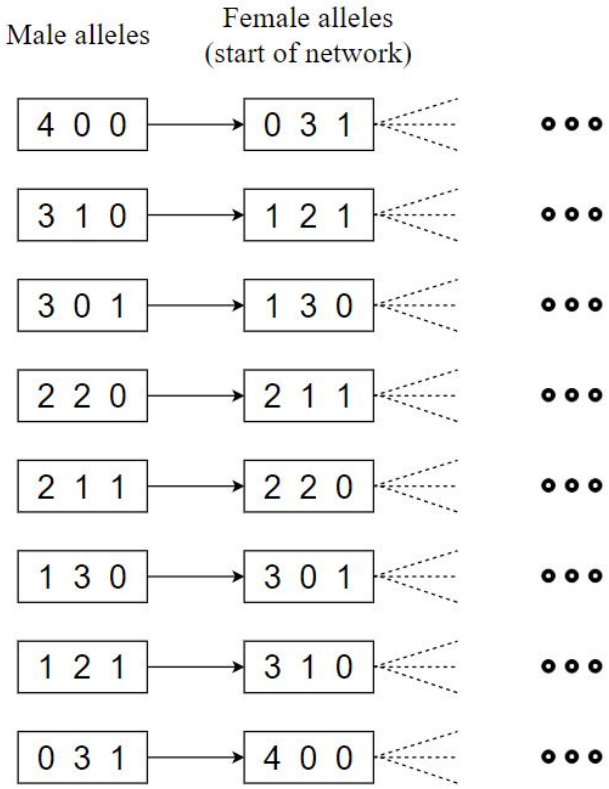


Figure 5.7: Starting exploring network for X chromosomal context

The homozygoteX() and heteroxygoteX() functions work exactly the same way as in the previous paragraph, so we avoid repeating them again. The only difference is that both receive one more parameter, K_{males}, in order to pass it as a value until the end of the algorithm. At the bottom of the *heterozygoteX()* function, then at the end of a path, when in the contingency table for female genotypes we arrive at the case in which r(ow) = 3 and c(olumn) = 2, this constant is subtracted from

the logarithm of the final probability, as per formula (5.6).

```
1 problT = constProbTerm - problT -dT * M_LN2 - constMales;
2 prob = exp(problT);
3 probSum += prob;
4 if(problT <= maxlPr) pPr += prob;
```

Finally, if the probability is less than or equal to `maxlPr` it adds up to the final p-value, which will be stored in `pPr` and returned as the program output.

The first attempts at use were aimed at verifying that what was realized was in line with what already existed, therefore analyzing the differences in results obtained by considering only 2 alleles. This led to good results, with fairly small differences, such as in the order of 10^{-7}. From here we started to consider the instrument as valid even for more complex samples and, although there are currently no algorithms to verify the reliability of the exact p-value, you can test the results using the permutation test, whose output value is expected to be similar in order of magnitude to what we have.

5.3 Performance in terms of CPU

To understand the importance of this chapter, it is necessary to evaluate the speed of execution, as well as the reliability of the results, in using the new algorithms rather than the most common ones. To do this we have taken two databases, one for autosomes and one for the X chromosome, which contain tri-allelic variants so that we have clear advantages using the network algorithm. The same premise explained in the previous chapter is also valid here, that is to say that, while the complete enumeration can be called from `R` directly, the network algorithm provides the call to an external executable program, written in `C`, with the resulting input and output operations influencing, even if slightly, the performance. Nevertheless we will see how the impact of the new tools is extraordinary and makes these delays infinitesimal.

5.3.1 Autosomal variants

Regarding the **autosomes**, the database chosen is the same one we have seen before, which contains 107 individuals from a population coming from Tuscany, Italy, and whose chromosome 7 is analyzed (The 1000 Genomes Project Consortium, 2015). In this case the distinction between genders does not count. Initially there were 4164 markers, of which only 3907 have been extracted, which are those with

3 alleles. Although Engels' algorithm works well even with a large number of alleles, this was done to evaluate not only the performance but also the truthfulness of the p-values obtained with an exact test. The `HWTriExact()` function of the `HardyWeinberg` package allows to calculate the exact test statistics for tri-allelic variants, both for autosomes and X-chromosomes. The analysis we did was to see how long it took this function to analyze a different number of markers ranging from 1 to the total number, proceeding for powers of 10. The same procedure was applied to the same data using the algorithm by Engels and the results are represented graphically in Figure 5.8, whose exact times are shown in Table 5.1.

