University of Liège

Faculty of Applied Sciences
Department of Electrical Engineering and Computer Science
Montefiore Institute

An Efficient and Flexible Software Tool for Genome-Wide Association Interaction Studies

Academic year
2015-2016

PhD thesis by
Van Lishout François

Promotor
Prof. Dr. Dr. Van Steen Kristel

Jury members

Boigelot Bernard (president)	Professor, University of Liège (Montefiore Institute)
Van Steen Kristel (promotor)	Professor, University of Liège (GIGA-R)
Wehenkel Louis	Professor, University of Liège (Montefiore Institute)
Farnir Frédéric	Professor, University of Liège (Faculty of Veterinary Medicine)
König Inke	Professor, University of Lübeck (Med. Biometrie & Stat. Inst.)
Van der Spek Peter	Professor, Erasmus MC Rotterdam (Bioinformatics Department)

Liège, 2016

Thesis submitted in fulfilment of the requirements for the degree of Doctor in Electrical Engineering and Computer Science

Acknowledgements

First of all, I would like to express my deepest gratitude to my promotor, professor Kristel Van Steen. She gave me the chance to work in a good multi-cultural environment on a very interesting subject, both from a theoretical and a practical point of view. With her, I learned many important aspects of conducting a research project successfully to its end. In the earlier days of this thesis, few people believed that the MB-MDR methodology could one day allow genome-wide association interaction studies. However, Kristel Van Steen did and gave me the opportunity to help her proving it. Despite the fact that this task seemed almost impossible to me, I trusted her and worked over the years in speeding up the methodology, to finally prove her right.

Second, I would like to thank Louis Wehenkel. He is truly the one who discovered my research skills, at a time when I was still master student and did not see them myself. After having worked as a software developer in industry for some years, my way came across his again and he offered me a position at the GIGA. Then, he introduced me to Kristel Van Steen and she made this PhD project possible. I would also like to thank Louis Wehenkel for the long and instructive discussions that we had over the years, about and around my PhD thesis. Finally, let me mention that working as a teaching assistant for him was always a real pleasure to me.

I am very grateful to the staff of the Electrical Engineering and Computer Science Department of the University of Liège, for all the help given. A special thanks to my colleagues Jestinah Mahachie John, Elena Gusareva, Kirill Bessonov, Francesco Gadaleta and Tom Cattaert who contributed heavily to my work and published with me. In addition, I would also like to mention Bärbel Maus, Ramouna Fouladi, Benjamin Dizier, Raphael Liégeois, Alejandro Marcos Alvarez, Cyril Soldani, Pierre Geurts, Damien Ernst, Guillaume Drion, Samuel Hiard, Julien Becker and many more with whom I had very interesting research talks that helped me to conceptualize my ideas.

My gratitude goes to my jury members, for taking the time to read and evaluate my thesis. I would also like to thank all the international collaborators that I interacted with in the course of my PhD: Jason Moore, Damian Gola, Malu Calle, Viktor Urea, Duan Quingling, Kelan Tantishira, Isabelle Cleynen, Céline Vens, Lars Wienbrandt and many more.

My most affectionate thanks go to my family. My mother Myriam Kremers, who was always there to listen to me in the moments of doubts. In the memory of my father Pierre Van Lishout, who taught me how to give my best in everything I do. My girl Célia Van Lishout, who is truly my sunshine. I went to some difficult moments in the last two years and she was really the one who gave me the energy to go through them.

François Van Lishout
Liège, April 2016

Abstract

Humans are made up of approximately 3.2 billion base pairs, out of which about 62 million can vary from one individual to another. These particular base pairs are called single nucleotide polymorphisms (SNPs). It is well known that some particular combination of SNP values increase dramatically the risk of contracting certain type of disease, like Crohn's disease, Alzheimer, diabetes and cancer, just to name a few. However, there are still a lot of new discoveries to make and specialized software is required for this task.

It has been shown that individual SNPs cannot account for much of the heritability on their own. Therefore, this PhD thesis is dedicated to interaction studies, the purpose of which is to identify pairs of SNPs and/or environmental factors that might regulate the susceptibility to the disease under investigation. Model-Based Multifactor Dimensionality Reduction (MB-MDR) is a powerful and flexible methodology to perform interaction analysis, while minimizing the amount of false discoveries.

Before this thesis, the only available implementation was an R-package taking days to analyze a dataset composed of just hundred of SNPs. However, a typical dataset contains hundreds of thousands or millions of SNPs, even after data cleaning and quality control. The aim of this thesis is to write a software able to analyze such datasets within a few days with the MB-MDR methodology. In other words, the goal is to get 10^8 times faster than the R-package, while still remaining powerful, flexible and keeping the amount of false discoveries low.

Several contributions were needed to reach this goal and are presented in this thesis. First, a new software was written from scratch in C++, in order to be able to optimize every single computation, instead of relying on too generic functions as was the case for the R-package. Second, the methodology itself was improved, irrespective of the programming language. Indeed, MB-MDR is based on the maxT algorithm (introduced by Westfall&Young in 1993) to assess significance of the results and it can be customized for interaction analysis. A first major contribution of this PhD work, called Van Lishout's implementation of maxT, was introduced in 2011. The parallel version of this algorithm enables to analyze a dataset composed of hundred thousands of SNPs within a few days. The most important contribution of this thesis, called the gammaMAXT algorithm, was introduced in 2014. The parallel version enables to analyze a dataset composed of one million SNPs within one day.

In this thesis, we also propose a new viewpoint to handle population stratification and correct for covariates. Many simulated and real-life data analysis are provided, to highlight the flexibility of the software and its ability to find interesting results from a biological point of view. The latest version, called `mbmdr-4.4.1.out`, can be downloaded freely at `http://www.statgen.ulg.ac.be` with the corresponding documentation.

Résumé

L'homme est composé d'approximativement 3,2 milliards de paires de bases, dont plus ou moins 62 millions peuvent varier d'un individu à l'autre. Ces paires de bases variant d'un individu à l'autre sont appelées des SNPs (polymorphisme d'un seul nucléotide). Il est un fait avéré que certaines combinaisons de valeurs de SNPs augmentent fortement les chances de contracter certaines maladies comme la maladie de Crohn, Alzheimer, le diabète et le cancer pour n'en citer que quelques unes. Par contre, il reste encore beaucoup de découvertes à faire et le développement de software spécialisé est nécessaire pour y parvenir.

Il a été montré qu'un SNP ne peut à lui seul apporter beaucoup d'information sur l'héritabilité d'une maladie. Par conséquent, ce doctorat est consacré aux études d'interactions, dont l'objet est d'identifier des paires de SNPs et/ou de facteurs environnementaux pouvant réguler la susceptibilité à contracter la maladie étudiée. La méthodologie MB-MDR (Model-Based Multifactor Dimensionality Reduction) est une technique puissante et flexible permettant d'effectuer des études d'interactions, tout en minimisant le nombre de fausses découvertes.

Avant cette thèse de doctorat, la seule implémentation disponible de cette méthodologie était un package écrit en R, prenant des jours pour analyser un des données contenant à peine quelques centaines de SNPs. Or, un jeu de données typique est à l'heure actuelle composé de centaines de milliers voir de millions de SNPs et ce même après avoir épuré les données et réalisé différents contrôles de qualité. L'objectif de ce doctorat est d'écrire un logiciel capable d'analyser ce genre de données en quelques jours à l'aide de la méthodologie MB-MDR. En d'autres mots, il s'agit de devenir 10^8 fois plus rapide que le package écrit en R, tout en restant au moins aussi puissant et rapide qu'avant et en maintenant un nombre minimum de fausses découvertes.

Plusieurs contributions furent nécessaires pour y arriver. Premièrement, un nouveau logiciel a été développé en C++ en partant de zéro et ceci afin d'optimiser toutes les opérations effectuées plutôt que de se baser sur des fonctions trop génériques comme c'était le cas dans le package R. Deuxièmement, la méthodologie elle-même a dû être améliorée et ce indépendamment du choix du langage de programmation. En effet, MB-MDR est basé sur l'algorithme maxT (Westfall & Young 1993) pour estimer la pertinence des résultats et cet algorithme a du être customisé pour le cas particulier des études d'interactions. Ainsi, la première contribution majeure de cette thèse, l'implémentation Van Lishout de maxT, a été proposée en 2011. La version parallèle permet d'analyser un jeu de données composé de cent mille SNPs en quelques jours. La contribution principale de ce doctorat est l'algorithme gammaMAXT, introduit en 2014. La version parallèle permet d'analyser un dataset composé d'un million de SNPs en un seul jour.

Cette thèse présente également un nouveau point de vue pour tenir compte de la stratification de la population et corriger les covariables. De nombreuses analyses de données (simulées ou réelles) sont également décrites dans ce manuscrit, pour mettre en évidence la flexibilité du software et sa capacité à découvrir des résultats intéressants d'un point de vue biologique. La version la plus récente du logiciel, mbmdr-4.4.1.out, peut être téléchargée gratuitement sur le site `http://www.statgen.ulg.ac.be`, avec la documentation correspondante.

Contents

List of Figures

List of Tables

List of Algorithms

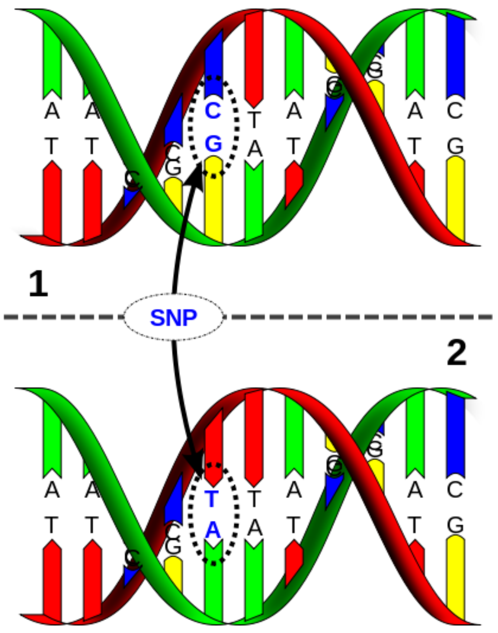
Chapter 1

General introduction

Public health genomics proposes to use genomic information to improve public health. The aim is to perform an effective and responsible translation of genome-based knowledge into public health policy and health services. One aspect of it is to assess the impact of genes and their interaction with environmental factors on human health. This research field is very wide and includes association studies, health prediction and descriptive genomics. Two major aims of the former are reaching a better understanding of the underlying mechanism behind diseases and identifying risk factors. The long-term ambition of predictive methods is to reach a personalized medicine, for which healthcare would be customized, i.e. based on molecular analysis and for instance clinical/familial history and lifestyle [106, 128, 37, 6, 71]. The main objective of descriptive genomics is molecular reclassification of subjects, using pattern recognition in a supervised or unsupervised way. This PhD thesis focuses on association studies.

Deoxyribonucleic acid (DNA) is the hereditary material of all known living organisms and many viruses. Most DNA molecules consist of two strands coiled around each other to form the famous double helix. The two DNA strands are composed of simpler units called nucleotides. Each nucleotide is composed of a chemical base - either adenine (A), thymine (T), cytosine (C), or guanine (G) - as well as a monosaccharide sugar called deoxyribose and a phosphate group. A critical feature of DNA is the ability of the nucleotides to make specific pairs: adenine pairs with thymine and cytosine with guanine. A gene is a region (locus) of DNA that encodes a functional ribonucleic acid (RNA) or protein product. RNA is a polymeric molecule assembled as a chain of nucleotides, which is more often found in nature as a single-strand folded onto itself, rather than a paired double-strand as for DNA molecules. Mutations can occur within genes, leading to different variants of the gene in the population, known as alleles.

The Human Genome Project (1990-2003), allowed the release of the first human reference genome by determining the sequence of about 3.2 billion base pairs and identifying the approximately 22 thousand human genes [55, 69, 72]. It comes without saying that nucleotides not differing from one individual to another cannot regulate susceptibility to disease. The focus in this thesis is therefore on the approximately 62 million remaining ones [89], called single nucleotide polymorphisms (SNPs) and illustrated in Figure 1.1. Note that the methods presented in this work are generic. As a consequence, other forms of polymorphisms can also be handled (for instance human hemoglobin and blood groups) and other species than humans can be studied.



Source: http://www.dnabaser.com/articles/SNP/SNP-single-nucleotide-polymorphism.html

Figure 1.1: A Single Nucleotide Polymorphism (SNP) is a DNA sequence variation in which a nucleotide in the genome differs between members of species. In this example, fragment 1 and 2 differ at a single base-pair location.

Genome-wide association studies (GWAs), using a dense map of SNPs, have become one of the standard approaches for unraveling the basis of complex genetic diseases [47]. However, significant genetic variants discovered in GWAS explain only a small proportion of the expected narrow-sense heritability, defined as the ratio of additive genetic variance to phenotypic variance [84, 29]. This is known as the *missing heritability problem*, i.e. the fact that individual genes cannot account for much of the heritability of phenotypes [109, 70]. Focusing on the combined effect of all the genes in the background, rather than on the disease genes in the foreground, is a promising idea for estimating heritability [145]. This PhD thesis is dedicated to genome-wide association interaction studies (GWAIs), the purpose of which is to identify pairs of SNPs that might regulate the susceptibility to the disease under investigation. Furthermore, we are particularly interested in detecting *epistasis*, occurring when the effect of one gene depends on the presence of one or more modifier genes. Epistatic interactions occur whenever one mutation alters the local environment of another residue (either by directly contacting it, or by inducing changes in the protein structure) [48]. Experimental studies in model organisms have demonstrated evidence of biological interactions among genes [78]. Furthermore, finding pairwise interactions opens the door for constructing statistical epistasis networks, that allows a better understanding of the biochemical mechanisms of complex diseases, as was demonstrated in the context of bladder cancer [118]. As an extension, we are also interested in genome-wide environmental interaction studies (GWEIs), whose aim is to identify gene-environmental factors interactions (for environmental factors such as age, gender, etc) [3]. Nowadays, a lot of genetical and/or environmental interactions impacting susceptibility to diseases still needs to be identified and dedicated software is needed for this task. The aim of this PhD is to develop an efficient and flexible software tool for GWAIs and GWEIs.

Note that from a clinical point of view, a better understanding of the etiology of complex diseases would be also be useful in the context of diagnosis of rare diseases. Indeed, around the world there are about 350 million people living with such disorders and in many cases the genetic factors are still unknown, making a molecular analysis pointless. The time to diagnosis is therefore undoubtedly long. It is not uncommon for parents having a child affected by such disorders, to wait for more than 10 years before finally knowing the type of illness affecting their infant! According to a 2013 survey, it takes on average 7.6 years in the US and 5.6 years in the UK to receive a proper diagnosis [97] (631 patients and 256 caregivers surveyed, 466 rare diseases represented). Furthermore, in the mean time, a patient typically visits four primary care doctors, four specialists and receives two to three misdiagnoses. According to Yves Moreau [88, 93], this is in line with what his colleagues geneticists observe in Belgium.

In this PhD project, we present an efficient and flexible software able to perform GWAIs on real-life data. The difficulty of this task is to find a good balance between four main issues, that we summarize in the following objectives:

1) Achieve sufficient statistical power to detect relevant pairs.

2) Minimize the amount of false discoveries.

3) Reduce the computational burden implied by the high number of interactions to investigate.

4) Provide a versatile software package that allow researchers to work with binary, continuous and survival traits, on datasets that may contain missing values, to consider multi-allelic data and categorical environmental exposure variables, to correct for main effects, regress covariates, correct for confouding factors, etc.

Among the numerous software designed for pairwise or higher-order SNP-SNP interactions, we consider BOOST [131], BiForce [45], epiGPU [50], EpiBlaster [61], GLIDE [60], Multifactor Dimensionality Reduction (MDR) [101, 46] and Model-Based Multifactor Dimensionality Reduction (MB-MDR) [13, 15]. Table 1.1 indicates which features related to our objectives are available in which software (without claiming to be exhaustive). Most methods do not handle missingness, but advocate to impute the missing data. In our lab, we advocate against imputing as such (without additional pruning steps) as it is known that linkage disequilibrium (LD) between markers can induce so-called redundant epistasis [87]. Two alleles at different loci are in LD if they occur together on the same chromosome more often than would be predicted by chance.

Table 1.1: Available features in some common association interaction studies software

Software tool	Handled traits	Handle missingness	Handle covariates	Maximum factor levels
BOOST	Binary	No	No	3
BiForce	Binary, Continuous	Yes	No	3
epiGPU	Continuous	No	No	3
EpiBlaster	Binary	No	No	3
GLIDE	Binary, Continuous	No	No	3
MDR	Binary, Continuous	No	No	3
MB-MDR	Binary, Continuous, Survival	Yes	Yes	Unlimited

The following comparison of these approaches is mainly inspired from [43] which reviews and discusses several practical aspects GWAIs typically involve. BOOST is a software based on fast Boolean operations, to quickly search for epistasis associated with a binary outcome. Its main drawbacks are its limitation to binary traits, its inability to accommodate missing data and its necessity to perform a multiple testing correction outside the software package. However, note that there is a module in PLINK to deal with BOOST in a more flexible, but still not perfect way. BiForce is a regression-based tool handling binary and continuous outcomes, that can take account of missing genotypes and has a built-in multiple testing correction algorithm. However, the latter is based on a fast Bonferroni correction implementation, which leads to reduced power for GWAIs (this subject will be deeply discussed in Chapter 3).

EpiBlaster, epiGPU and GLIDE are all GPU-based approaches. An obvious drawback of GPU-dependent software is that it is tuned for a particular GPU-infrastructure. Therefore, users are advocated to acquire the exact same infrastructure and only experts can adapt the code to specific needs. Note that users willing to work on dedicated hardware to speed up the computations can even turn to field-programmable gate array (FPGa) [137].

MDR is a non-parametric alternative to traditional regression-based methods that converts two or more variables into a single lower-dimensional attribute. The end goal is to identify a representation that facilitates the detection of non-linear or non-additive interactions. Over-fitting issues in MDR are solved via cross-validation and permutations. Since the design of MDR, several adaptations have been made [127].

MB-MDR breaks with the tradition of cross-validation and invests computing time in permutation-based multiple multilocus significance assessments and the implementation of the most appropriate association test for the data at hand. It is able to correct for important main effects. Its main asset compared to the other methods is its versatility. MB-MDR can for instance be used to highlight gene-gene (GxG), gene-environment (GxE) or gene-gene-environment (GxGxE) interactions in relation to a trait of interest, while efficiently controlling type I error rates and false positives. The trait can either be expressed on a binary or continuous scale, or as a censored trait. Before the start of this thesis in 2011, the only available software implementing the MB-MDR methodology was a very slow R-package [12].

In this PhD, we propose a C++ implementation of the MB-MDR methodology, fulfilling the aforementioned four objectives. The latest version, mbmdr-4.4.1.out, can be downloaded at `http://www.statgen.ulg.ac.be`, with the associated documentation. Older versions are also still available. mbmdr-4.4.1.out is an easy to use command-line software, including a man-like help. To see its home page, use the following command:

```
./mbmdr-4.4.1.out help
```

In Chapter 2, we introduce our epistasis detection methods. In other words, the aim of this chapter is to answer the research question "*How strongly is a particular GxG, GxE or GxGxE interaction related to the disease under investigation?*". This chapter successively describes the cases of a trait expressed on a binary scale, on a continuous scale and as a censored trait. Each time, methods are given for computing a number, called *MB-MDR statistic*, representing the degree of association between the interaction and the trait. These methods can be split into two categories: with and without correction for lower order effects.

Chapter 3 addresses the multiple testing problem. In other words, it answers the research question "*Correcting for the fact that a huge set of G*x*G, G*x*E or G*x*G*x*E interactions is investigated, what is the probability that a particular interaction is not significantly associated to the disease?*". This chapter is the corner stone of this PhD thesis. In particular, it contains a presentation of three classical multiple-testing correction algorithms, Bonferroni, maxT and minP [39], but proposes new ways to implement them in order to reach better performances. The purpose of these three algorithms is to control the family-wise error rate (FWER), Bonferroni being the most conservative correction method. MaxT, the default algorithm used in the MB-MDR framework, has requirements in terms of computing time and memory proportional to the number of permutations. This makes it unsuitable for GWAIs. The first main contribution of this PhD thesis solves the memory issue: Van Lishout's implementation of maxT makes the memory usage independent from the size of the dataset [125]. The second and most important contribution, gammaMAXT, solves the computing time issue: it speeds up the computing time further, while still controlling the FWER strongly and displaying a power similar to the original maxT algorithm.

Chapter 4 addresses the issue of population stratification and also discusses covariate adjustments. Two new algorithms, STRAT1 and STRAT2, are introduced to automatically correct for unmeasured confounding factors. Furthermore, better performances can be observed by correcting for covariates. In this context, two strategies are presented to correct for covariates. Residuals-based correction is a simple method, whose computing time is almost the same as if no correction for covariates would be performed, that increases the power to detect significant interactions when a clear relationship between the trait and the covariates that are regressed out exists. On-the-fly correction is a more computational intensive method, that allows to correct for any combination of categorical covariates and main effects of the SNPs, i.e. find out "pure" biological interactions that are not driven by any available variable.

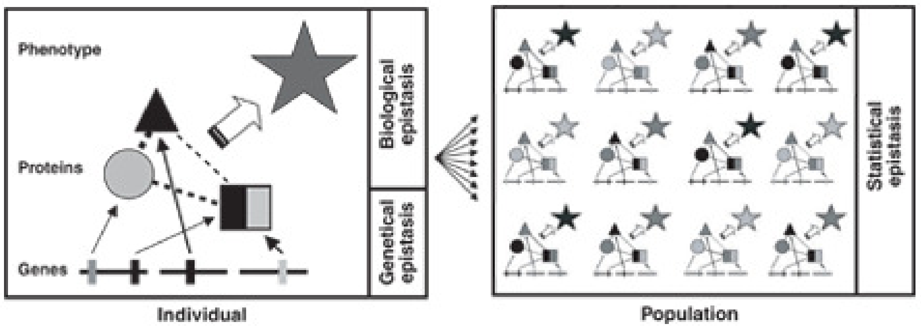
Chapter 5 presents different applications of the software, both on simulated and real-life datasets. This chapter demonstrates that the software can be used successfully to either retrieve known interactions from the literature or find new ones. A lot could be learned from the data analysis performed in this chapter, leading to recommendations for the users of our software, as well as new insights into the assets and limitation of the methods developed in this PhD work.

Chapter 2

Epistasis detection

2.1 Outline

Epistasis is important to tackle complex human disease genetics [78]. It can be viewed from two perspectives, biological and statistical, illustrated in Figure 2.1. Biological epistasis is "*the result of physical interactions among bio-molecules within gene regulatory networks and biological pathways in an individual such that the effect of a gene on a phenotype depend on one or more other genes*" [87]. Statistical epistasis is defined as "*deviation from additivity in a mathematical model summarizing the relationship between multilocus genotypes and phenotypic variations in a population*" [87].

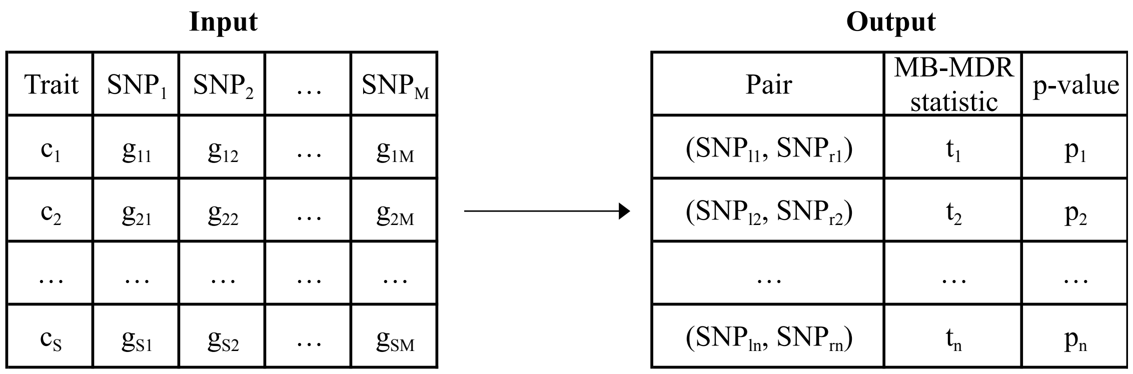


Source: http://www.nature.com/ng/journal/v37/n1/fig_tab/ng0105-13_F1.html

Figure 2.1: Biological epistasis is defined at the individual level and involves DNA sequence variations (vertical bars), bio-molecules (square, circle and triangle) and their physical interactions (dashed lines), leading to a phenotype (star) [87]. Statistical epistasis is defined at the population level, as a phenomenon made possible by inter-individual variability in genotypes, bio-molecules and their interactions [87].

Writing a software that detects statistical epistasis is a challenging task. A typical genome-wide real-life dataset is composed of millions of SNPs, leading to a huge amount of interactions to investigate, even when restricting our attention to two-order interactions (in this case about 10^{12} SNPs). In this chapter, we specify our methodology to compute an *MB-MDR statistic*, capturing the *degree of association between a particular pair of SNPs and the trait.* This enables to produce a ranking of the most promising interactions. In Chapter 3, we show how to compute a p-value indicating if a pair is statistically significantly associated to the trait or not.

Figure 2.2 describes the input information that we have and the output that we aim to produce[1].



Input

Trait	SNP_1	SNP_2	…	SNP_M
c_1	g_{11}	g_{12}	…	g_{1M}
c_2	g_{21}	g_{22}	…	g_{2M}
…	…	…	…	…
c_S	g_{S1}	g_{S2}	…	g_{SM}

Output

Pair	MB-MDR statistic	p-value
(SNP_{l1}, SNP_{r1})	t_1	p_1
(SNP_{l2}, SNP_{r2})	t_2	p_2
…	…	…
(SNP_{ln}, SNP_{rn})	t_n	p_n

Figure 2.2: The input information consists of the trait and SNP values for the S subjects under study. The trait can either be expressed on a binary or continuous scale, or as a censored trait (in the latter case the trait column is replaced by two columns, respectively for the time and censoring variables). In the event of a binary scale, if the s^{th} subject is a case (control), $c_s = 1(0)$ $(s = 1 \dots S)$. In the case of a continuous scale, c_s is a continuous value representing the state of the s^{th} subject. SNP_b is a label referring to the b^{th} SNP $(b = 1, \dots M)$. The genotype of an individual s at locus b is denoted as g_{sb} (0 if homozygous for the first allele, 1 if heterozygous, 2 if homozygous for the second allele and -9 if missing). Multi-allelic variables and categorical environment variables are also covered, as long as they are coded with natural numbers. The task consists in generating a ranking of the most significant SNP pairs in relation with the trait. (SNP_{lj}, SNP_{rj}) refers to the j^{th} best SNP pair, i.e. the pair with the j^{th} highest MB-MDR statistic t_j.

In this chapter, we propose different strategies for computing an MB-MDR statistic. Obviously, the scale of the trait leads to different types of statistics. Furthermore, missing values in the dataset can be handled in different ways. Nonetheless, an important aspect is also whether to adjust the statistic for the main effect of the SNPs or not. It comes without saying that not adjusting leads to simpler statistics, faster to compute. However, as soon as the data contains at least one SNP with an important main effect, these statistics are not good enough. Indeed, the output would undoubtedly contain a lot of significant pairs, making it impossible to distinguish between real pure epistatic effects and results driven by the main effect. The default option in our software is thus to always adjust for main effects. We show in this chapter that there are several ways to perform the adjustment and discuss the benefits and drawbacks of each choice. In the next section, we focus on the scenario where the trait is expressed on a binary scale. We show how to compute a simple statistic (not adjusting for main effects) using the MB-MDR framework and how to compute two more elaborate statistics to regress the main effects. We also discuss the pros and cons of using other approaches from the literature. Traits expressed on a continuous and on a survival scale are handled in Sections 2.3 and 2.4.

[1] In this document, we assign index 1 to the first element of a sequence to facilitate reading. However, note that the actual implementation uses zero-based numbering for efficiency reasons

2.2 Trait expressed on a binary scale

Case-controls studies are widely used in medicine. Typically, a group of patients having a disease (the "cases") is compared to a group of patients not affected by the disease (the "controls"), as illustrated in Figure 2.3. This design does not factually restrict to diseased versus healthy patients. It can in general be used to split subjects in two groups depending on any condition (for instance, in the context of BMI, morbid obese or not). In this work, we are interested in finding the pairs of SNPs and/or environmental variables that bests predict the status of the subjects.

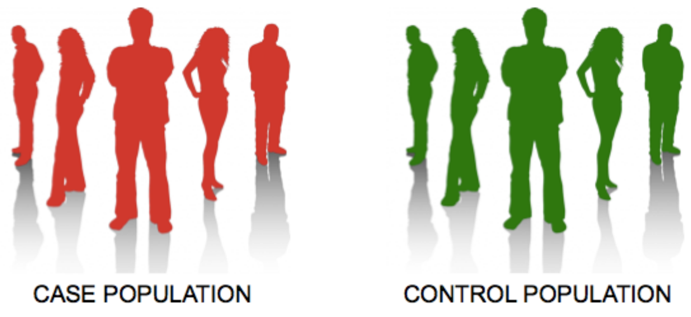


Figure 2.3: The aim of case-control studies is to determine if there is a statistically significant difference between the affected and unaffected subjects. In this thesis, we focus on genetic interactions that might regulate the susceptibility to the disease.

Our aim is to compute, for every pair $[SNP_{lj}, SNP_{rj}]$, a number t_j capturing its degree of association with the trait. This allows to sort the pairs by decreasing chances of regulating the disease. Let N_1 be the number of possible values for SNP_{lj} and N_2 the number of possible values for SNP_{rj}. In practice, most of the studies concern bi-allelic SNPs and $N_1 = N_2 = 3$. However, the software described in this thesis automatically detects the exact values of N_1 and N_2, so that multi-allelic markers (such as microsatellites) are also covered. Furthermore, environment variables can be also used as hypothesis, instead of SNPs, as long as they can be treated as categorical. They should be coded $0, 1, \ldots, N_1 - 1$, resp. $0, 1, \ldots, N_2 - 1$, in any convenient order.

2.2.1 Without correction for main effects

The MB-MDR framework proposes a slightly elaborate way to compute a statistic t_j representing the strength of the association between the pair $[SNP_{lj}, SNP_{rj}]$ and the trait $\mathcal{T}$. In the discussion section at the end of this chapter, we discuss the pros and cons of taking a more direct approach, based on information theory. Since the MB-MDR method comes from its ancestor MDR, we first present the latter. The idea of the MDR methodology, is to split the subjects into groups, depending on their genotypes. In the case of bi-allelic genetic marker, there are nine possible groups, as illustrated in Figure 2.4, inspired from [102]. Every non-empty group is either labeled "L" if it contains more controls than cases (low-risk) or "H" otherwise (high-risk). In this way, the dimensionality has been reduced to one dimension with two levels. However, note that although the dimension has been reduced, the effective degrees of freedom may not. It has been estimated to be 5.6 for MDR in case of interactions of order two [91].

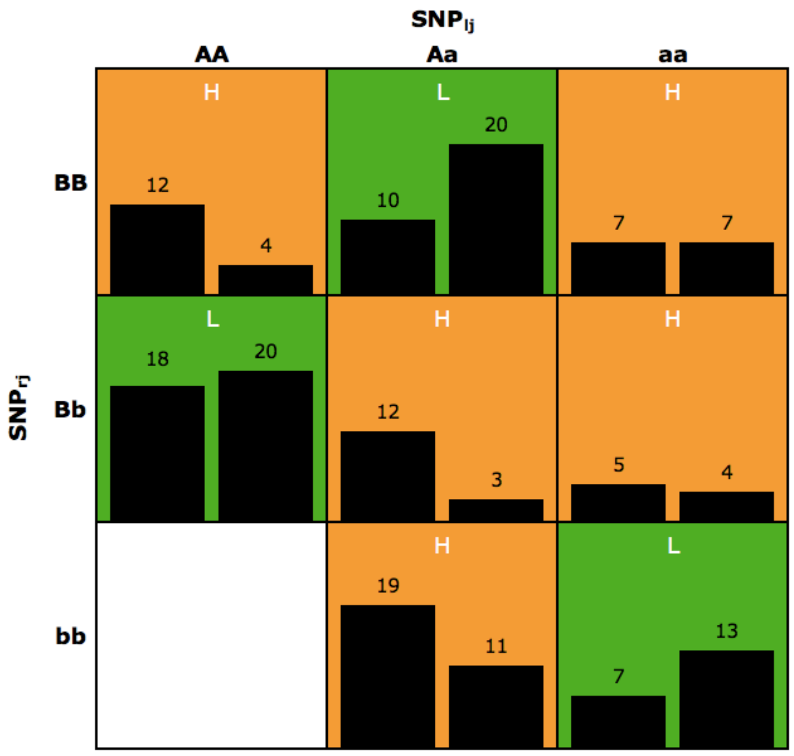


Figure 2.4: In the MDR methodology, the subjects are split into groups depending on their genotypes. These multifactor classes can be organized in a matrix representation. For every subject, the alleles of SNP_{lj} defines to which column it belongs to and the alleles of SNP_{rj} to which line. In every cell, the amount of cases is reported on the left and the amount of controls on the right. If the controls represent the majority, the group is considered to be in low risk for the disease (in this work such cells are colored in green). Otherwise, the group is considered to be in high risk for the disease (in this work such cells are colored in orange). Empty cells are ignored. Pooling all the orange cells together and all the green ones together, forms the genetic *model* of the effect of the pair of SNPs on the disease.