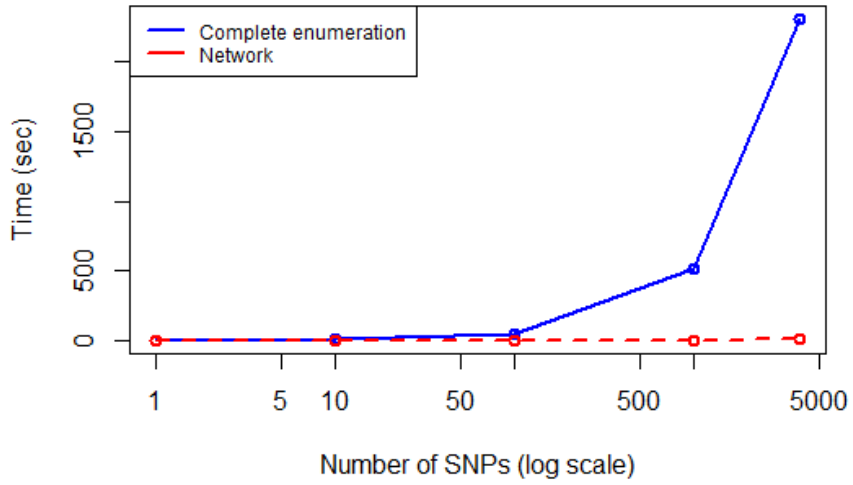


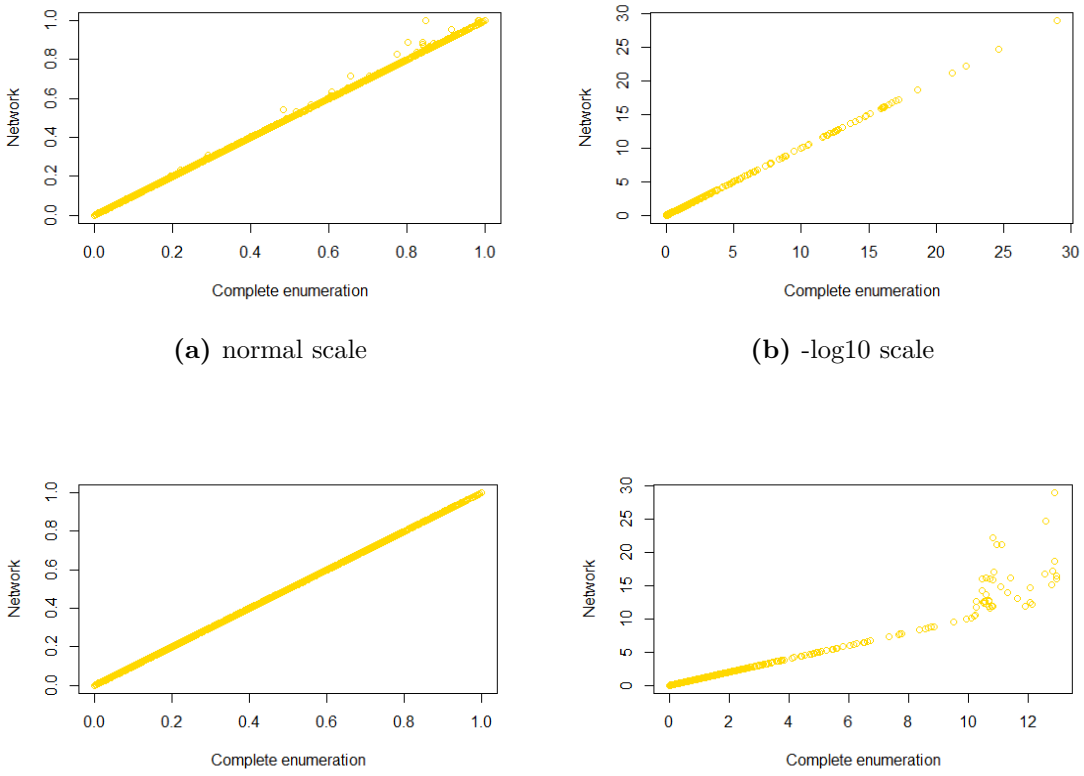
Figure 5.8: Performance results for 3-allelic variants

	Complete	Network
1	0.011	0.121
10	6.368	0.114
10^2	46.540	0.115
10^3	510.224	1.014
All	2309.469	4.259

Table 5.1: Performance results for autosomal variants with 3 alleles (in seconds)

The figure that jumps out at once is the order of magnitude of the final results, especially for the entire dataset. If the usual algorithm takes about 38 minutes, the network algorithm in just over 4 seconds leads us to the same result. Now the network is fully utilized and we understand why we should opt for this.

From the point of view of the final p-values obtained, this example brings out a very interesting problem related to ties, introduced in the final part of the Section 4.1.3. From an immediate analysis in which we compare the results of the network algorithm and the complete enumeration a maximum difference of 0.15285482 emerges, visible in Figure 5.9a and 5.9b. Here a large difference emerges, for which one of the two algorithms can testify a deviation from the HWE while the other one cannot. Going into detail in the marker with this problem, the permutation test was also used to verify which of the two algorithms could be more reliable. The permutation test (`HWPerm.mult`) gave a result very similar to the full enumeration (`HWTriExact`), which suggests that the network algorithm gives a wrong solution. Looking better at the autosomal variant in analysis, the permutation distribution suggests the observed sample is the most likely sample. If that is true, the p-value should be 1, like in the network algorithm, and it is not. Thus it seems that samples with the same probability as the observed sample are "not counted", whereas they should. They might be "missed" because the probability is not regarded smaller than that of the observed sample.
Therefore, playing a bit with the `eps` parameter of the `HWTriExact` function, it turns out that if $eps = 10^{-13}$ then the maximum difference between the two algorithms drops to $4.998338e-08$. This is shown in Figure 5.9c and 5.9d, which basically show very small difference appear, especially in the most significant results (up to 10^{-10}). This can be expected due to finite precision and confirms how much the **ties** problem affects the algorithms and their results.
In this case, therefore, the network algorithm has better coped with this situation, reaching a higher precision than the complete enumeration has to be dictated "by hand" by the programmer. This does not detract from the fact that, in another context, it is possible to find the opposite situation and that, therefore, this problem must be analyzed even more in detail to minimize its negative effects.