A classical 10-fold cross-validation procedure is used to compute the cross-validation consistency and the prediction error [101]. The entire procedure is performed 10 times, to diminish the odds of observing false results due to a poor division of the subjects. The cross-validation consistency has to be maximized and is defined as "*a measure of the number of times a particular set of loci and/or factors is identified across the cross-validation subset*" [101]. The prediction error has to be minimized and is defined as "*the average of prediction errors across each of the 10 cross-validation subsets*" [101]. When the optimal is different among these two metrics, the simplest model is chosen.

An important benefit of MDR is that it is a model-free approach. It does neither assume any particular genetic model, nor estimate any parameter. The dimensionality problem involved in interaction detection is tackled by pooling the subjects into two groups ("H" or "L"), hence the name of the method. Furthermore, this methodology does not just produce a list of pairs possibly linked to the trait, it also returns the corresponding genetic models of the disease. Empirical and theoretical studies suggest that MDR has excellent power for identifying epistasis [101].

An important drawback of this approach, it that the groups are forced to be either in high-risk or low-risk, even when there is no statistically significant difference between the amount of cases and controls [11]. Another hindrance is that the original method can only handle traits expressed on a binary scale, not a continuous one. The major drawback however, is probably that it cannot correct for the main effects of the SNPs, i.e. the effect that the SNPs have on their own on the trait. Therefore, we turn to the MB-MDR approach in this thesis, resolving all the aforementioned issues.

The idea of the MB-MDR methodology, is to split the subjects into groups in the same way as in the MDR methodology, except that there are now three possible categories instead of two: "O" if there is no statistical evidence for any risk change (neither high-risk, nor low-risk) and "L" or "H" otherwise, as illustrated in Figure 2.5.

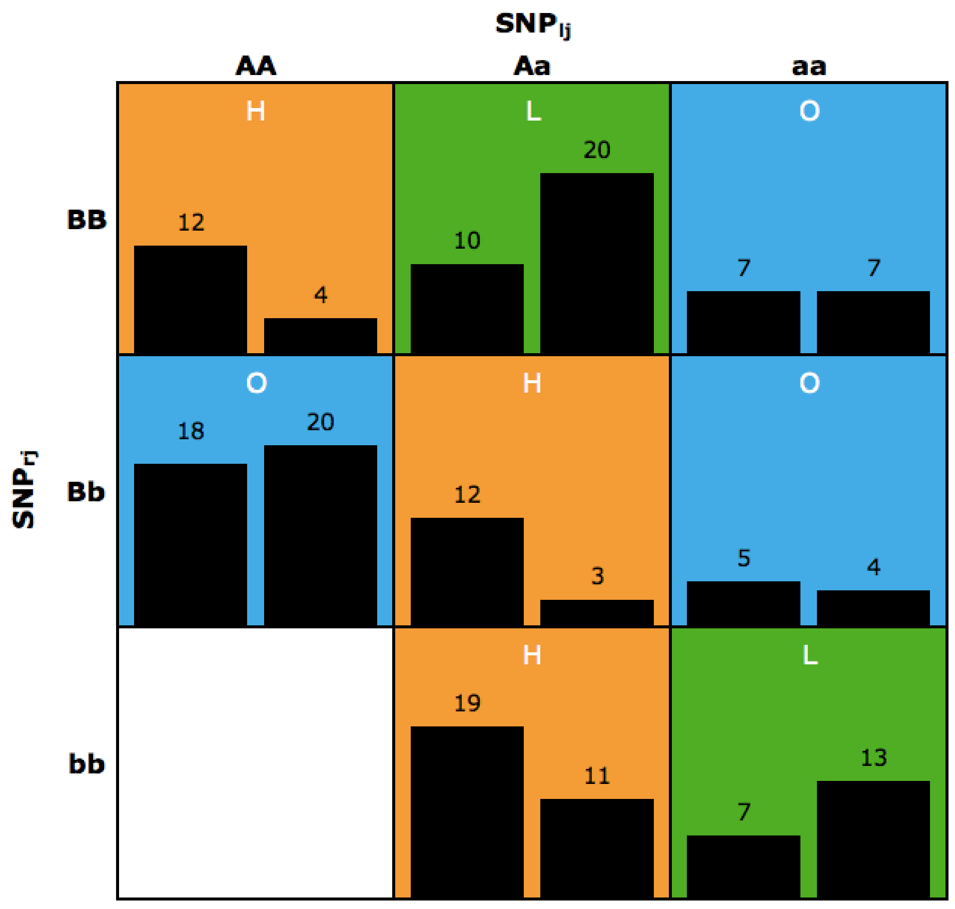


Figure 2.5: In the MB-MDR methodology, the subjects are still split into groups depending on their genotypes, in the same way as in Figure 2.4. However, there are now three possible categories for the cells. A group is immediately assigned to the "O" category (no-evidence for any risk change) if it consists of less than ten individuals. Otherwise, a statistical test is performed to decide if there is a statistically significant difference between the number of cases and controls. If there is, the group is assigned to either the "H" or "L" category, in the same way as before. Otherwise, it is assigned to the "O" category (in this work such cells are colored in blue).

Figure 2.6 explains the computation of a statistic t_j representing the strength of the association between the pair $[SNP_{lj}, SNP_{rj}]$ and the trait $\mathcal{T}$.

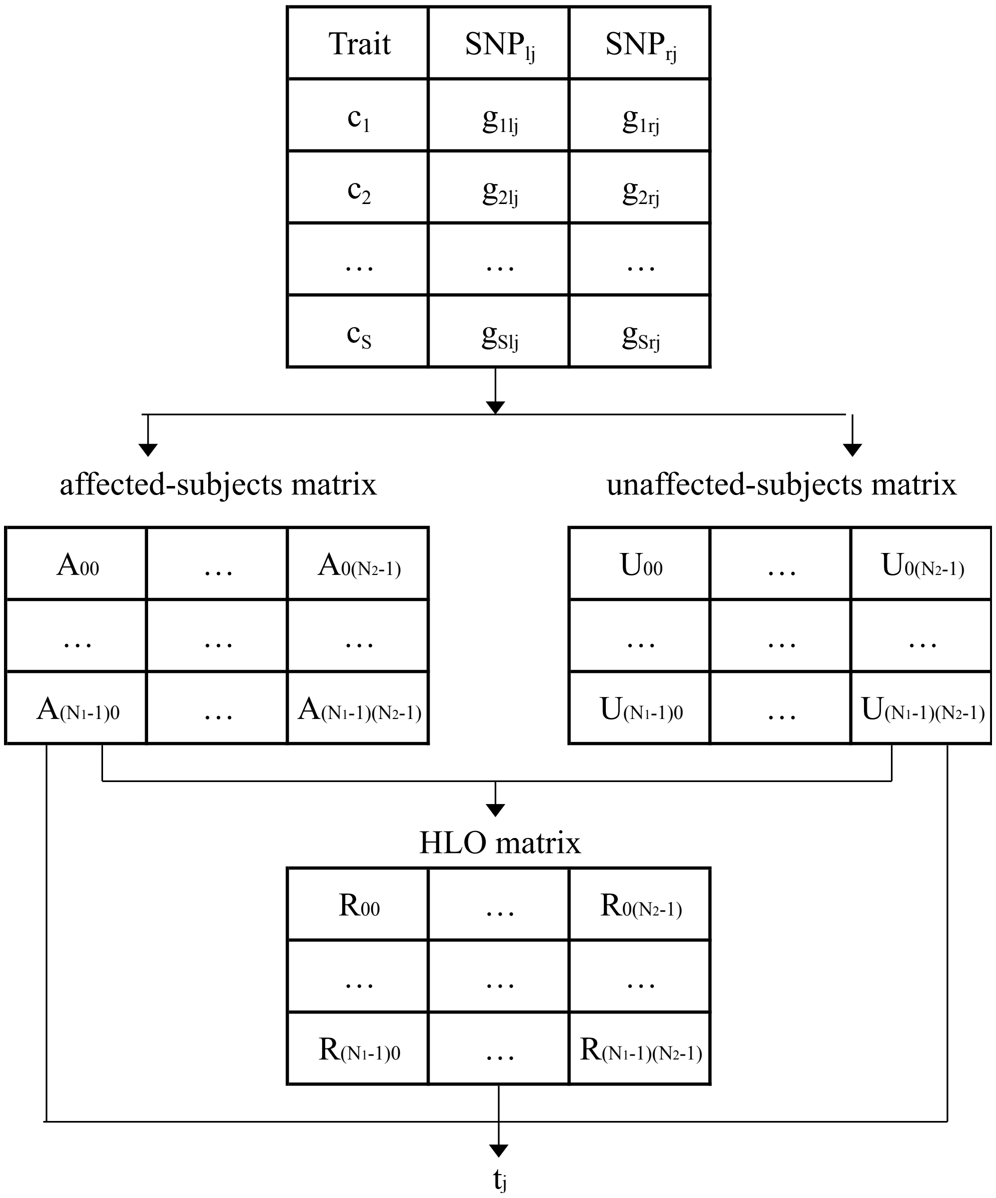


Figure 2.6: Computation of an MB-MDR statistic for a binary trait, without correction for main effects. Input: c_s is 1 (0) if the s^{th} subject is a case (control), g_{slj} and g_{srj} are the genotypes of the s^{th} subject for SNP_{lj} and SNP_{rj} respectively. The computation can be decomposed in three steps. First, the affected-subjects and unaffected-subjects matrices are constructed. A_{mn} and U_{mn} are respectively the number of affected/unaffected subjects, whose genotype $g_{slj} = m$ and $g_{srj} = n$. Second, the HLO matrix is constructed. R_{mn} is either "H" if the subjects whose genotype is m for SNP_{lj} and n for SNP_{rj} have a high statistical risk of disease, "L" if they have a low statistical risk and "O" if there is no statistical evidence (or no data). Third, the final t_j value is computed from the three matrices constructed at steps 1 and 2.

The three steps of the computation of the number t_j capturing the degree of association between the pair $[SNP_{lj}, SNP_{rj}]$ and the trait $\mathcal{T}$ are as follows:

1) Generation of the affected-subjects and unaffected-subjects matrices A and U. For efficiency reasons and to take account of empty groups more easily, these objects are in fact coded as vectors[2]. They are created with a size $N_1 \times N_2$ and initialized with zero values. Then, a loop over the subjects of the dataset is performed. For $s = 1, \dots S$: if no value is missing for subject s: increment a cell of the affected-subjects vector if $c_s = 1$, otherwise a cell of the unaffected-subjects one. The index of the cell to be incremented is given by $g_{slj} \times N_2 + g_{srj}$, i.e. depends on the subject's genotype[3]. The total amount N_A of affected and N_U of unaffected subjects without missing values can easily be deduced from these vectors. After this process, a loop is performed to create a vector T containing the total number of subjects in each non-empty group and a vector Y containing the ratio of cases versus number of subjects in each non-empty group[4]. Empty groups are removed from A and U on the fly during this process. Let dim be the final size of vectors A, U, T and Y.

2) Generation of the HLO-matrix from the objects generated at step 1. This object is in fact also coded as vector for efficiency reasons and has a size of dim. Each R_h element depends on a test for association between the trait and the belonging to the genotype group satisfying $(SNP_{lj} \times N_2 + SNP_{rj} = h)$, with $h = 0, \dots, dim-1$. A χ^2 test with one degree of freedom (1 df) is performed by default, to keep things simple. However, the architecture of the software makes it easy to implement other test statistics that are appropriate for the data at hand. The statistic that we use follows a χ^2 distribution and is defined as $\frac{(ad-bc)^2(a+b+c+d)}{(a+b)(c+d)(b+d)(a+c)}$, where a (c) refers to the number of affected (unaffected) subjects having a genotype satisfying $(SNP_{lj} \times N_2 + SNP_{rj} = h)$ and b (d) refers to the number of affected (unaffected) subjects having a different genotype. Those values are easy to compute: $a = A_h$, $b = N_A - A_h$, $c = U_h$ and $d = N_U - U_h$. At this point, if either $a + c$ or $b + d$ is below a threshold that is a parameter of the program (default value 10) then the test is not performed at all, since it would not be statistically significant. In this case the value of R_h is automatically set to "O", to indicate the absence of evidence that the subset of individuals with multilocus genotype satisfying $SNP_{lj} \times N_2 + SNP_{rj} = h$ has neither a high nor a low risk for disease. Otherwise, the test is performed. When the computed χ^2_{obs} value is below the critical value[5] $\chi^2_{\text{crit}} = 2,705543$, the value of R_h is set to "O", to indicate that we cannot reject the independence hypothesis. Otherwise, R_h is set to either "H" if $(ad - bc) > 0$, to indicate a high risk of having the trait, or to "L" if $(ad - bc) < 0$, to indicate a low risk for this event.

[2] From the programmers perspective, coding a matrix as a vector is just an implementation choice. Matrices are used in Figure 2.6 because from a conceptual point of view it is easier to understand.

[3] Note that this is the same as working in matrix format and numbering the cells from left to right and top to bottom, by starting with 0 on the top left corner and finishing on the bottom right one.

[4] The vectors T and Y are not necessary from a conceptual point of view (they can be deduced from A and U) and do therefore not appear in Figure 2.6. In practice, they are useful to win computing time by avoiding some operations to be performed several times in the next steps.

[5] 90th percentile of the χ^2 distribution with 1 degree of freedom.

3) Computation of t_j from the objects generated at steps 1 and 2. It consists in performing two χ^2 tests with 1 df and returning the maximum of both. The first one tests association between the trait and the belonging to the "H" category versus the "L" or "O" one. The second one tests association between the trait and the belonging to the "L" category versus the "H" or "O" one. In the first (second) case, a and c are respectively the number of affected and unaffected subjects belonging to the "H" ("L") category and b and d to the "L" ("H") or "O" category. Computing this can easily be achieved by initializing a, b, c and d to zero, and for $h = 0, \ldots, dim - 1$ adding A_h to a and U_h to c if $R_h =$ "H" ("L") and A_h to b and U_h to d otherwise. Note that in many case, there are only cells associated to the "O" category and t_j is readily equal to zero.

The detailed algorithm computing statistics t_j (without correction for the main effects of the SNPs) is reported in Box 2.1.

Box 2.1. Binary MB-MDR statistic computing (no 1st order correction)

```
(1) Create vectors $A, U, T$ and $Y$ of size $dim = N_1 \times N_2$. Define $\chi^2_{crit} = 2,705543$.
(2) Fill $A$ and $U$ with 0's. For $s = 1, \ldots, S$: if $g_{slj}$ and $g_{srj}$ are not missing do
    either $A_{g_{slj} \times N_2 + g_{srj}}{++}$ if $c_s = 1$ or $U_{g_{slj} \times N_2 + g_{srj}}{++}$ otherwise.
(3) Set $a = 0$. For $h = 0, \ldots, dim - 1$: compute $T_a = A_a + U_a$ ; if ($T_a = 0$) erase
    the $a^{\text{th}}$ element of vectors $A$ and $U$, else compute $Y_a = \frac{A_a}{T_a}$ and perform $a{++}$.
    After the loop compute the final dimension: $dim = a$.
(4) Compute $N_A = A_0 + \ldots + A_{dim-1}$, $N_U = U_0 + \ldots + U_{dim-1}$ and $N = N_A + N_U$.
(5) Create HLO vector $R$ of size $dim$. For $h = 0, \ldots, dim - 1$:
    (a) Define $a = A_h$, $b = N_A - a$, $c = U_h$ and $d = N_U - c$.
    (b) If $(a + c) < 10$ or $(b + d) < 10$: set $R_h =$ "O", else compute
        $\chi^2_{obs} = \frac{(ad-bc)^2 N}{N_A N_U (b+d)(a+c)}$. If $\chi^2_{obs} < \chi^2_{crit}$ set $R_h =$ "O", else if
        $(ad - bc) < 0$ set $R_h =$ "H", else set $R_h =$ "L".
(6) If there is no "H" and no "L" in the R vector, return $t_j = 0$.
(7) If there is at least one "H":
    (a) Initialize $a = 0$ and $c = 0$. For $h = 0, \ldots, dim - 1$: if $R_h =$ "H" compute
        $a {+}{=} A_h$ and $c {+}{=} U_h$.
    (b) Define $b = N_A - a$ and $d = N_U - c$. Compute $\chi^2_{\text{HvsLO}} = \frac{(ad-bc)^2 N}{N_A N_U (b+d)(a+c)}$.
(8) If there is at least one "L":
    (a) Initialize $a = 0$ and $c = 0$. For $h = 0, \ldots, dim - 1$: if $R_h =$ "L" compute
        $a {+}{=} A_h$ and $c {+}{=} U_h$.
    (b) Define $b = N_A - a$ and $d = N_U - c$. Compute $\chi^2_{\text{LvsHO}} = \frac{(ad-bc)^2 N}{N_A N_U (b+d)(a+c)}$.
(9) Return $t_j = \max(\chi^2_{\text{HvsLO}}, \chi^2_{\text{LvsHO}})$.
```

Since we are interested in GWAIs and GWEIs, we must realize that the part of the code that reads the data at the start of the program cannot store it in cache because of its size. Accessing to the trait and SNP values is thus slow and must be avoided as much as possible. For this reason, the three columns *Trait*, SNP_{lj} and SNP_{rj} of Figure 2.6 are passed by value and not by reference to the function. In this way, an explicit local copy of them is performed, on which the function is able to work faster. Note that the loops at step (3), (4), (5), (7) and (8) in Box 2.1. involves only up to nine iterations (in the bi-allelic case) and are therefore very fast. Everything has been optimized to reduce the number of operations to a minimum. This is indeed crucial, since this function is called about 10^{15} times for GWAIs, as shown in Chapter 3.

An important benefit of MB-MDR, is that the robust association tests that are performed, hereby acknowledging that not all multi-locus genotype combination are informative, have proven to give optimal performance compared to MDR, especially in the presence of genetic heterogeneity [15]. Another value of MB-MDR is that it is flexible: it can be adapted to correct for the main effects of the SNPs in different ways, it can handle missingness optimally, the trait can be expressed on different scales,... The main drawback of the computation of MB-MDR statistics compared to MDR ones, is that they require more computing time. However, by optimizing all steps as we did, the calculation time can be kept low (numerical results are given in Chapter 3).

2.2.2 With correction for main effects

In the previous section, we have shown strategies to perform a global epistasis test. In this section, we show how to perform a targeted epistasis test within the MB-MDR framework, by adjusting for lower-order effects. This is in fact the origin of the model-based (MB) component in the name of the MB-MDR methodology. Correcting for lower-order effects avoids main effects signals giving rise to false epistasis effects [80].

Coding scheme

In genome-wide association testing, the SNPs are often coded in an additive way, i.e. the code is given by the number of minor alleles [38]. For instance, if A is the major allele and C the minor one, AA is coded 0, AC is coded 1 and CC is coded 2. If C has a negative effect on the disease, the genotype CC is considered to be two times worse for the subject's health than the AC one. The additive scheme is very straightforward and can work well in a lot of scenarios, but power can be gained by taking the true underlying genetic models into account [108]. An alternative is to use a codominant coding. In the bi-allelic case, two binary variables are used: one equal to 1 for AC and 0 otherwise and the other equal to 1 for CC and 0 otherwise. In mbmdr-4.4.1.out, the user can choose between these two coding. Codominant is the default, since it is an all-purpose acceptable (in terms of power and type I error) choice [79]. In practice, choosing between additive and codominant coding schemes implies choosing between the least and most severe such removal of effects [79]. The general idea of the lower-order correction is as follows. First, construct the affected-subjects and unaffected-subjects matrices of Figure 2.6, but implemented as vectors. Then, create again a vector Y of size dim containing the ratio of cases versus number of subjects for every non-empty group. Then, try to fit the following model:

$$E\{Y|X\} = X\beta \tag{2.1}$$

where X is the model matrix and β a row vector of parameters to fit. For an additive correction, the model matrix X_{additive} consists of $dim_c = 3$ columns (one for the intercept, 1 for SNP_{lj} and 1 for SNP_{rj}) and dim rows [14]. The second column corresponds to the value of SNP_{lj} and the third one to the value of SNP_{rj} . Vector β is obviously composed of $dim_c = 3$ elements. In the bi-allelic case, when no group is empty, the model matrix is given explicitly by

$$X_{additive} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 2 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 0 \\ 1 & 2 & 1 \\ 1 & 2 & 2 \end{bmatrix} \tag{2.2}$$

For a codominant main effects correction, the model matrix $X_{\text{codominant}}$ consists of $dim_c = N_1 + N_2 - 1$ columns (one for the intercept, $N_1 - 1$ for SNP_{lj} and $N_2 - 1$ for SNP_{rj}) and dim rows [14]. Vector β is obviously composed of $dim_c = N_1 + N_2 - 1$ elements as well. In the bi-allelic case, the second (third) column is 1 if SNP_{lj} has value 1 (2) and 0 otherwise and the fourth (fifth) column is 1 if SNP_{rj} has value 1 (2) and 0 otherwise. In this case, when no group is empty, the matrix is given by

$$X_{codominant} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{2.3}$$

When a group is empty, the corresponding row is removed from X and the corresponding element from Y. This process can produce a model matrix that is singular. Trying to fit the model would fail. An obvious way to avoid this would be to check if the matrix is singular or not and act accordingly. However, in practice the matrix is almost never singular and testing the singularity each time would lead to a big waste of time. For this reason, the code is organized as a "try and catch" block. First, the software tries to estimate the model without taking care of this issue. Then, in the rare cases where it fails, the matrix is investigated in details and the linearly dependent columns are removed. From a programming point of view, it is important to realize that the model matrix is strictly the same for most of the SNP pairs. Indeed, the majority of the SNPs are bi-allelic. For this reasons, the $X_{additive}$ and $X_{codominant}$ matrices are hard-coded. In this way, significant computing time can be saved. When empty groups are present, the actual model matrix used is easy to construct from the hard-coded one, by keeping only the rows corresponding to the non-empty groups. When at least one SNP is not bi-allelic, the most common scenarios are again hard-coded and in the very rare other cases, the model matrix is constructed explicitly.

Fitting the model

This section is mainly influenced by [14]. The strategy to fit equation 2.1 is to start from an initial guess and to update it iteratively. This can be achieved using the Newton-Raphson method [19], since the Y vector contains expected outcome values and not binary ones. In fact, for generalized linear models (GLMs), this comes down to iteratively reweighted least squares (IRLS). The implementation is based on the R function *glm.fit* and described in Box 2.2.

Box 2.2. Model fitting using IRLS

(1) Create vectors A, U, T and Y of size $dim = N_1 \times N_2$. Define $\chi^2_{crit} = 2,705543$.

(2) Fill A and U with 0's. For $s = 1, \ldots, S$: if g_{slj} and g_{srj} are not missing do either $A_{g_{slj} \times N_2 + g_{srj}}{++}$ if $c_s = 1$ or $U_{g_{slj} \times N_2 + g_{srj}}{++}$ otherwise.

(3) Create matrix X consisting of $dim_c = 3$ columns in case of additive coding and $dim_c = N_1 + N_2 - 1$ columns in case of codominant coding. Set $a = 0$. For $h = 0, \ldots, dim - 1$: compute $T_a = A_a + U_a$; if $(T_a = 0)$ erase the a^{th} element of vectors A and U, else perform the following operations

(a) Add the a^{th} row of the hard-coded model matrix to X.

(b) Compute $Y_a = \frac{A_a}{T_a}$.

(c) Perform $a{++}$.

Compute $dim = a$.

(4) Compute $N_A = A_0 + \ldots + A_{dim-1}$, $N_U = U_0 + \ldots + U_{dim-1}$ and $N = N_A + N_U$.

(5) Create fitted values vector μ of size dim and initialize all elements to $\frac{N_A}{N}$.

(6) Create linear predictor vector $\eta = ln(\frac{\mu}{1-\mu})$.

(7) Compute $\mathcal{D} = 2\{N \ln N - N_A \ln N_A - N_U \ln N_U + \sum_{i=0}^{dim-1} [A_i \ln(Y_i) + U_i \ln(1-Y_i)]\}$.

(where only the first term of the latter sum is taken if $Y_i = 1$ and only the second if $Y_i = 0$)

(8) Perform the following iterations:

(a) Create a vector $z = \eta + \frac{Y-\mu}{\mu(1-\mu)}$ (taking $z_i = \eta_i - 1$ if $\mu_i = 0$ and $z_i = \eta_i + 1$ if $\mu_i = 1$)

(b) Calculate the $dim \times dim$ diagonal matrix W where $W_{ii} = T_i \mu_i (1 - \mu_i)$.

(c) Calculate $dim_c \times dim_c$ symmetric positive definite matrix $\mathcal{I} = X'WX$.

(d) Calculate the right-hand side vector v of size dim_c defined by $v = X'Wz$.

(e) Calculate vector β of size dim_c by trying to solve $\mathcal{I}\beta = v$. If it fails, remove all linearly dependent columns from X and go back to step (c).

(f) Update $\eta = X\beta$ and $\mu = \frac{1}{1+e^{-\eta}}$.

(g) Compute $\mathcal{D} = 2\{\sum_{i=0}^{dim-1} A_i \ln\frac{Y_i}{\mu_i} + U_i \ln\frac{1-Y_i}{1-\mu_i})$.

(where only the first term is taken if $Y_i = 1$ and only the second if $Y_i = 0$)

(h) Stop iterating if $\frac{|\mathcal{D} - \mathcal{D}_{\text{old}}|}{0.1 + \mathcal{D}} < 10^{-8}$ or after 25 iterations.

At step (5), the fitted values vector μ is initialized to the values under the null model (only intercept), i.e. the trivial model fit consisting of the observed ratios of cases versus individuals in each group. The difficult part is step (8) (e). Because $\mathcal{I}$ is symmetric and positive definite, this linear system can be solved based on the Cholesky decomposition of $\mathcal{I}$, which is guaranteed to exist and be unique if X is not singular:

$$\mathcal{I} = LDL' \tag{2.4}$$

where L is a $dim_c \times dim_c$ lower triangular matrix with diagonal elements equal to 1 and D a $dim_c \times dim_c$ diagonal matrix with strictly positive diagonal elements. The linear system $\mathcal{I}\beta = v$ can then be solved by subsequently solving $Lv_1 = v$ for v_1, $Dv_2 = v1$ for v_2 and $L'\beta = v_2$ for β. Hence, no explicit matrix inversion is involved. In practice, we call a library (alglib) that performs the Cholesky decomposition very fast. As already mentioned, this part of the code is organized as a try-catch block. Indeed, the decomposition can only fail if X is singular (which leads to a matrix $\mathcal{I}$ that is not positive definite) and in this case the linearly dependent columns are removed from X.

Test statistic

So far, we have defined a model and have shown how to fit it. However, there was no obvious link to the MB-MDR methodology. The connection is in fact that, instead of performing simple χ^2 tests as in steps (5), (7) and (8) of Box 2.1, score statistics are used to take account of the model. In each of these steps, the idea was to compare a selection of subjects S_1 versus the others S_0. At step (5), the subjects with one particular genotype versus the rest. At step (7) (respectively 8), the subjects belonging to the "H" (respectively "L") category versus the rest. To simultaneously take account of the main effects and of the group belongings, the model matrix needs to be extended. A column is added at the back of matrix X, whose value is 1 if the row corresponds to subjects belonging to S_1 and 0 otherwise. To avoid confusion in the rest of this section, we note X_{main} the original model matrix (taking only the main effect into account) and X_{c+1} the aforementioned extra column. The score statistic is defined by

$$\mathcal{S} = \frac{u^2}{i} \tag{2.5}$$

where the score u and information i can be computed in different ways. A stable calculation is based on the QR decomposition $Q_1R_1 = \sqrt{W}X_{main}$ [110]. This QR decomposition is thin and needs to be executed only once. The extra column X_{c+1} is explicitly orthogonalized with respect to X_{main} by solving $R_1\mathcal{V} = Q_1^T\sqrt{W}X_{c+1}$ for vector $\mathcal{V}$ and calculating $\mathcal{O} = X_{c+1} - X_{main}\mathcal{V}$. From this, we can calculate

$$u = \sum_i W_i\mathcal{O}_i\rho_i \tag{2.6}$$

$$i = \sum_i W_i\mathcal{O}_i^2 \tag{2.7}$$

where the residual vector ρ of size dim is given by

$$\rho = \frac{Y-\mu}{\mu(1-\mu)} \tag{2.8}$$

Algorithm

In the previous section, we have given all the pieces of the puzzle to compute a number t_j, capturing the degree of association between the pair $[SNP_{lj}, SNP_{rj}]$ and the trait $\mathcal{T}$, adjusted for the main effects of the two SNPs (or environmental variables), using either a codominant or an additive coding scheme. In this section, we sum up everything and give the exact algorithm. The computation still follows the general scheme of Figure 2.6, but the three steps are now as follows:

1) The first step still begins by constructing the affected-subjects and unaffected-subjects matrices, but again coded as vectors. They are obtained in the exact same way as in Section 2.2.1. Then, vectors T and Y are again constructed, while adjusting the size of A and U to take account of empty groups, but with a small difference compared to Section 2.2.1: the model matrix X is as well constructed on the fly during this process as well. It is created empty and at each iteration $h = 0, \ldots, dim - 1$, if there is at least one subject whose genotype satisfies $(SNP_{lj} \times N_2 + SNP_{rj} = h)$, the h^{th} row of the hard coded-matrix is added as the next row of X. Recall that in the rare cases where the hard-coded matrix does not exist, i.e. when either $N_1 > 3$ or $N_2 > 3$, then the row is created explicitly. At the end of this step, A, U, T and Y contains dim elements and X contains dim rows, where dim is the number of non-empty groups.

2) The second step still consists in generating the HLO-matrix, but again coded in vector format. First, the model is fitted as described in Box 2.2 and the QR decomposition $Q_1 R_1 = \sqrt{W} X_{main}$ is obtained by calling a library (alglib) dedicated for this task. Then, a column is added at the back of X. Each R_h element of the HLO-vector depends on a test for association between the trait and the belonging to the genotype group satisfying $(SNP_{lj} \times N_2 + SNP_{rj} = h)$, but this time adjusted for the main effects of the SNPs. For $h = 0, \ldots, dim - 1$, if $T_h < 10$ set the h^{th} element of the HLO-vector to "O". Otherwise, overwrite the last column of X with 0's, except the h^{th} element which is set to 1. Then compute the score statistic from equation 2.5. When this value, following a 1df (one degrees of freedom) χ^2, is below the critical value $\chi^2_{\text{crit}} = 2,705543$, set the h^{th} element of the HLO-vector to "O". Otherwise, set it to either "H" if $u > 0$ or "L" otherwise.

3) If there is neither "H" nor "L" values in the HLO vector, return 0. Otherwise, compute the following two score statistics. First, overwrite the last column of X with 0's, except for the elements for which the corresponding element of the HLO vector is equal to "H" which are set to 1. Then, compute the score statistic from equation 2.5. Second, do exactly the same except that the elements set to 1 are now those corresponding to a "L" value. Of course, if there are no "H" ("L") values, the first (second) number is not computed and the functions obviously returns the second (first) one. Otherwise, the function returns the maximum of the two numbers.

The detailed algorithm is reported in Box 2.3.

Box 2.3. Binary MB-MDR statistic computing (with 1st order correction)

(1) Execute the algorithm from Box 2.2. Find $Q_1R_1 = \sqrt{W}X$.

(2) Create empty vectors $\mathcal{V}$ of size dim_c and $\mathcal{O}$ of size dim. Create residual vector ρ of size dim. For $i = 0, \ldots, dim - 1$: if $(\mu_i = 0)$ set $\rho_i = -1$, else if $(\mu_i = 1)$ set $\rho_i = 1$, else set $\rho_i = \frac{Y_i - \mu_i}{\mu_i(1-\mu_i)}$.

(3) Create empty HLO vector R of size dim. For $h = 0, \ldots, dim - 1$:

   (a) If $(T_h < 10)$ set $R_h =$ "O" and skip steps (b), (c), and (d).

   (b) For $j = 0, \ldots, dim_c - 1$: execute $\mathcal{V}_j = Q_{1hj}\sqrt{W_h}$. For $i = dim_c - 1, \ldots, 0$: do (for $j = i+1, \ldots, dim_c - 1$: $\mathcal{V}_i = \mathcal{V}_i - R_{1ij}\mathcal{V}_j$) and ($\mathcal{V}_i = \frac{\mathcal{V}_i}{R_{1ii}}$).

   (c) For $s = 0, \ldots, dim - 1$: do $(\mathcal{O}_s = 0)$ and (for $t = 0, \ldots, dim_c - 1$: $\mathcal{O}_s = \mathcal{O}_s - X_{st}\mathcal{V}_t$). Execute $\mathcal{O}_h{+}{+}$.