(a) normal scale

(b) -log10 scale

(c) normal scale

(d) -log10 scale

Figure 5.9: Complete enumeration and network outputs comparison for autosomal variants with 3 alleles without (a and b) and with *eps* parameter (c and d)

5.3.2 X chromosomal variants

We now can move on to the much more interesting analysis of the **X chromosome**. The chosen database contains 2979 tri-allelic variants, whose population is divided into 56 males and 48 females. This data are used by Graffelman and Weir (2018) in their paper, in which they analyze the behavior and performance of the full enumeration algorithm on all these variants. After the high computational effort that allows this study in a lot of hours, the authors themselves say: "We expect that large computational gains can be achieved using the aforementioned computational improvements and by re-programming the algorithms in the `C++` computer language". This is exactly what we tried to do, let's see how. The study is similar to the previous one, where the `HWTriExact()` function is still used for the complete enumeration and the `C` executable for the network algorithm. Also this time the

analysis has been done with a number of variants that proceeds for powers of 10 and the timing has been evaluated. What came out is represented in Figure 5.10 and Figure 5.11, with a zoom in the first 100 markers. The result is of extraordinary importance in terms of performance.

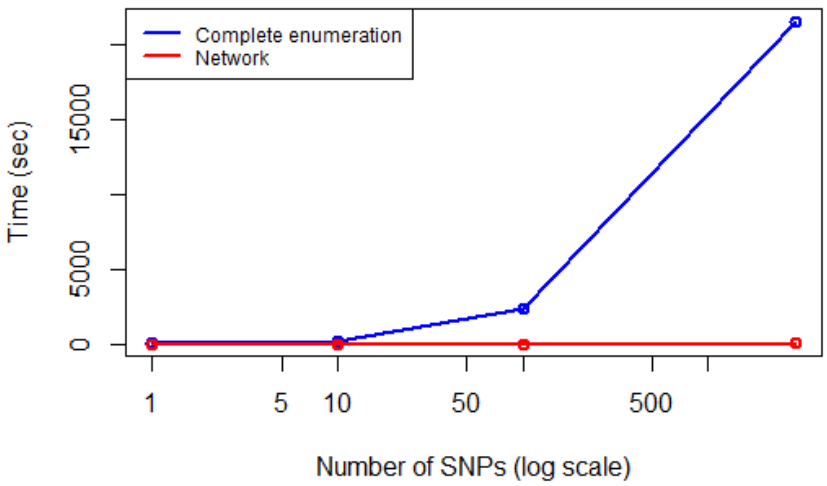


Figure 5.10: Performance plot for X chromosome with 3 alleles (whole dataset)

The execution times are shown in detail in Table 5.2. The new algorithm reduces the calculation time exponentially compared to the complete enumeration. The passage from the order of **hours** to **seconds** shows how the new instrument can help to obtain equally accurate results in a short time.

	Complete	Network
1	95.265	0.106
10	185.526	0.114
10^2	2297.976	4.491
All	~6 hrs	11.191

Table 5.2: Performance results for X-chromosomal variants with 3 alleles (in seconds)

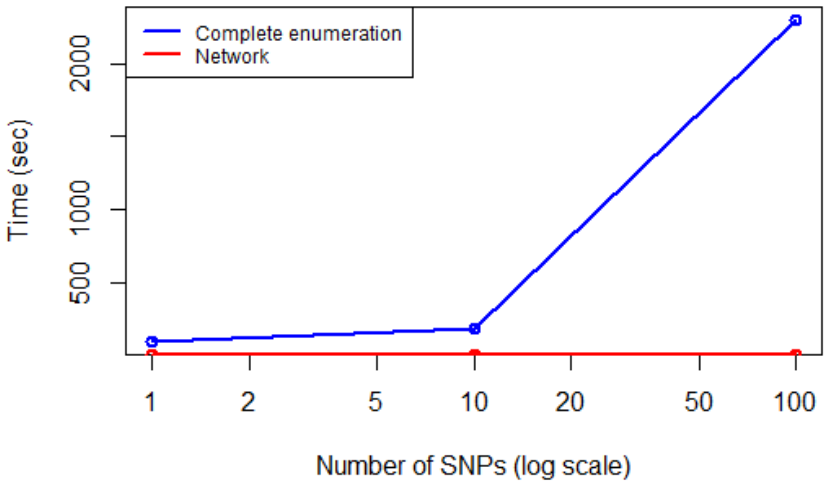
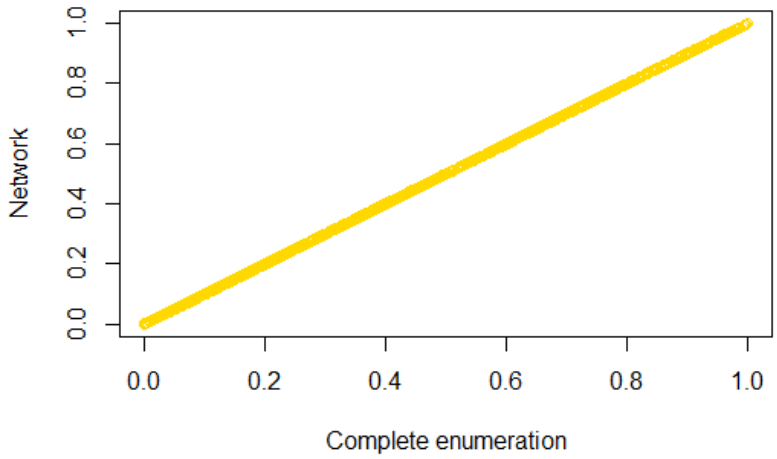


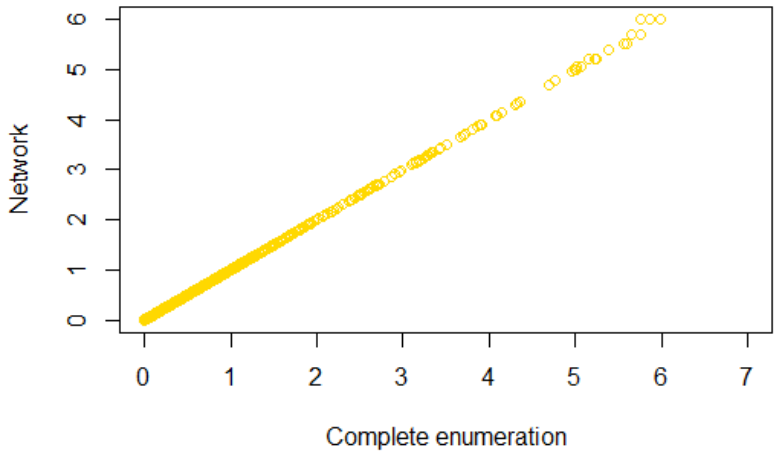
Figure 5.11: Zoom only on first 100 markers

In fact in Figure 5.12 the p-values resulting from the two techniques have been plotted and the maximum difference is $2.393676e-05$. This denotes a work of enormous precision as well as efficiency and the full potential of which can be exploited.

The permutation test was not carried out because, as seen for only 2 alleles, it is even slower than the complete enumeration. Despite this the times are comparable for a number of alleles greater than 5, for which the algorithm works well but takes too long. Although the accuracy is lacking, the permutation test can be a useful weapon in these cases with the instruments present today.



(a) normal scale



(b) -log10 scale

Figure 5.12: Complete enumeration and network outputs comparison for X-chromosomal variants with 3 alleles

Chapter 6

Empirical studies

To conclude the speech about the new algorithm we want to present a complete final example, on a real dataset where several markers are analyzed, in order to show how the new method should be used. The strength of what has been explained in the previous chapters lies in being able to study the presence of a deviation from HWE in a population whose X chromosome is analyzed. This can now be done for more than 2 alleles in a more efficient way. The database chosen is the TSI, the same as seen in the previous chapter, which contains a population from Tuscany but of which we have extracted the X chromosome this time (The 1000 Genomes Project Consortium, 2015). The result is a total of 4219 variants, of which we have taken into account at the moment only those up to 5 alleles, and 107 individuals (53 males and 54 females). This has been done not because the network algorithm is not able to analyze the remaining markers, but because we believe that the efficiency with more alleles is not equal to these great advances made so far. The times have been reduced incredibly when we analyze variants with 3, 4 or 5 alleles compared to the usual tools, while for higher numbers we think that at the moment perhaps using the permutation test is more appropriate in terms of performance, although not precision. Despite this it is always possible to launch the new algorithm but with a very long time.

Having said that, let's see step by step how to work with the new instrument in `R`. Of course, the implementation may vary depending on the format of the data you receive as input. In this case, after loading the data with the `load()` function we remove the variants that are in a region where there is pairing between X and Y chromosomes (`par.region==1` or `par.region==2`). Being only interested in the X chromosome, these variants are not interesting because they behave as autosomal variants.

```
load("./variantsTSICHRX.rda")

data <- c()
indexes <- c()
for (i in 1:length(variants)) {
  if(variants[[i]]$par.region == 3 | variants[[i]]$par.region == 0){
    indexes <- c(indexes, i)
  }
}

#107 individuals, only X  (53 males, 54 females)
maleFemale <- c(53,54)
data <- variants[indexes]
```

From here our data are presented in the following format,

```
data[1]
[[1]]
[[1]]$k
[1] 3

[[1]]$als
[1] "0" "1" "2"

[[1]]$pos
[1] 2703501

[[1]]$id
[1] "rs11371846"

[[1]]$alt
[1] "AT,T"

[[1]]$ref
[1] "A"

[[1]]$X.f
   0 1  2
0  3 0  0
```

```
1  2 0  0
2 19 3 27

[[1]]$X.m
 0  1  2
12  0 41

[[1]]$par.region
[1] 0
```

where

- `$k` is the number of alleles this polymorphism has
- `$als` gives the names of the existing alleles
- `$pos` gives the position along the chromosome in base pairs
- `$id` is the RS identifier of the polymorphism
- `$alt` are the alternate alleles
- `$ref` the reference allele for the polymorphism
- `$X.f` the triangular table of genotype counts for the females
- `$X.m` the vector of gentoype counts for the Males.
- `$par.region` if it is in pairing with Y chromosome.