   (d) Set $u = i = 0$. For $s = 0, \ldots, dim - 1$: do $(u{+}{=}W_s\mathcal{O}_s\rho_s)$ and $(i{+}{=}W_s\mathcal{O}_s^2)$. If i=0 set $R_h =$ "O", else calculate $\mathcal{S} = \frac{u^2}{i}$. If $\mathcal{S} < \chi^2_{\text{crit}}$ set $R_h =$ "O", else if $(u > 0)$ set $R_h =$ "H", else set $R_h =$ "L".

(4) If there is no "H" and no "L" in the HLO vector R, return $t_j = 0$.

(5) If there is at least one "H":

   (a) For $i = 0, \ldots, dim_c - 1$: do $(\mathcal{V}_i = 0)$ and (for $j = 0, \ldots, dim - 1$: if ($R_j =$ "H") do ($\mathcal{V}_i = \mathcal{V}_i + Q_{1ji}\sqrt{W_j}$). For $i = dim_c - 1, \ldots, 0$: do (for $j = i+1, \ldots, dim_c - 1$: $\mathcal{V}_i = \mathcal{V}_i - R_{1ij}\mathcal{V}_j$) and ($\mathcal{V}_i = \frac{\mathcal{V}_i}{R_{1ii}}$).

   (b) For $s = 0, \ldots, dim - 1$: do (if ($R_j =$ "H") do $\mathcal{O}_s = 1$ else do $\mathcal{O}_s = 0$) and (for $t = 0, \ldots, dim_c - 1$: do $\mathcal{O}_s = \mathcal{O}_s - X_{st}\mathcal{V}_t$).

   (c) Set $u = i = 0$. For $s = 0, \ldots, dim - 1$: do $(u{+}{=}W_s\mathcal{O}_s\rho_s)$ and $(i{+}{=}W_s\mathcal{O}_s^2)$. If (i>0) compute $\mathcal{S}_{\text{HvsLO}} = \frac{u^2}{i}$ else set it to zero.

(6) If there is at least one "L":

   (a) For $i = 0, \ldots, dim_c - 1$: do $(\mathcal{V}_i = 0)$ and (for $j = 0, \ldots, dim - 1$: if ($R_j =$ "L") do ($\mathcal{V}_i = \mathcal{V}_i + Q_{1ji}\sqrt{W_j}$). For $i = dim_c - 1, \ldots, 0$: do (for $j = i+1, \ldots, dim_c - 1$: $\mathcal{V}_i = \mathcal{V}_i - R_{1ij}\mathcal{V}_j$) and ($\mathcal{V}_i = \frac{\mathcal{V}_i}{R_{1ii}}$).

   (b) For $s = 0, \ldots, dim - 1$: do (if ($R_j =$ "L") do $\mathcal{O}_s = 1$ else do $\mathcal{O}_s = 0$) and (for $t = 0, \ldots, dim_c - 1$: do $\mathcal{O}_s = \mathcal{O}_s - X_{st}\mathcal{V}_t$).

   (c) Set $u = i = 0$. For $s = 0, \ldots, dim - 1$: do $(u{+}{=}W_s\mathcal{O}_s\rho_s)$ and $(i{+}{=}W_s\mathcal{O}_s^2)$. If (i>0) compute $\mathcal{S}_{\text{LvsHO}} = \frac{u^2}{i}$ else set it to zero.

(7) Return $t_j = \max(\mathcal{S}_{\text{HvsLO}}, \mathcal{S}_{\text{LvsHO}})$.

2.2.3 Main effect screening

Before searching for GxG or GxE interactions influencing the disease, users may be interested in solving the simpler task of finding direct genetical or environmental factors. We propose an option in the software to perform such an analysis (the *-d 1D* option) whose implementation is based on the idea that the source code from Section 2.2.1 can be degenerated to search for single SNPs or environmental factors. The input is still the same as in Figure 2.2, but the output is slightly different. The first column no longer contains a ranking of the most significant pairs of SNPs, but a simple ranking $\mathrm{SNP}_{t1}, \dots, \mathrm{SNP}_{tn}$ of the most significant main effects. In this case, the aim is to compute, for every SNP_{tj}, a number t_j capturing its degree of association with the trait. Let N_1 be the number of possible values for SNP_{tj}. The different $N_1 \times N_2$ matrices of Figure 2.5 and 2.6 now becomes vectors of size N_1. Box 2.4 gives the algorithm.

Box 2.4. Binary main effect statistic computing

```
(1) Create vectors $A$, $U$, $T$ and $Y$ of size $dim = N_1$. Define $\chi^2_{crit} = 2,705543$.
(2) Fill $A$ and $U$ with 0's. For $s = 1, \dots, S$: if $g_{slj}$ is not missing do either $A_{g_{slj}}{++}$
    if $c_s = 1$ or $U_{g_{slj}}{++}$ otherwise.
(3) Set $a = 0$. For $h = 0, \dots, dim - 1$: compute $T_a = A_a + U_a$ ; if ($T_a = 0$) erase
    the $a^{\text{th}}$ element of vectors $A$ and $U$, else compute $Y_a = \frac{A_a}{T_a}$ and perform $a{++}$.
    After the loop compute the final dimension: $dim = a$.
(4) Compute $N_A = A_0 + \dots + A_{dim-1}$, $N_U = U_0 + \dots + U_{dim-1}$ and $N = N_A + N_U$.
(5) Create HLO vector $R$ of size $dim$. For $h = 0, \dots, dim - 1$:
    (a) Define $a = A_h$, $b = N_A - a$, $c = U_h$ and $d = N_U - c$.
    (b) If $(a + c) < 10$ or $(b + d) < 10$: set $R_h =$ "O", else compute
        $\chi^2_{obs} = \frac{(ad-bc)^2 N}{N_A N_U (b+d)(a+c)}$. If $\chi^2_{obs} < \chi^2_{crit}$ set $R_h =$ "O", else if
        $(ad - bc) < 0$ set $R_h =$ "H", else set $R_h =$ "L".
(6) If there is no "H" and no "L" in the R vector, return $t_j = 0$.
(7) If there is at least one "H":
    (a) Initialize $a = 0$ and $c = 0$. For $h = 0, \dots, dim - 1$: if $R_h =$ "H" compute
        $a{+}{=}A_h$ and $c{+}{=}U_h$.
    (b) Define $b = N_A - a$ and $d = N_U - c$. Compute $\chi^2_{\mathrm{HvsLO}} = \frac{(ad-bc)^2 N}{N_A N_U (b+d)(a+c)}$.
(8) If there is at least one "L":
    (a) Initialize $a = 0$ and $c = 0$. For $h = 0, \dots, dim - 1$: if $R_h =$ "L" compute
        $a{+}{=}A_h$ and $c{+}{=}U_h$.
    (b) Define $b = N_A - a$ and $d = N_U - c$. Compute $\chi^2_{\mathrm{LvsHO}} = \frac{(ad-bc)^2 N}{N_A N_U (b+d)(a+c)}$.
(9) Return $t_j = \max(\chi^2_{\mathrm{HvsLO}}, \chi^2_{\mathrm{LvsHO}})$.
```

2.2.4 High-dimensional interaction screening

In practical studies, users may be interested in searching for three-order interactions. The user usually chooses between a GxGxG interaction analysis or GxGxE one, but any combination of genes and environmental factors is allowed. The GxGxG interaction analysis is the most computational intensive, since it implies the highest amount of hypothesis to investigate. This section is an extension of the code presented in Section 2.2.1 and 2.2.2. Although the input is the same as in Figure 2.2, the first column of the output is different and contains a ranking $(\text{SNP}_{l1}, \text{SNP}_{m1}, \text{SNP}_{r1}), \ldots, (\text{SNP}_{ln}, \text{SNP}_{mn}, \text{SNP}_{rn})$ of the most important triplets. Our aim is to compute, for every $(\text{SNP}_{lj}, \text{SNP}_{mj}, \text{SNP}_{rj})$ triplet, a number t_j capturing its degree of association with the trait. Let N_1, N_2 and N_3 be the number of possible values for $\text{SNP}_{lj}, \text{SNP}_{mj}$ and SNP_{rj} respectively. The different $N_1 \times N_2$ matrices of Figure 2.5 and 2.6 becomes 3D matrices of size $N_1 \times N_2 \times N_3$. Box 2.5 gives the code computing t_j without main effects correction.

Box 2.5. Binary three-order statistic computing (no 1st order correction)

```
(1) Create vectors $A$, $U$, $T$, $Y$ of size $dim = N_1 \times N_2 \times N_3$. Set $\chi^2_{crit} = 2,705543$.
(2) Fill $A$ and $U$ with 0's. For $s = 1, \ldots, S$: if $g_{slj}$, $g_{smj}$ and $g_{srj}$ are not missing do either $A_{g_{slj} \times N_2 \times N_3 + g_{smj} \times N_3 + g_{srj}}{+}{+}$ if $c_s = 1$ or $U_{g_{slj} \times N_2 \times N_3 + g_{smj} \times N_3 + g_{srj}}{+}{+}$.
(3) Set $a = 0$. For $h = 0, \ldots, dim - 1$: compute $T_a = A_a + U_a$ ; if $(T_a = 0)$ erase the $a^{\text{th}}$ element of vectors $A$ and $U$, else compute $Y_a = \frac{A_a}{T_a}$ and perform $a{+}{+}$. After the loop compute the final dimension: $dim = a$.
(4) Compute $N_A = A_0 + \ldots + A_{dim-1}$, $N_U = U_0 + \ldots + U_{dim-1}$ and $N = N_A + N_U$.
(5) Create HLO vector $R$ of size $dim$. For $h = 0, \ldots, dim - 1$:
    (a) Define $a = A_h$, $b = N_A - a$, $c = U_h$ and $d = N_U - c$.
    (b) If $(a + c) < 10$ or $(b + d) < 10$: set $R_h =$ "O", else compute $\chi^2_{obs} = \frac{(ad-bc)^2 N}{N_A N_U (b+d)(a+c)}$. If $\chi^2_{obs} < \chi^2_{crit}$ set $R_h =$ "O", else if $(ad - bc) < 0$ set $R_h =$ "H", else set $R_h =$ "L".
(6) If there is no "H" and no "L" in the R vector, return $t_j = 0$.
(7) If there is at least one "H":
    (a) Initialize $a = 0$ and $c = 0$. For $h = 0, \ldots, dim - 1$: if $R_h =$ "H" compute $a{+}{=}A_h$ and $c{+}{=}U_h$.
    (b) Define $b = N_A - a$ and $d = N_U - c$. Compute $\chi^2_{\text{HvsLO}} = \frac{(ad-bc)^2 N}{N_A N_U (b+d)(a+c)}$.
(8) If there is at least one "L":
    (a) Initialize $a = 0$ and $c = 0$. For $h = 0, \ldots, dim - 1$: if $R_h =$ "L" compute $a{+}{=}A_h$ and $c{+}{=}U_h$.
    (b) Define $b = N_A - a$ and $d = N_U - c$. Compute $\chi^2_{\text{LvsHO}} = \frac{(ad-bc)^2 N}{N_A N_U (b+d)(a+c)}$.
(9) Return $t_j = \max(\chi^2_{\text{HvsLO}}, \chi^2_{\text{LvsHO}})$.
```

The source code computing t_j with main effects correction can easily be deduced from the algorithms described in Section 2.2.2. Vector Y still contains the ratio of cases versus number of individuals for every non-empty group of the affected-subjects and unaffected-subjects matrices, but these matrices are here 3D ones. The aim is once more to fit the model $E\{Y|X\} = X\beta$, where the number of rows in X is still equal to the number of elements in Y. For an additive correction, the model matrix X_{additive} obviously consists of $c = 4$ columns (one for the intercept and one for each SNP) and for a codominant correction, the model matrix $X_{\text{codominant}}$ now consists of $c = N_1 + N_2 + N_3 - 2$ columns. In the bi-allelic case, when no group is empty, these objects are given explicitly by

$$X_{additive} = \begin{bmatrix} 1&0&0&0\\ 1&0&0&1\\ 1&0&0&2\\ 1&0&1&0\\ 1&0&1&1\\ 1&0&1&2\\ 1&0&2&0\\ 1&0&2&1\\ 1&0&2&2\\ 1&1&0&0\\ 1&1&0&1\\ 1&1&0&2\\ 1&1&1&0\\ 1&1&1&1\\ 1&1&1&2\\ 1&1&2&0\\ 1&1&2&1\\ 1&1&2&2\\ 1&2&0&0\\ 1&2&0&1\\ 1&2&0&2\\ 1&2&1&0\\ 1&2&1&1\\ 1&2&1&2\\ 1&2&2&0\\ 1&2&2&1\\ 1&2&2&2 \end{bmatrix} \qquad X_{codominant} = \begin{bmatrix} 1&0&0&0&0&0&0\\ 1&0&0&0&0&1&0\\ 1&0&0&0&0&0&1\\ 1&0&0&1&0&0&0\\ 1&0&0&1&0&1&0\\ 1&0&0&1&0&0&1\\ 1&0&0&0&1&0&0\\ 1&0&0&0&1&1&0\\ 1&0&0&0&1&0&1\\ 1&1&0&0&0&0&0\\ 1&1&0&0&0&1&0\\ 1&1&0&0&0&0&1\\ 1&1&0&1&0&0&0\\ 1&1&0&1&0&1&0\\ 1&1&0&1&0&0&1\\ 1&1&0&0&1&0&0\\ 1&1&0&0&1&1&0\\ 1&1&0&0&1&0&1\\ 1&0&1&0&0&0&0\\ 1&0&1&0&0&1&0\\ 1&0&1&0&0&0&1\\ 1&0&1&1&0&0&0\\ 1&0&1&1&0&1&0\\ 1&0&1&1&0&0&1\\ 1&0&1&0&1&0&0\\ 1&0&1&0&1&1&0\\ 1&0&1&0&1&0&1 \end{bmatrix} \tag{2.9}$$

The algorithm provided in Box 2.2 is almost readily applicable to the new setting. The only change is at the first step which is replaced by steps (1), (2) and (3) of Box 2.5 (instead of steps (1), (2) and (3) of Box 2.1 before). The algorithm from Box 2.3 does not change at all. In conclusion, adapting the code to three-order interactions is rather straightforward.

Higher-order interactions

It is of course possible to write a source code allowing the user to go for four-order interactions, or even higher. Its is also possible to write a generic code enabling the user to pick any natural number. These options have not been implemented in the software for several reasons, enumerated in the discussion section of this chapter.

Customized analysis

Our software contains a lot of options to tune the analysis to the data at hand. For instance, it can accommodate covariates. This feature is discussed in details in Chapter 4. In Section 2.2.2 we have shown how the software corrects for main effects. This obviously increases the computing time. For this reason, an intermediate option between correcting for main effects or not has been developed. In this case, no correction is performed while the HLO matrices are constructed, but a correction is executed to compute the final t_j value. In other words, the first six steps of Box 2.1 are executed, followed by steps (5), (6) and (7) of Box 2.3. This smooth correction is about 2.5 times faster than the full one [79]. If the user is not interested in all possible combination of SNPs, he can use the different filtering options provided. For instance, he can create two files and investigate only the pairs that can be obtained by taking one SNP from the first file and one SNP from the second file. We refer to the documentation presented in the Appendix for more details.

2.3 Trait expressed on a continuous scale

Studies with patients having different degrees of severity for the disease are widely used in medicine. This degree of severity is often measured on a numeric scale (for instance the lungs capacity of the patient can be used as an indicator of the patient's health for certain diseases). The aim of continuous trait studies is to determine if there is a statistically significant association between the genetic patrimonies of the subjects and their continuous traits or phenotypes. In this thesis, we focus on genetic interactions that might regulate the susceptibility to the disease. This design does not factually restrict to a numerical value representing a disease status. It can in general handle any continuous trait, as height, which is a highly heritable one.

Our aim is to compute for every pair $[SNP_{lj}, SNP_{rj}]$, a number capturing its degree of association with the trait. This enables to sort the pairs by decreasing chances of regulating the disease. Let N_1 and N_2 be the number of possible values for SNP_{lj} and SNP_{rj} respectively. Note that categorical environment variables can still be handled, as long as they are coded $0, 1, \ldots, N_1 - 1$, resp. $0, 1, \ldots, N_2 - 1$.

2.3.1 Without correction for main effects

The starting point of the computation is similar to what was done for a binary trait in Figure 2.5. All subjects are still split into groups depending on their genotypes (nine groups in the bi-allelic case) and the risk of each group is still assessed ("H", "L" or "O"). However, in the case of a continuous trait, the assessment is based on the distribution of the trait values of the subjects across the group. Figure 2.7 illustrates the process with histograms.

In the case of a binary trait, χ^2 tests were used whenever two groups of subjects needed to be compared. The equivalent in the continuous world is the t-test, defined by

$$t = \frac{\bar{X}_1 - \bar{X}_2}{S_{X_1X_2}\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{2.10}$$

where $\bar{X}_1$ ($\bar{X}_2$) is the average trait value for group 1 (2), n_1 (n_2) is the number of subjects in group 1 (2) and $S_{X_1X_2}$ is an estimator of the common standard deviation of the two samples defined by

$$S_{X_1X_2} = \sqrt{\frac{(n_1 - 1)S^2_{X_1} + (n_2 - 1)S^2_{X_2}}{n_1 + n_2 - 2}} \tag{2.11}$$

In this equation, $S^2_{X_1}$ ($S^2_{X_2}$) is the unbiased estimators of the variance of group 1 (2).

This definition of the t-test is customized for unequal sample sizes and equal variance, as suggested by [79]. Note that from a programming point of view, it is much faster to do things slightly differently. First, computing a square root is computational intensive, so instead of computing t, t^2 should be computed. Second, the definition of the unbiased estimator of the variances of group 1 and 2 are given by

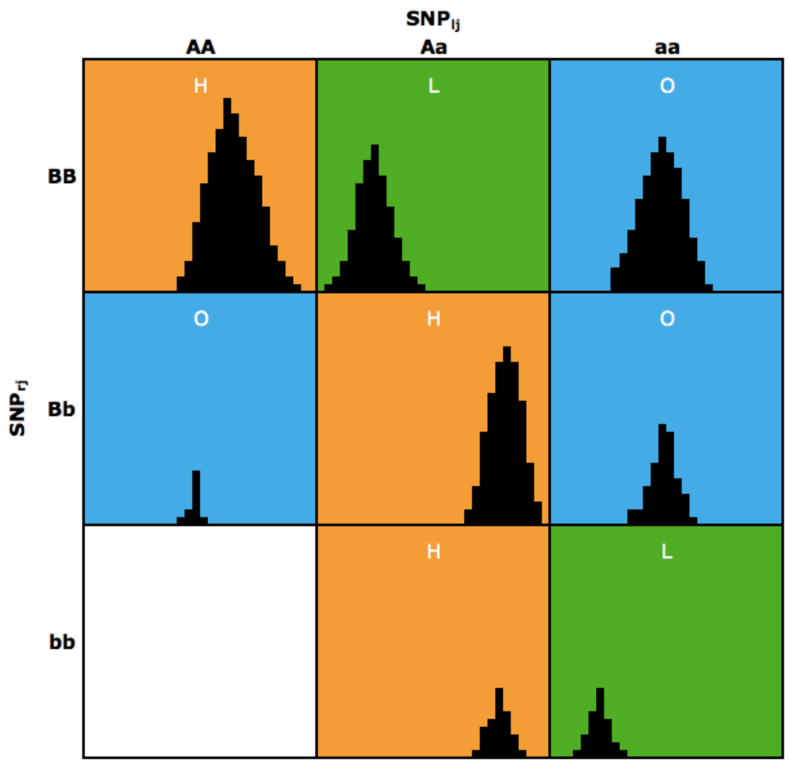


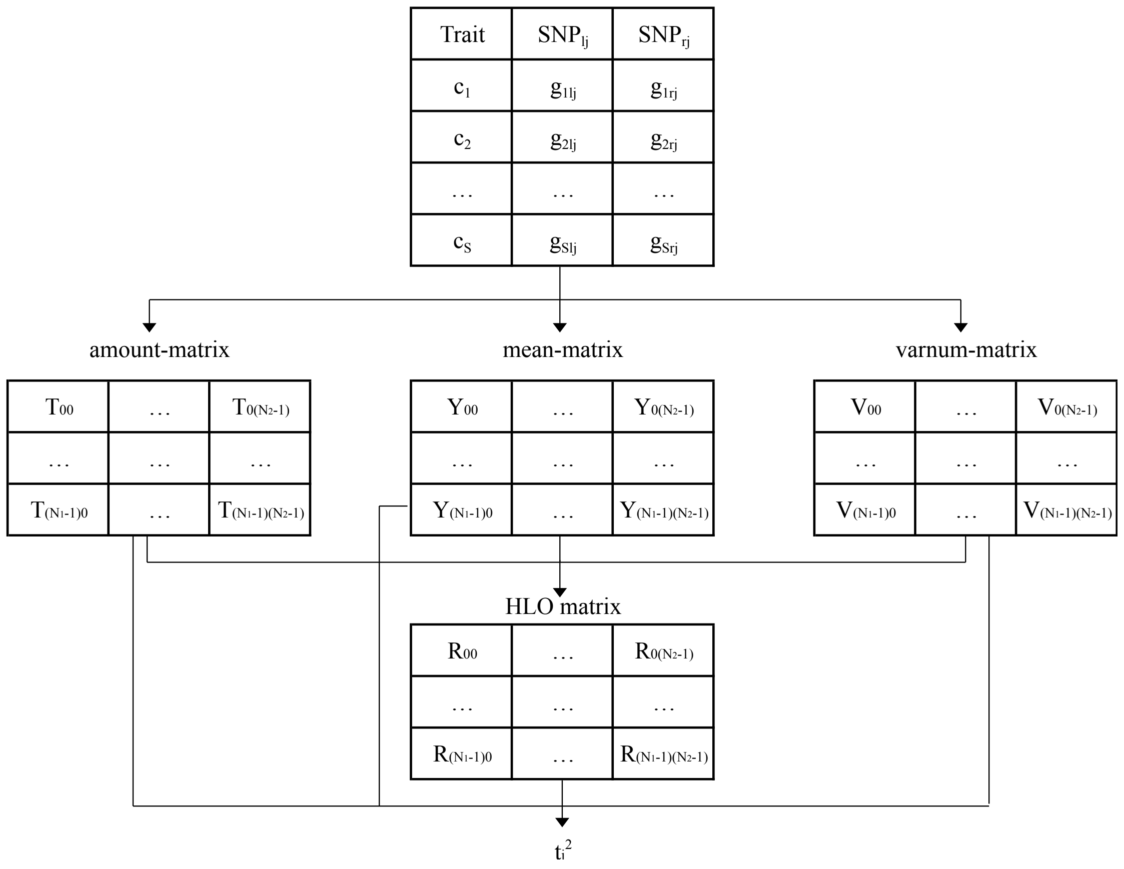
Figure 2.7: For a continuous trait, the subjects are still split into cells depending on their genotypes, in the same way as in Figure 2.5. However, here the cells contain an estimation of the distribution of the continuous trait across the group of subjects having the same genotypes. A cell is immediately assigned to the "O" category (no-evidence for any risk change) if it consists of less than ten individuals. Otherwise, a statistical test is performed to decide if there is a statistically significant difference between the group of subjects belonging to the cell and the group consisting of all other subjects. If there is, the cell is assigned to either the "H" or "L" category, depending on which group has the highest average. Otherwise, it is assigned to the "O" category.

$$S^2_{X_1} = \frac{\sum_{i=1}^{n_1} (\bar{X}_1 - X_i)^2}{n_1 - 1} \tag{2.12}$$

and

$$S^2_{X_2} = \frac{\sum_{i=1}^{n_2} (\bar{X}_2 - X_i)^2}{n_2 - 1} \tag{2.13}$$

A close look at equation 2.11 shows that the $(n_1 - 1)$ and $(n_2 - 1)$ terms are divided by themselves. Since this equation is used billions of times, it is crucial to optimize its implementation and avoid such unnecessary division. For this reason, only the numerator of $S^2_{X_1}$ and $S^2_{X_2}$ is computed. Figure 2.8 sketches the computation of a statistic t^2_j representing the strength of the association between the pair $[SNP_{lj}, SNP_{rj}]$ and the trait $\mathcal{T}$.



Trait
SNP_lj
SNP_rj
c1 g1lj g1rj
c2 g2lj g2rj
cS gSlj gSrj
amount-matrix
mean-matrix
varnum-matrix
T00 T0(N2-1) T(N1-1)0 T(N1-1)(N2-1)
Y00 Y0(N2-1) Y(N1-1)0 Y(N1-1)(N2-1)
V00 V0(N2-1) V(N1-1)0 V(N1-1)(N2-1)
HLO matrix
R00 R0(N2-1) R(N1-1)0 R(N1-1)(N2-1)
ti2

Figure 2.8: MB-MDR statistic computing for a continuous trait, without 1st order correction. Input: c_s is the continuous trait value for the s^{th} subject, g_{slj} and g_{srj} are the genotypes of the s^{th} subject for SNP_{lj} and SNP_{rj} respectively. The computation can be decomposed in three steps. First, the amount, mean and varnum matrices are constructed. T_{mn}, Y_{mn} and V_{mn} are respectively the total number of subjects, the mean and the variance numerator of the group of subjects whose genotype is $g_{slj} = m$ and $g_{srj} = n$. Second, the HLO matrix is constructed. R_{mn} is either "H" if the subjects whose genotype is m for SNP_{lj} and n for SNP_{rj} have a high statistical risk of disease, "L" if they have a low statistical risk and "O" if there is no statistical evidence. Third, the final t_j^2 value is computed from the four matrices constructed at steps 1 and 2.

The three steps of the computation of the number t_j^2 capturing the degree of association between the pair $[SNP_{lj}, SNP_{rj}]$ and the trait $\mathcal{T}$ are as follows:

1) Generation of the amount-, mean- and varnum- matrices. For efficiency reasons these objects are again coded as vectors. They are created with a size $N_1 \times N_2$ and initialized with zeros. Then, several loops are performed. First, a loop over the subjects of the dataset: for $s = 1, \dots S$, if no value is missing for subject s, increment the $(g_{slj} \times N_2 + g_{srj})^{\text{th}}$ element of the amount-vector by one and the $(g_{slj} \times N_2 + g_{srj})^{\text{th}}$ element of the mean-vector by c_s. Second, loop over the elements of the mean-vector: for $h = 0, \dots, (N_1 \times N_2) - 1$, if the h^{th} element of the amount-vector is not equal to zero, divide the h^{th} element of the mean-vector by the h^{th} of the amount-vector. The quantity N of subjects without missing values is computed on the fly during this process. Third, loop over the subjects of the

dataset again: for $s = 1, \dots S$, if no value is missing for subject s increment the $(g_{slj} \times N_2 + g_{srj})^{\text{th}}$ element of the varnum-vector by the square of the difference between the $(g_{slj} \times N_2 + g_{srj})^{\text{th}}$ value of the mean-vector and the c_s value. Finally, remove empty groups from these three vectors. Let dim be their final sizes.

2) Generation of the HLO-matrix from the objects generated at step 1, but coded as vector. The value of each R_h elements depends on an independent two-sample t-test between the subjects having a genotype satisfying $(SNP_{lj} \times N_2 + SNP_{rj} = h)$ and the other ones. For $h = 0, \dots, dim - 1$, we compute the square of equation 2.10. The $n_1, \bar{X}_1$ and numerator of $S^2_{X_1}$ values are readily given by the h^{th} element of the amount-, mean- and varnum-vectors. n_2 is given by $N - n_1$ and $\bar{X}_2$ calculated as the sum of the elements of the mean-vector except the h^{th} one, divided by n_2. The numerator of $S^2_{X_2}$ is a bit more tricky to compute. Each element of the varnum-vector is an expression of the deviance from the corresponding element of the mean-vector. However, we are now interested in the deviance from $\bar{X}_2$ so that these values should be rescaled. Therefore, the numerator of $S^2_{X_2}$ can be obtained by summing all elements of the varnum-matrix except the h^{th} one and each time adding a rescaling factor computed as the product of two terms. First, the square of the difference between $\bar{X}_2$ and the corresponding element of the mean-vector. Second, the corresponding element of the amount-vector. The latter term is in fact a weighting one, to properly take account of the different group sizes. Note that, if either n_1 or n_2 is below a threshold that is a parameter of the program (default value 10) then the test is not performed at all, since it would not be statistically significant. In this case the value of R_h is automatically set to “O”, to indicate the absence of evidence that the subset of individuals with multilocus genotype satisfying $(SNP_{lj} \times N_2 + SNP_{rj} = h)$ has neither a high nor a low risk for disease. Otherwise, the test is performed, based on a critical value F_{crit} defined by an F-distribution with parameters 1 and degrees of freedom N-2 (based on a liberal significance threshold of 0.1) [6]. When the computed t^2 value is below F_{crit} the value of R_h is set to “O”, to indicate that we cannot reject the independence hypothesis. Otherwise, R_h is set to either “H” if $\bar{X}_1 > \bar{X}_2$, to indicate a high risk of having the trait, or to “L” if $\bar{X}_1 < \bar{X}_2$, for a low one.

3) Computation of t_j from the four objects generated at steps 1 and 2. It consists in performing two t-tests and returning the maximum of both. The first one tests independence between the trait and the belonging to the “H” category versus the “L” or “O” category. The second one tests independence between the trait and the belonging to the “L” category versus the “H” or “O” category. In the first (second) case, n_1, $\bar{X}_1$ and the numerator of $S^2_{X_1}$ are respectively the number, the mean and the numerator of the variance of the group of individuals belonging to the “H” (“L”) category and n_2, $\bar{X}_2$ and the numerator of $S^2_{X_2}$ to the “L” (“H”) or “O” category. Computing this can be easily achieved in a similar way as in step 2. Note that in many cases, there are only cells associated to the “O” category and t_j^2 can readily be set to zero.

[6] Indeed, when the degrees of freedom are high, which is the case here since the number of subjects is important, the t^2 distribution comes down to an F distribution, as can be proved by
$t^2 = (\frac{N(0,1)}{\sqrt{\chi^2(p)/p}})^2 = \frac{N^2(0,1)}{\chi^2(p)/p} = \frac{\chi^2(1)}{\chi^2(p)/p} = \frac{\chi^2(1)/1}{\chi^2(p)/p} \sim F(1,p)$

The detailed algorithm computing the t_j^2 statistic is reported in Box 2.6. Note that for efficiency reasons, the implementation is based on a vector B that was not presented so far. The elements of this vector are simply given by the product of the corresponding elements of the amount- and mean-vectors. This vector is not necessary from a conceptual point of view and does therefore not appear in Figure 2.8. However, it is useful in practice since it helps avoiding some operations to be performed several times (within a function that is called billions of times in GWAIs).

Box 2.6. Continuous MB-MDR statistic computing **(no 1st order correction)**

(1) Create vectors T, B, Y, V of size $dim = N_1 \times N_2$.

(2) Fill T and B with 0's. For $s = 1, \ldots, S$: if g_{slj} and g_{srj} are not missing do $T_{g_{slj} \times N_2 + g_{srj}}{++}$ and $B_{g_{slj} \times N_2 + g_{srj}} {+=} c_s$.

(3) Set $N = 0$. For $h = 0, \ldots, dim - 1$: if $(T_h > 0)$ do $Y_h = \frac{B_h}{T_h}$ and $N {+=} T_h$. Define $F_{crit} = F_{0,1}(1, N - 2)$.

(4) Fill V with 0's. For $s = 1, \ldots, S$: if g_{slj} and g_{srj} are not missing do $V_{g_{slj} \times N_2 + g_{srj}} {+=} (Y_{g_{slj} \times N_2 + g_{srj}} - c_s)^2$.

(5) Set $a = 0$. For $h = 0, \ldots, dim - 1$: if $(T_h = 0)$ erase the a^{th} element of vectors T, B, Y and V, else do $a{++}$. After the loop compute $dim = a$.

(6) Create HLO vector R of size dim. For $h = 0, \ldots, dim - 1$:

   (a) Define $n_1 = T_h$, $\bar{X}_1 = Y_h$, $S_{num1} = V_h$ and $n_2 = N - n_1$.

   (b) If $(n_1 < 10)$ or $(n_2 < 10)$ set $R_h =$ "O" and skip steps (c), (d), and (e).

   (c) Set $\bar{X}_2 = 0$. For $i = 0, \ldots, dim - 1$: if $(i \neq h)$ execute $\bar{X}_2 {+=} B_i$. Compute $\bar{X}_2 = \frac{\bar{X}_2}{n_2}$.

   (d) Set $S_{num2} = 0$. For $i = 0, \ldots, dim - 1$: if $(i \neq h)$ execute $S_{num2} {+=} V_i + (Y_i - \bar{X}_2)^2 \times T_i$.