The next step is to extract from the data only variants with a number of alleles less than or equal to 5 and store them in a structure called `dataUpToFive`. Finally, for each of the markers present we calculate the p-values using the data structure just created and its attributes. In particular, we extract the matrix of female genotypes and the vector of male alleles that serve to have a precise count of each allele and the probability of the observed sample. The union of the two things, with the vector of the counts ordered in a decreasing way for reasons of efficiency, is the input that serves the executable to work. Finally you read the data from the output file and store it in a vector.

```
1 dataUpToFive <- c()
2 indexesUpToFive <- c()
3 for (i in 1:length(data)) {
```

```
  if(data[[i]]$k <= 5){
    indexesUpToFive <- c(indexesUpToFive, i)
  }
}
dataUpToFive <- data[indexesUpToFive]
otherData <- data[-indexesUpToFive]

pvalsUpToFive <- c()
for (i in 1:length(dataUpToFive)) {
  m <- dataUpToFive[[i]]$X.m
  f <- dataUpToFive[[i]]$X.f
  ftot <- alleleCounts(f)
  tot <- m+ftot
  prob <- observedProb(f, m, maleFemale)
  alleles <- sort(tot, decreasing = T)
  countsUpToFive <- c(alleles,prob)
  nAll=dataUpToFive[[i]]$k
  nInput = 1
  write.table(countsUpToFive, file="countsUpToFive.txt", row
   .names=FALSE, col.names=FALSE)
  str <- paste("HWx.exe", nAll, nInput, maleFemale[1],
   maleFemale[2], "countsUpToFive.txt", "outpvaluesUpToFive.
   txt")
  system(str)
  pvalsUpToFive <- c(pvalsUpToFive, read.table("
   outpvaluesUpToFive.txt"))

}
```

A total of 3864 markers with 3, 4 and 5 alleles were analyzed. In Table 6.1 it is possible to see exactly how many variants there are for each number of alleles and the time needed to analyze each of them.

Number of alleles	Number of markers	Time (minutes)
3	3496	6.688
4	333	9.169
5	35	13.465
Total	3864	29.332

Table 6.1: Performance of the network algorithm

It is easy to see how the complexity of the problem increases with increasing number of alleles, so analysing fewer markers still costs more time. Despite this,

we are very satisfied with these results. The reliability of the algorithm with 3 alleles has already been positively evaluated in the previous chapter. Nevertheless we wanted to try one last time to see if the results obtained with more alleles are reliable or not. The only test that allows us to approach this evaluation is the permutation test, which is slower than the network algorithm but gives us a good truthfulness index. We used this test only in the 35 variants with 5 alleles and we saw the difference with the newly obtained p-values. In a time of 15.274 minutes the function `HWPerm.mult()` gave us back the p-values for the variants with 5 alleles with a maximum difference of the same calculated with the network algorithm of 0.01903.

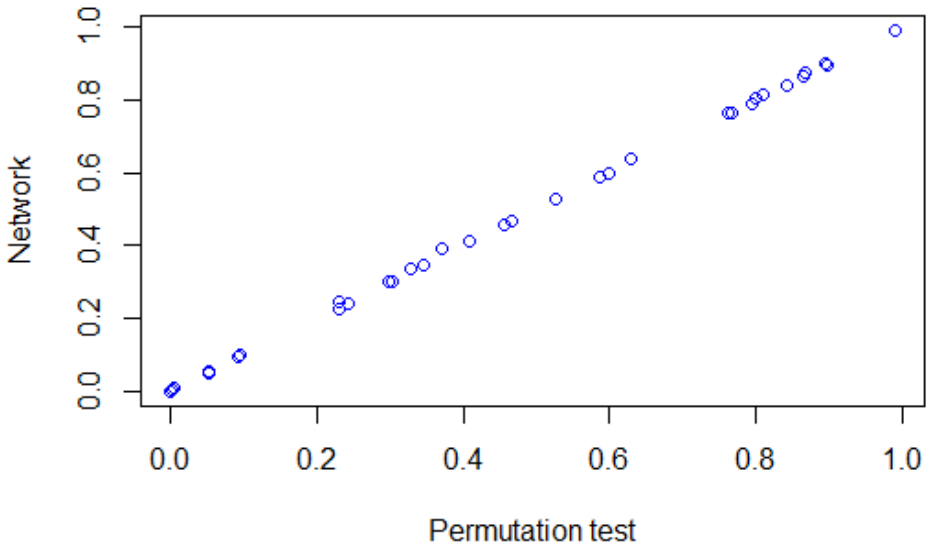


Figure 6.1: Permutation test and network algorithm p-values in comparison for 5-allelic variants

The graph in Figure 6.1 shows the two algorithms in comparison. We can therefore conclude that, even with a higher number of alleles, the new instrument appears both efficient and reliable.

Chapter 7

Conclusion and Discussion

In this thesis we wanted to analyze with the exact test the deviation from the Hardy-Weinberg equilibrium of X chromosomal markers, taking into account both male and female gender. The study began by introducing the fundamental definitions to understand the topic, so as to facilitate the subsequent reading even to those who are not in the field. In particular, the biological definitions of DNA, gene and alleles are pivotal points to understand why this thesis came to life and what are the reasons that make the project interesting under different aspects that do not stop at the computer field, but also enclose the medical, biological, forensic, as well as the statistical one that can be seen as a link between the first and the others. Subsequently, precisely, we have gone further in the technical field by introducing the theory and from the statistical point of view that lies at the basis of important analyses such as these and how it is actually possible to carry them out in a practical way. The definitions of probability, statistical test and p-values are those which, in the end, produce numbers with very precise interpretations to give sensible conclusions. Since the study of HWE is of extraordinary importance both in statistical genetics and in all the areas mentioned above, being able to help those in need by providing useful and fast tools has been a great challenge full of motivation and has been gladly accepted.
If there are already software and functions suitable for a complete X chromosome analysis for bi-allelic variants, we were able to expand this potential to multiallelic markers, whose balance has so far only been tested for females, simplifying the problem by far. In fact, while for autosomal chromosomes there is a kind of symmetry between the two genders, the X chromosome, present only once in male hemizygous individuals, makes statistical calculations much more complicated, making the set of tables to explore much larger and adding more factorial terms in the equations. Statistical techniques seen in theory, such as the chi-square test, are not adequate with many alleles because of the scattered data and low counts. The only really accurate test is the exact test, but it has the disadvantage of a high

computational load. The new network algorithm, which manages to reconcile power and accuracy, can be a starting point for achieving the improvements expected in the article by Graffelman and Weir (2018). The result obtained was very positive from both points of view, managing to solve problems that require several hours of computational time in the order of a few seconds. Just look at the final example in Chapter 5, where a dataset of tri-allelic X chromosomal variants goes from an analysis time of about 6 hours to just over 11 seconds between old and new tools. The big improvement of the new algorithm has two fundamental elements at its base: on the one hand there is the passage from the `R` environment to that of `C` and `C++` languages that are more suitable for the execution of complicated algorithms and that make much use of recursive cycles and techniques; the second aspect is precisely this, recursion. The strong point of the network algorithm is to weave a very wide and deep network that has the task of passing reusable calculations through the recursive *homozygoteX* and *heterozygoteX* functions in such a way that the computational effort is minimal and unnecessary counting is avoided. The combination of these factors has resulted in increased power and reliability.

All this can only be considered a "starting point" because, as seen, the work is not completely finished and can still be improved. The first aspect to improve is surely the 5 alleles barrier, i.e. as far as the algorithm works well enough to be considered valid. Although the speed depends very much on the distribution of the alleles in the observed marker, it has also been seen as for a number of alleles greater than or equal to 6, the exact exploration of all the tables, despite the recursion and the network, is very slow. The cue is very interesting, however, to work on it in the future, avoiding to repeat counts that lead to the same results for, for example, two different sets of male individuals. For the moment the structure of the algorithm is very simple and adapts almost naturally to what Engels (2009) has already explained, but it is certain that few further improvements would give more strength to what has been done, further extending its usability.
The network algorithm developed for the X chromosome in this thesis could be adapted to allow for faster computation of alternative statistical tests, such as the likelihood ratio test or the chi-square test. In fact, if we only used probability as an index to extract the exact test, Engels generalizes its algorithm to a wider range of statistical tests, which also allows you to notice the differences between them and easily extract histograms to represent the results. For the moment these aspects have been neglected, focusing instead on the novelties due to the male gender in the X chromosome and reliability compared to existing technologies.
Another aspect that has come up is the issue of *ties*. We have shown that the numerical differences between the algorithms often derive from them, which affect the final result for differences in accuracy even infinitesimal. Further work to identify and accommodate for ties could improve the accuracy of test procedures in a way that makes the work even more reliable than already described.

The last element that should not be underestimated is that of a full integration of the new algorithm within `R`, exploiting the functions written in `C` but importing them in the same development environment, so that they can be imported in the `HardyWeinberg` package and be available to all developers. For the moment the use of the executable has been enough to extract data within the project, but this step is of fundamental importance disseminate the new algorithm.

We conclude therefore saying that the development of this new algorithm has been important and useful in every aspect. We are already in the process of writing an article to report all the improvements achieved with respect to the existing tools that have been shown. In this way the work will remain exposed in a more synthetic way, showing only the really important and innovative contents. Moreover, we hope it will be the cue for many future improvements, all the more so given the impact this has in applied scientific contexts such as genetics, forensics and epidemiology among others, and as can be seen from the presented analysis of empirical datasets; not only toy examples. We are aware of the excellent progress we have made, and are confident that this will spur further developments.

Bibliography

Broman, K. W., Sen, Ś., Owens, S. E., Manichaikul, A., Southard-Smith, E. M. and Churchill, G. A. (2006), The X chromosome in quantitative trait locus mapping, *Genetics* **174**(4), 2151–2158.

Brown, T. (2002), The Human Genome - Genomes - NCBI Bookshelf.
URL: *https://www.ncbi.nlm.nih.gov/books/NBK21134/*

Chen, P., He, G., Zou, X., Wang, M., Jia, F., Bai, H., Li, J., Yu, J. and Han, Y. (2018), Forensic characterization and genetic polymorphisms of 19 X-chromosomal STRs in 1344 Han Chinese individuals and comprehensive population relationship analyses among 20 Chinese groups, *PLoS One* **13**(9).