   (e) Compute $t^2 = \frac{(\bar{X}_1 - \bar{X}_2)^2 (n_1 + n_2 - 2)}{(S_{num1} + S_{num2})(\frac{1}{n_1} + \frac{1}{n_2})}$. If $t^2 < F_{crit}$ set $R_h =$ "O", else if $\bar{X}_1 > \bar{X}_2$ set $R_h =$ "H", else set $R_h =$ "L".

(7) If there is no "H" and no "L" in the R matrix, return $t_j^2 = 0$.

(8) If there is at least one "H":

   (a) Initialize $n_1 = 0$, $\bar{X}_1 = 0$, $S_{num1} = 0$, $n_2 = 0$, $\bar{X}_2 = 0$ and $S_{num2} = 0$.

   (b) For $h = 0, \ldots, dim - 1$: if $R_h =$ "H" execute $n_1 {+=} T_h$ and $\bar{X}_1 {+=} B_h$, else $n_2 {+=} T_h$ and $\bar{X}_2 {+=} B_h$. Compute $\bar{X}_1 = \frac{\bar{X}_1}{n_1}$ and $\bar{X}_2 = \frac{\bar{X}_2}{n_2}$.

   (c) For $h = 0, \ldots, dim - 1$: if $R_h =$ "H" execute $S_{num1} {+=} V_h + (Y_h - \bar{X}_1)^2 \times T_h$, else execute $S_{num2} {+=} V_h + (Y_h - \bar{X}_2)^2 \times T_h$.

   (d) Compute $t^2_{\text{HvsLO}} = \frac{(\bar{X}_1 - \bar{X}_2)^2 (n_1 + n_2 - 2)}{(S_{num1} + S_{num2})(\frac{1}{n_1} + \frac{1}{n_2})}$

(9) If there is at least one "L":

(a) Initialize $n_1 = 0$, $\bar{X}_1 = 0$, $S_{num1} = 0$, $n_2 = 0$, $\bar{X}_2 = 0$ and $S_{num2} = 0$.

(b) For $h = 0, \dots, dim-1$: if $R_h =$ "L" execute $n_1 {+}{=} T_h$ and $\bar{X}_1 {+}{=} B_h$, else $n_2 {+}{=} T_h$ and $\bar{X}_2 {+}{=} B_h$. Compute $\bar{X}_1 = \frac{\bar{X}_1}{n_1}$ and $\bar{X}_2 = \frac{\bar{X}_2}{n_2}$.

(c) For $h = 0, \dots, dim-1$: if $R_h =$ "L" execute $S_{num1} {+}{=} V_h + (Y_h - \bar{X}_1)^2 \times T_h$, else execute $S_{num2} {+}{=} V_h + (Y_h - \bar{X}_2)^2 \times T_h$.

(d) Compute $t^2_{\text{LvsHO}} = \frac{(\bar{X}_1 - \bar{X}_2)^2 (n_1 + n_2 - 2)}{(S_{num1} + S_{num2})(\frac{1}{n_1} + \frac{1}{n_2})}$

(10) Return $t_j^2 = \max(t^2_{\text{HvsLO}}, t^2_{\text{LvsHO}})$.

2.3.2 With correction for main effects

In the previous section, we have proposed a strategy to compute a generic epistasis test. However, this is not what we advocate by default. Indeed, users performing a GWAIs are most of the time interested in finding pure epistatic effects, since they already know the main effects from the literature or a previous analysis realized with software customized for this task. The implementation presented in the previous section would detect any interaction linked to the trait, regardless of the fact that it comes from the main effect of one of the SNP involved in the interaction or from a pure interaction. In this section, we describe the default option of the software in the case of a continuous trait, correcting for lower-order effects.

Coding scheme

The general idea of the lower-order correction is as follows. First, construct the amount-matrix, mean-matrix and varnum-matrix as in Figure 2.8, but store them again in vectors T, Y and V. Empty groups are not added at all to these vectors. Let dim be their final sizes. Then, try to fit the following model:

$$E\{Y|X\} = X\beta \tag{2.14}$$

where X is the model matrix and β a row vector of parameters to fit.

The model matrix is the same as for a binary trait. For an additive correction, in the bi-allelic case, when no group is empty, it is given explicitly by

$$X_{additive} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 2 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 0 \\ 1 & 2 & 1 \\ 1 & 2 & 2 \end{bmatrix} \tag{2.15}$$

For a codominant main effects correction, when no group is empty, it is given explicitly by

$$X_{codominant} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{2.16}$$

When a group is empty, the corresponding row is removed from matrix X and the corresponding element is removed from vector Y. This process can again produce a model matrix that is singular. This issue is handled in the same way as for a binary trait. First, the software tries to estimate the model without taking care of this issue. Then, in the rare cases where it fails, the matrix is investigated in details and the linearly dependent columns are removed.

For efficiency reasons, the aforementioned $X_{additive}$ and $X_{codominant}$ matrices are still hard-coded in the software. In this way, significant computing time can be saved. When empty groups are present, the actual model matrix used for the corresponding pair is easy to construct from the hard-coded one, by keeping only the rows corresponding to the non-empty groups. When at least one SNP is not bi-allelic, the most common scenarios are again hard-coded (in fact, all scenario where $N_1 \leq 3$ and $N_2 \leq 3$) and in the very rare other cases, the model matrix needs to be constructed explicitly.

Fitting the model

This section is mainly influenced by [14]. In the case of a continuous trait, a closed form to obtain the parameter estimates in equation 2.14 exists. Therefore, there is no need to use a Newton-Raphson type of strategy, as was the case for a binary trait. The different steps are given in Box 2.7.

Box 2.7. Model fitting for a continuous trait

(1) Create $dim \times dim$ diagonal weights matrix W. The diagonal elements correspond to the elements of the amount-vector.

(2) Calculate the $c \times c$ symmetric positive definite information matrix $\mathcal{I} = X'WX$.

(3) Calculate the right-hand side vector $v = X'WY$ of size dim_c.

(4) Estimate parameter vector β of size dim_c by solving the linear system $\mathcal{I}\beta = v$. If it fails, find and remove all linearly dependent columns from X and go back to step (2).

(5) Create fitted values vector μ of size dim as $\mu = X\beta$.

The difficult part is step (4). Because $\mathcal{I}$ is symmetric and positive definite, this linear system can again be solved based on the Cholesky decomposition of $\mathcal{I}$, which is guaranteed to exist and be unique if X is not singular:

$$\mathcal{I} = LDL' \tag{2.17}$$

where L is a $c \times c$ lower triangular matrix with diagonal elements equal to 1 and D a $c \times c$ diagonal matrix with strictly positive diagonal elements. The linear system $\mathcal{I}\beta = v$ can then be solved by subsequently solving $Lv_1 = v$ for v_1, $Dv_2 = v1$ for v_2 and $L'\beta = v_2$ for β. Hence, no explicit matrix inversion is involved. In practice, we call a library (alglib) that performs the Cholesky decomposition very fast. As already mentioned, this part of the code is organized as a try-catch block. Indeed, the decomposition can only fail if X is singular (which leads to a matrix $\mathcal{I}$ that is not positive definite) and in this case the linearly dependent columns are removed from X to ensure that the decomposition succeeds.

Wald statistic

To make use of the fitted model, simple t-tests can no longer be used as in steps (6), (8) and (9) of Box 2.6. In each of these three steps, the idea was to compare a selection of subjects S_1 versus the others S_0. At step (6), the subjects with one particular genotype versus the rest. At step (7) (respectively 8), the subjects belonging to the "H" (respectively "L") category versus the rest. These comparisons should now take into account the main effects and the group belongings simultaneously. To achieve this, the model is again extended by adding a column at the back of matrix X, whose value is 1 if the row corresponds to subjects belonging to S_1 and 0 otherwise. Note that the whole process of calculating the weights matrix W, the information matrix $\mathcal{I}$, its Cholesky decomposition and the right-hand-side vector v is done column by column [14]. Therefore, this only needs to be performed for the one extra column here. Note that such optimization is only possible because the implementation is done explicitly. Calling a package that would take care of the model fitting would not enable such optimizations so easily.

Wald statistic enables to test if the precision of the model gets statistically significantly better by adding the extra column or not. The computation is based on the sum of squared errors (SSE) in both scenarios. To avoid confusion, we note SSE_{main} the one without the extra column (taking only the main effects into account) and SSE_{main+} the one with the group-belongings column. In our context, the Wald statistic is also an F statistic, since the two are the same for a 1df test. It can be defined by

$$\mathcal{W} = (SSE_{main} - SSE_{main+})\frac{N-(c+1)}{SSE_{main+}} \tag{2.18}$$

where N is the total number of subjects considered.

Denoting by σ_i^2 the within genotype cell variance, the SSE of the full model (main effects and epistasis) is given by

$$SSE_{full} = \sum_i (n_i - 1)\sigma_i^2 \tag{2.19}$$

Note that for each term of this sum, the computation comes down to the calculation of the numerator of the definition of σ_i, since the denominator simplifies itself with the $(n_i - 1)$ term, similarly as what was done in the case of a binary trait. The two SSE from equation 2.18 can be now be defined as

$$SSE_{main} = SSE_{full} + \sum_i n_i(Y_i - \mu_i)^2 \tag{2.20}$$

and

$$SSE_{main+} = SSE_{main} + \sum_i n_i(Y_i - \mu_i^{main+})^2 \tag{2.21}$$

Note that in equation 2.20, μ is as expected the vector from Box 2.7. In equation 2.21, μ^{main+} can be assessed trivially for the extra column by

$$\mu_i^{main+} = m_1 I(i \in S_1) + m_0 I(i \in S_0) \tag{2.22}$$

where

$$m_j = \frac{\sum_{i \in S_j} n_i Y_i}{\sum_{i \in S_j} n_i} \tag{2.23}$$

are the means within the selected and non-selected groups.

Significance of the Wald test is assessed with respect to the F distribution with degrees of freedom 1 and $N - (c + 1)$. When a significant group is found during the HLO matrix construction, it is assigned to the "H" category when $m_1 > m_0$ and to the "L" one otherwise. Note that in practice we test if $\beta_{c+1} > 0$, since this is obviously the same as testing if $m_1 > m_0$.

Algorithm

This section describes the algorithm computing a number t_j, capturing the degree of association between the pair $[SNP_{lj}, SNP_{rj}]$ and the trait $\mathcal{T}$, adjusted for the main effects of the two SNPs (or environmental variables), using either a codominant or an additive coding scheme. The computation still follows the general strategy of Figure 2.8, but the three steps are now as follows:

1) The first step still begins by constructing the amount-, mean- and varnum- matrices, but again coded as vectors for efficiency and facility reasons. They are created with a size $N_1 \times N_2$ and initialized with zeros. Then, several loops are performed. First, a loop over the subjects of the dataset: for $s = 1, \dots S$, if no value is missing for subject s, increment the $(g_{slj} \times N_2 + g_{srj})^{\text{th}}$ element of the amount-vector by one and the $(g_{slj} \times N_2 + g_{srj})^{\text{th}}$ element of the mean-vector by c_s. Second, loop over the elements of the mean-vector: for $h = 0, \dots, (N_1 \times N_2) - 1$, if the h^{th} element of the amount-vector is not equal to zero, divide the h^{th} element of the mean-vector by the h^{th} of the amount-vector. The quantity N of subjects without missing values is computed on the fly during this process. Third, loop over the subjects of the dataset again: for $s = 1, \dots S$, if no value is missing for subject s increment the $(g_{slj} \times N_2 + g_{srj})^{\text{th}}$ element of the varnum-vector by the square of the difference between the $(g_{slj} \times N_2 + g_{srj})^{\text{th}}$ value of the mean-vector and the c_s value. Finally, remove empty groups from the amount-, mean- and varnum- vectors and on the fly construct the model matrix from the hard-coded one (or explicitly in the rare scenarios explained earlier) and compute the SSE_{full} value. Let dim be the final size of the vectors.

2) The second step still consists in generating the HLO-matrix, but again coded in vector format. First, the model is fitted as described in Box 2.7. Then, a column is added to X. For $h = 1, \ldots, dim - 1$, if the h^{th} element of the amount-matrix is less than 10, set the h^{th} element of the HLO-vector to "O", to indicate the absence of evidence that the subset of individuals with multilocus genotype satisfying $SNP_{lj} \times N_2 + SNP_{rj} = h$ has neither a high nor a low risk for disease. Otherwise, overwrite the last column of X with 0's, except the h^{th} element which is set to 1. Then compute the Wald statistic from equation 2.18. When this value is below the critical value F_{crit} defined by an F distribution with degrees of freedom 1 and $N - (c + 1)$, set the h^{th} element of the HLO-vector to "O". Otherwise, set it to either "H" if $\beta_{c+1} > 0$, to indicate a high risk for disease, or to "L" to indicate a low risk for the disease.

3) If there are neither "H" nor "L" values in the HLO vector, return 0. Otherwise, compute the following two Wald statistics and return the maximum. First, overwrite the last column of X with 0's, except for the elements corresponding to HLO-vector cells equal to "H", which are set to 1. Then, compute again the Wald statistic from equation 2.18. Second, do exactly the same except that the elements set to 1 are now those corresponding to a "L" value. Of course, if there are no "H" ("L") values, the first (second) number is not computed and the function readily returns the second (first) one.

The algorithm is reported in Box 2.8. Note that for efficiency reasons, the code is again based on a vector B which is the elements-wise product of T and Y.

Box 2.8. Continuous MB-MDR statistic computing **(with 1st order correction)**

```
(1) Create vectors $T$, $B$, $Y$, $V$ of size $dim = N_1 \times N_2$.
(2) Fill $T$ and $B$ with 0's. For $s = 1, \ldots, S$: if $g_{slj}$ and $g_{srj}$ are not missing do
    $T_{g_{slj} \times N_2 + g_{srj}}{++}$ and $B_{g_{slj} \times N_2 + g_{srj}} {+=} c_s$.
(3) Set $N = 0$. For $h = 0, \ldots, dim - 1$: if $(T_h > 0)$ do $Y_h = \frac{B_h}{T_h}$ and $N {+=} T_h$.
    Define $F_{crit} = F_{0,1}(1, N - 2)$.
(4) Fill $V$ with 0's. For $s = 1, \ldots, S$: if $g_{slj}$ and $g_{srj}$ are not missing do
    $V_{g_{slj} \times N_2 + g_{srj}} {+=} (Y_{g_{slj} \times N_2 + g_{srj}} - c_s)^2$.
(5) Set $a = 0$. Create empty matrix $X$. Set $SSE_{full} = 0$. For $h = 0, \ldots, dim - 1$:
    if $(T_h = 0)$ erase the $a^{\text{th}}$ element of vectors $T$, $B$, $Y$ and $V$, else take the $h^{th}$
    row of the hard coded model matrix and add it to $X$, execute $SSE_{full} {+=} V_a$
    and do $a{++}$. After the loop compute $dim = a$.
(6) Execute the algorithm from Box 2.7. Initialize $SSE_{main} = SSE_{full}$.
    For $h = 0, \ldots, dim - 1$: execute $SSE_{main} {+=} (Y_h - \mu_h)^2 \times T_h$.
(7) Update the sizes of $X$ to $dim \times (c+1)$, $\mathcal{I}$ to $(c+1) \times (c+1)$ and $v$ to $c+1$.
```

(8) Create HLO vector R of size dim. For $h = 0, \dots, dim - 1$:

   (a) If $T_h < 10$: set $R_h =$ "O" and skip steps (b), (c) and (d).

   (b) Overwrite the last column of X with 0's except at the h^{th} row put to 1.

   (c) Execute steps (2) to (5) of Box 2.7. Initialize $SSE_{main+} = SSE_{main}$. For $h = 0, \dots, dim - 1$: execute $SSE_{main+} += (Y_h - \mu_h)^2 \times T_h$.

   (d) Compute $W = (SSE_{main} - SSE_{main+})\frac{N-(c+1)}{SSE_{main+}}$. If $W < F_{crit}$ set $R_h =$ "O", else if $\beta_{c+1} > 0$ set $R_h =$ "H", else set $R_h =$ "L".

(9) If there is no "H" and no "L" in the HLO vector R, return $t_j = 0$.

(10) If there is at least one "H":

   (a) Overwrite the last column of X with 0's except for rows corresponding to elements of R equals to "H" which are set to 1.

   (b) Execute steps (2) to (5) of Box 2.7. Initialize $SSE_{main+} = SSE_{main}$. For $h = 0, \dots, dim - 1$: execute $SSE_{main+} += (Y_h - \mu_h)^2 \times T_h$.

   (c) Compute $\mathcal{W}_{\text{HvsLO}} = (SSE_{main} - SSE_{main+})\frac{N-(c+1)}{SSE_{main+}}$.

(11) If there is at least one "L":

   (a) Overwrite the last column of X with 0's except for rows corresponding to elements of R equals to "L" which are set to 1.