Emigh, T. H. (1980), A comparison of tests for Hardy-Weinberg equilibrium, *Biometrics* **36**(4), 627–642.
URL: *http://www.jstor.org/stable/2556115*

Engels, W. R. (2009), Exact tests for Hardy-Weinberg proportions, *Genetics* **183**(4), 1431–1441.

Foulkes, A. S. (2009), *Applied statistical genetics with R*, Springer.

François, R. and Eddelbuettel, D. (2011), Rcpp: Seamless R and C++ integration, *Journal of Statistical Software* **40**.
URL: *https://cran.r-project.org/web/packages/Rcpp*

Graffelman, J. (2015), Exploring diallelic genetic markers: The HardyWeinberg package, *Journal of Statistical Software* **64**(3), 1–23.

Graffelman, J. and Camarena, J. (2008), Graphical tests for Hardy-Weinberg equilibrium based on the ternary plot, *Human heredity* **65**, 77–84.
URL: *https://cran.r-project.org/web/packages/HardyWeinberg/index.html*

Graffelman, J. and Weir, B. S. (2016), Testing for Hardy-Weinberg equilibrium at biallelic genetic markers on the X chromosome, *Heredity* **116**(6), 558–568.

Graffelman, J. and Weir, B. S. (2018), Multi-allelic exact tests for Hardy–Weinberg equilibrium that account for gender, *Molecular Ecology Resources* **18**(3), 461–473.

Guo, S. W. and Thompson, E. A. (1992), Performing the exact test of Hardy-Weinberg proportion for multiple alleles, *Biometrics* **48**(2), 361.

Haldane, J. S. (1954), An exact test for randomness of mating, *Journal of Genetics* **52**(1), 631–635.

Hedrick, P. (2011), *Genetics of populations*, Jones & Bartlett Learning.

Laird, N. M. and Lange, C. (2010), *The fundamentals of modern statistical genetics*, Springer.

Levene, H. et al. (1949), On a matching problem arising in genetics, *The annals of mathematical statistics* **20**(1), 91–94.

Li, R., Wang, M., Jin, L. and He, Y. (2013), A Monte Carlo permutation test for random mating using genome sequences, *PLoS One* **8**(8).

Louis, E. J. and Dempster, E. R. (1987), An exact test for Hardy-Weinberg and multiple alleles, *International Biometric Society Stable* **43**(4), 805–811.
URL: *http://www.jstor.org/stable/2531534*

Mehta, C. R. and Patel, N. R. (1983), A network algorithm for performing Fisher's exact test in $r \times c$ contingency tables, *Journal of the American Statistical Association* **78**(382), 427–434.

Mölkänen, T., Rostila, A., Ruotsalainen, E., Alanne, M., Perola, M. and Järvinen, A. (2010), Genetic polymorphism of the C-reactive protein (CRP) gene and a deep infection focus determine maximal serum CRP level in staphylococcus aureus bacteremia, *European journal of clinical microbiology & infectious diseases* **29**(9), 1131–1137.

Panawala, L. (2017), Difference between DNA and genes.
URL: *https://pediaa.com/difference-between-dna-and-chromosome*

Rachna, C. (2019), Difference between autosomes and sex chromosomes.
URL: *https://biodifferences.com/difference-between-autosomes-and-sex-chromosomes.html*

Takei, N., Miyashita, A., Tsukie, T., Arai, H., Asada, T., Imagawa, M., Shoji, M., Higuchi, S., Urakami, K., Kimura, H. et al. (2009), Genetic association study on in and around the APOE in late-onset Alzheimer disease in Japanese, *Genomics* **93**(5), 441–448.

The 1000 Genomes Project Consortium (2015), A global reference for human genetic variation, *Nature* **526**(7571), 68–74.

Wellek, S. and Ziegler, A. (2019), Testing for goodness rather than lack of fit of an X–chromosomal SNP to the Hardy-Weinberg model, *PLoS One* **14**(2).

Wigginton, J. E., Cutler, D. J. and Abecasis, G. R. (2005), A note on exact tests of Hardy-Weinberg equilibrium, *American Journal of Human Genetics* **76**(5), 887–893.

Acknowledgements

Writing thanks in a thesis is something that, to be honest, I've been dreaming about for a long time. I dreamt about it because maybe, during all these years, I've never really been able to make people who were close to me understand that they were much more important than they imagined.
I start by thanking Professor Jan Graffelman for giving me this opportunity, starting me into a world that was completely unknown to me but making me become more and more passionate about every single subject.
I also thank Professor Mauro Gasparini who, despite the distance, managed to supervise the work done by Turin, giving me the opportunity to easily create a bridge between the two universities.
The biggest thanks goes to my family to whom I owe practically everything I am today. Mom and Dad, thank you for having the courage to let me go, to let me live these experiences, to improve myself. I will never forget your faces when, that cloudy afternoon in Turin, you left me on that sidewalk alone on your way to the airport. I know how new and difficult all this was for you, but I assure you it was no less for me. I think that, in the end, it was worth it to live days like these and make you proud of me, I love you.
Antonio, I also apologize to you for leaving you alone in that room shared by a lifetime enjoying the best years without almost ever having me near you. In spite of this we are even closer than before, even if now you are turning into a man of law with whom it is increasingly difficult to get along.
I thank my grandmothers, for keeping me company every day between calls and packages from downstairs, who have bridged the gap by always making me feel at home.
I thank my uncles and cousins, who have been close to me from the beginning, motivating me when I thought I wouldn't make it, like the very first days, and who didn't think for a second to come all together to celebrate in Turin with me.
Viviana, I think that every word could be superfluous or taken for granted. Only I know how fundamental you have been in my path. You came into my life in an anonymous moment, making every single day more cheerful, lively and giving me strength when I needed it. Sorry for the distance, sorry for my character, sorry for

the too many hours spent studying, sorry even for those spent on the bus to see us. Thank you for having been there from the first moment, making a walk this path that we are taking in parallel, despite the distance, with joys and sufferings that divided by two weigh much less.
I thank all my friends in Sicily, Marco, Peppe, Cesare, who supported me from a distance and always welcomed me with open arms when I got home.
I thank the friends and colleagues I met in Turin, with whom I probably shared more during these years. The outings, the dinners, the "arrustute", the parties. Thank you for giving me a second family in that city to which we all owe so much and in which we keep memories that I will never forget.
Thanks also to all the people I met in Barcelona, with whom I immediately had a strong bond despite the few months together, thanks for making our Erasmus one of the best experiences of my life.
I hope I won't forget anyone, if I am what I am today it is also thanks to each and every one of you.
I thank myself for having been able to push myself beyond what I would have never thought of doing alone five years ago, for having had the courage to dare with a great weight on my shoulders, for having succeeded and for having made you proud of me.

Ringraziamenti

Scrivere i ringraziamenti in una tesi è un qualcosa che, ad essere sincero, sognavo da molto tempo. Lo sognavo perché forse, durante tutti questi anni, non sono mai riuscito a far capire veramente alle persone che mi sono state vicine che sono state molto più importanti di quello che si immaginano.
Inizio con il ringraziare il professor Jan Graffelman per avermi dato questa opportunità, iniziandomi ad un mondo che mi era completamente sconosciuto ma facendomi appassionare ad ogni singolo argomento sempre di più. Ringrazio anche il professor Mauro Gasparini che, nonostante la distanza, è riuscito a supervisionare il lavoro svolto da Torino, dandomi la possibilità di creare facilmente un ponte tra le due università.
Il ringraziamento più grande va alla mia famiglia cui devo praticamente tutto quello che sono oggi. Mamma e papà, grazie per aver avuto il coraggio di lasciarmi andare, di farmi vivere queste esperienza, di migliorarmi. Non dimenticherò mai le vostre facce quando, quel nuvoloso pomeriggio a Torino mi lasciaste su quel marciapiede da solo dirigendovi all'aeroporto. So quanto tutto questo sia stato nuovo e difficile per voi, ma vi assicuro che per me non è stato da meno. Penso che, alla fine, ne sia valsa la pena per vivere giornate come queste e rendervi orgogliosi di me, vi voglio bene.
Antonio, chiedo scusa anche a te per averti lasciato da solo in quella stanza condivisa da una vita a goderti i migliori anni senza avermi quasi mai vicino. Nonostante questo siamo ancora più legati di prima, anche se adesso ti stai trasformando in uomo di legge con cui è sempre più difficile essere d'accordo.
Ringrazio le mie nonne, per avermi fatto compagnia quotidianamente tra chiamate e pacchi da giù, che hanno colmato la distanza facendomi sempre sentire a casa.
Ringrazio i miei zii ed i miei cugini, che mi sono stati vicini fin dall'inizio, motivandomi quando credevo che non ce l'avrei fatta, come i primissimi giorni, e che non ci hanno pensato neanche un secondo a venire tutti insieme a festeggiare a Torino insieme a me.
Viviana, penso che ogni parola potrebbe risultare superflua o scontata. Solo io so quanto tu sia stata fondamentale nel mio percorso. Sei arrivata nella mia vita in un momento anonimo, rendendo ogni singolo giorno più allegro, vivace e dandomi

forza quando ne avevo bisogno. Scusa per la distanza, scusa per il mio carattere, scusa per le troppe ore dedicate allo studio, scusa anche per quelle trascorse in autobus per vederci. Grazie per esserci stata fin dal primo momento, rendendo una passeggiata questo percorso che stiamo portando avanti in parallelo, nonostante la distanza, con gioie e sofferenze che divise per due pesano molto meno.
Ringrazio tutti i miei amici di giù, Marco, Peppe, Cesare, che mi hanno supportato a distanza e che mi hanno sempre accolto a braccia aperte quando tornavo a casa.
Ringrazio gli amici e i colleghi conosciuti a Torino, con cui probabilmente ho condiviso di più durante questi anni. Le uscite, le cene, le "arrustute", le partitelle. Grazie per avermi dato una seconda famiglia in quella città a cui tutti noi dobbiamo tanto e in cui custodiamo ricordi che non dimenticherò mai.
Grazie anche a tutte le persone che ho conosciuto a Barcellona, con cui ho avuto subito un forte legame nonostante i pochi mesi insieme, grazie per aver reso in nostro Erasmus una delle migliori esperienze della mia vita.
Spero di non dimenticare nessuno, se oggi sono quel che sono è anche merito di ognuno di voi.
Ringrazio me stesso per essere stato in grado spingermi oltre quello che mai avrei pensato di fare da solo cinque anni fa, per aver avuto il coraggio di osare con un grande peso sulle spalle, di avercela fatta e di avervi resi orgogliosi di me.