   (b) Execute steps (2) to (5) of Box 2.7. Initialize $SSE_{main+} = SSE_{main}$. For $h = 0, \dots, dim - 1$: execute $SSE_{main+} += (Y_h - \mu_h)^2 \times T_h$.

   (c) Compute $\mathcal{W}_{\text{LvsHO}} = (SSE_{main} - SSE_{main+})\frac{N-(c+1)}{SSE_{main+}}$.

(12) Return $t_j = \max(\mathcal{W}_{\text{HvsLO}}, \mathcal{W}_{\text{LvsHO}})$.

2.3.3 Main effect screening

For the same reasons as for a binary trait, users may first be interested in performing a main effect analysis. Performing different kinds of analysis on the same dataset helps indeed to get a deeper understanding of the biological processes regulating the disease. The code for performing a main effect analysis is obtained by degenerating the algorithm from Box 2.6. The input is still the same as in Figure 2.2, but the output is again slightly different. The first column no longer contains a ranking of the most significant pairs $(\text{SNP}_{l1}, \text{SNP}_{r1}), \dots, (\text{SNP}_{ln}, \text{SNP}_{rn})$, but a simple ranking $\text{SNP}_{t1}, \dots, \text{SNP}_{tn}$ of the most significant main effects or environmental factors. Here our aim is to compute, for every SNP_{tj}, a number t_j capturing its degree of association with the trait. Let N_1 be the number of possible values for SNP_{tj}. The different $N_1 \times N_2$ matrices of Figure 2.7 and 2.8 now becomes vectors of size N_1. Box 2.9 gives the detailed algorithm.

Box 2.9. Continuous main effect statistic computing

(1) Create vectors T, B, Y, V of size $dim = N_1$. Fill T and B with 0's. For $s = 1, \ldots, S$: if g_{stj} is not missing do $T_{g_{stj}}$++ and $B_{g_{stj}}$+=c_s.

(2) Set $N = 0$. For $h = 0, \ldots, dim - 1$: if $(T_h > 0)$ do $Y_h = \frac{B_h}{T_h}$ and N+=T_h. Define $F_{crit} = F_{0,1}(1, N - 2)$.

(3) Fill V with 0's. For $s = 1, \ldots, S$: if g_{stj} is not missing do $V_{g_{stj}}$+=$(Y_{g_{stj}} - c_s)^2$.

(4) Set $a = 0$. For $h = 0, \ldots, dim - 1$: if $(T_h = 0)$ erase the a^{th} element of vectors T, B, Y and V, else do a++. After the loop compute $dim = a$.

(5) Create HLO vector R of size dim. For $h = 0, \ldots, dim - 1$:

  (a) Define $n_1 = T_h$, $\bar{X}_1 = Y_h$, $S_{num1} = V_h$ and $n_2 = N - n_1$.

  (b) If $(n_1 < 10)$ or $(n_2 < 10)$ set R_h = "O" and skip steps (c), (d), and (e).

  (c) Set $\bar{X}_2 = 0$. For $i = 0, \ldots, dim - 1$: if $(i \neq h)$ execute $\bar{X}_2$+=B_i. Compute $\bar{X}_2 = \frac{\bar{X}_2}{n_2}$.

  (d) Set $S_{num2} = 0$. For $i = 0, \ldots, dim - 1$: if $(i \neq h)$ execute S_{num2}+=$V_i + (Y_i - \bar{X}_2)^2 \times T_i$.

  (e) Compute $t^2 = \frac{(\bar{X}_1 - \bar{X}_2)^2 (n_1 + n_2 - 2)}{(S_{num1} + S_{num2})(\frac{1}{n_1} + \frac{1}{n_2})}$. If $t^2 < F_{crit}$ set R_h = "O", else if $\bar{X}_1 > \bar{X}_2$ set R_h = "H", else set R_h = "L".

(6) If there is no "H" and no "L" in the R matrix, return $t_j^2 = 0$.

(7) If there is at least one "H":

  (a) Initialize $n_1 = 0$, $\bar{X}_1 = 0$, $S_{num1} = 0$, $n_2 = 0$, $\bar{X}_2 = 0$ and $S_{num2} = 0$.

  (b) For $h = 0, \ldots, dim - 1$: if R_h = "H" execute n_1+=T_h and $\bar{X}_1$+=B_h, else n_2+=T_h and $\bar{X}_2$+=B_h. Compute $\bar{X}_1 = \frac{\bar{X}_1}{n_1}$ and $\bar{X}_2 = \frac{\bar{X}_2}{n_2}$.

  (c) For $h = 0, \ldots, dim - 1$: if R_h = "H" do S_{num1}+=$V_h + (Y_h - \bar{X}_1)^2 \times T_h$, else do S_{num2}+=$V_h + (Y_h - \bar{X}_2)^2 \times T_h$.

  (d) Compute $t^2_{\text{HvsLO}} = \frac{(\bar{X}_1 - \bar{X}_2)^2 (n_1 + n_2 - 2)}{(S_{num1} + S_{num2})(\frac{1}{n_1} + \frac{1}{n_2})}$

(8) If there is at least one "L":

  (a) Initialize $n_1 = 0$, $\bar{X}_1 = 0$, $S_{num1} = 0$, $n_2 = 0$, $\bar{X}_2 = 0$ and $S_{num2} = 0$.

  (b) For $h = 0, \ldots, dim - 1$: if R_h = "L" execute n_1+=T_h and $\bar{X}_1$+=B_h, else n_2+=T_h and $\bar{X}_2$+=B_h. Compute $\bar{X}_1 = \frac{\bar{X}_1}{n_1}$ and $\bar{X}_2 = \frac{\bar{X}_2}{n_2}$.

  (c) For $h = 0, \ldots, dim - 1$: if R_h = "L" do S_{num1}+=$V_h + (Y_h - \bar{X}_1)^2 \times T_h$, else do S_{num2}+=$V_h + (Y_h - \bar{X}_2)^2 \times T_h$.

  (d) Compute $t^2_{\text{LvsHO}} = \frac{(\bar{X}_1 - \bar{X}_2)^2 (n_1 + n_2 - 2)}{(S_{num1} + S_{num2})(\frac{1}{n_1} + \frac{1}{n_2})}$

(9) Return $t_j^2 = \max(t^2_{\text{HvsLO}}, t^2_{\text{LvsHO}})$.

2.3.4 High-dimensional interaction screening

In practical studies, users may be interested in searching for three-order interactions. The user can chose between a GxGxG interaction analysis, a GxGxE one, or any other three-way combination of G's and E's. To perform a GxGxE analysis, the user just needs to use a filtering option[7] to ensure that only triplets containing the environmental factor are investigated.

The code without correction for main effects is a generalization of the algorithm described in Section 2.3.1. The input is still the same as in Figure 2.2, but the output is as usual slightly different. The first column no longer contains a ranking of the most significant SNP pairs $(\text{SNP}_{l1}, \text{SNP}_{r1}), \ldots, (\text{SNP}_{ln}, \text{SNP}_{rn})$, but a ranking of the most important SNP triplets $(\text{SNP}_{l1}, \text{SNP}_{m1}, \text{SNP}_{r1}), \ldots, (\text{SNP}_{ln}, \text{SNP}_{mn}, \text{SNP}_{rn})$. Here, our aim is to compute, for every triplet $(\text{SNP}_{lj}, \text{SNP}_{mj}, \text{SNP}_{rj})$, a number t_j capturing its degree of association with the trait, without trying to correct for the main effects of the SNPs. Let N_1, N_2 and N_3 be the number of possible values for $\text{SNP}_{lj}, \text{SNP}_{mj}$ and SNP_{rj} respectively. The different $N_1 \times N_2$ matrices of Figure 2.7 and 2.8 now becomes 3D matrices of size $N_1 \times N_2 \times N_3$. Box 2.10 gives the detailed algorithm.

Box 2.10. Continuous 3D statistic computing **(no 1st order correction)**

(1) Create vectors T, B, Y, V of size $dim = N_1 \times N_2 \times N_3$. Fill T and B with 0's. For $s = 1, \ldots, S$: if g_{slj}, g_{smj} and g_{srj} are not missing do $T_{g_{slj} \times N_2 \times N_3 + g_{smj} \times N_3 + g_{srj}}$++ and $B_{g_{slj} \times N_2 \times N_3 + g_{smj} \times N_3 + g_{srj}} \mathrel{+}= c_s$.

(2) Set $N = 0$. For $h = 0, \ldots, dim - 1$: if $(T_h > 0)$ do $Y_h = \frac{B_h}{T_h}$ and $N \mathrel{+}= T_h$. Define $F_{crit} = F_{0,1}(1, N-2)$.

(3) Fill V with 0's. For $s = 1, \ldots, S$: if g_{slj}, g_{smj} and g_{srj} are not missing do $V_{g_{slj} \times N_2 \times N_3 + g_{smj} \times N_3 + g_{srj}} \mathrel{+}= (Y_{g_{slj} \times N_2 \times N_3 + g_{smj} \times N_3 + g_{srj}} - c_s)^2$.

(4) Set $a = 0$. For $h = 0, \ldots, dim - 1$: if $(T_h = 0)$ erase the a^{th} element of vectors T, B, Y and V, else do a++. After the loop compute $dim = a$.

(5) Create HLO vector R of size dim. For $h = 0, \ldots, dim - 1$:

  (a) Define $n_1 = T_h$, $\bar{X}_1 = Y_h$, $S_{num1} = V_h$ and $n_2 = N - n_1$.

  (b) If $(n_1 < 10)$ or $(n_2 < 10)$ set $R_h =$ "O" and skip steps (c), (d), and (e).

  (c) Set $\bar{X}_2 = 0$. For $i = 0, \ldots, dim - 1$: if $(i \neq h)$ execute $\bar{X}_2 \mathrel{+}= B_i$. Compute $\bar{X}_2 = \frac{\bar{X}_2}{n_2}$.

  (d) Set $S_{num2} = 0$. For $i = 0, \ldots, dim - 1$: if $(i \neq h)$ execute $S_{num2} \mathrel{+}= V_i + (Y_i - \bar{X}_2)^2 \times T_i$.

  (e) Compute $t^2 = \frac{(\bar{X}_1 - \bar{X}_2)^2 (n_1 + n_2 - 2)}{(S_{num1} + S_{num2})(\frac{1}{n_1} + \frac{1}{n_2})}$. If $t^2 < F_{crit}$ set $R_h =$ "O", else if $\bar{X}_1 > \bar{X}_2$ set $R_h =$ "H", else set $R_h =$ "L".

[7] The -f option allows users to investigate only the triplets composed of exactly one marker from the comma-separated list of markers passed as argument, while the -F option does the same, but instead of passing the marker names as arguments, they are read from a file.

(6) If there is no "H" and no "L" in the R matrix, return $t_j^2 = 0$.

(7) If there is at least one "H":

(a) Initialize $n_1 = 0$, $\bar{X}_1 = 0$, $S_{num1} = 0$, $n_2 = 0$, $\bar{X}_2 = 0$ and $S_{num2} = 0$.

(b) For $h = 0, \ldots, dim - 1$: if $R_h =$ "H" execute $n_1 {+}{=} T_h$ and $\bar{X}_1 {+}{=} B_h$, else $n_2 {+}{=} T_h$ and $\bar{X}_2 {+}{=} B_h$. Compute $\bar{X}_1 = \frac{\bar{X}_1}{n_1}$ and $\bar{X}_2 = \frac{\bar{X}_2}{n_2}$.

(c) For $h = 0, \ldots, dim-1$: if $R_h =$ "H" do $S_{num1} {+}{=} V_h + (Y_h - \bar{X}_1)^2 \times T_h$, else do $S_{num2} {+}{=} V_h + (Y_h - \bar{X}_2)^2 \times T_h$. Compute $t^2_{\text{HvsLO}} = \frac{(\bar{X}_1 - \bar{X}_2)^2 (n_1 + n_2 - 2)}{(S_{num1} + S_{num2})(\frac{1}{n_1} + \frac{1}{n_2})}$

(8) If there is at least one "L":

(a) Initialize $n_1 = 0$, $\bar{X}_1 = 0$, $S_{num1} = 0$, $n_2 = 0$, $\bar{X}_2 = 0$ and $S_{num2} = 0$.

(b) For $h = 0, \ldots, dim - 1$: if $R_h =$ "L" execute $n_1 {+}{=} T_h$ and $\bar{X}_1 {+}{=} B_h$, else $n_2 {+}{=} T_h$ and $\bar{X}_2 {+}{=} B_h$. Compute $\bar{X}_1 = \frac{\bar{X}_1}{n_1}$ and $\bar{X}_2 = \frac{\bar{X}_2}{n_2}$.

(c) For $h = 0, \ldots, dim-1$: if $R_h =$ "L" do $S_{num1} {+}{=} V_h + (Y_h - \bar{X}_1)^2 \times T_h$, else do $S_{num2} {+}{=} V_h + (Y_h - \bar{X}_2)^2 \times T_h$. Compute $t^2_{\text{LvsHO}} = \frac{(\bar{X}_1 - \bar{X}_2)^2 (n_1 + n_2 - 2)}{(S_{num1} + S_{num2})(\frac{1}{n_1} + \frac{1}{n_2})}$

(9) Return $t_j^2 = \max(t^2_{\text{HvsLO}}, t^2_{\text{LvsHO}})$.

The code correcting for main effects is a generalization of the algorithm presented in Section 2.3.2. Vector Y still contains the mean trait value of every non-empty group and the aim is again to fit the model $Y = X\beta$. For an additive correction, the model matrix X_{additive} now consists of $c = 4$ columns (one for the intercept and one for each SNP) and for a codominant correction, the model matrix $X_{\text{codominant}}$ now consists of $c = N_1 + N_2 + N_3 - 2$ columns. In the bi-allelic case, when no group is empty, these objects are exactly the same as in the case of a binary trait reported in equations 2.9. The algorithm from Box 2.7 can be used without any modification to fit the model. The algorithm from Box 2.8 requires a slight modification: steps (1) and (4) are replaced by steps (1) and (3) of Box 2.10 respectively.

Higher-order interactions

The discussion from Section 2.2.4 regarding higher-order interactions is also applicable in the case of a continuous trait. In summary, investigation pairs and triplets is already a big enough challenge and anyway looking for higher-order interactions is even from a conceptual point of view doubtless. Trying to construct networks is a better strategy.

Customized analysis

Users are offered a panel of options to customize their analysis, similar to those proposed in the case of a binary trait. However, in the case of a continuous trait there is another option that has no equivalent in the binary world. The *-rt* option consists in performing either a rank transformation of the trait, or a rank transformation to normality. These pre-processing steps can increase the power in particular situations, as was shown in [81].

2.4 Trait expressed on a survival scale

In survival analysis, subjects are followed over a certain time period and the aim is to model factors that influence the time to an event to occur, for instance death. Ordinary least squares (OLS) regression methods fall short because the time to event is usually not normally distributed. Furthermore, the model cannot handle censoring without modification. Censoring can be seen as a form of missingness. Typically, some subjects did not show up at the hospital anymore for a reason independent from the event of interest. Such observations are called right censored and are very common in survival data. A classical survival dataset contains two trait columns: one is the time of follow-up and the other is the event status, which records if the event of interest occurred or not. In the case of such a binary variable response, the residuals from an OLS linear probability model violate the homoskedasticity (variance homogeneity) and normality of errors assumptions of OLS regression, resulting in invalid standard errors and hypothesis tests. For a more thorough discussion of this problematic see [105].

Time can be expressed on any measurement scale (days, months, years,...), as long as the same scale is used for all subjects. One can then estimate two functions that are dependent on time, the survival and hazard functions. The first gives, for every time, the probability of surviving (or not experiencing the event) up to that time [63]:

$$S(t) = P(\mathcal{T} > t) \tag{2.24}$$

where t is a time of interest and $\mathcal{T}$ a random variable denoting the time of the event, for instance death. Usually, at the start of the study, the patients did not experience the event yet, so that $S(0) = 1$. This function has of course to be decreasing, i.e. $S(a) \leq S(b)\ \forall a \geq b$. In the context of death, this equation expresses the fact that survival at a later age is only possible if an individual survived earlier days. Given this property, we can define the lifetime distribution function

$$F = P(\mathcal{T} \leq t) = 1 - S(t) \tag{2.25}$$

and event density, i.e. rate of failure events per unit of time

$$f = F'(t) \tag{2.26}$$

The hazard function λ, gives the potential that the event will occur, per time unit, given that an individual has survived up to the specified time [63]:

$$\lambda(t) = \lim_{dt \Rightarrow 0} \frac{P(t \leq \mathcal{T} < t + dt)}{dt S(t)} = \frac{f(t)}{S(t)} = \frac{-S'(t)}{S(t)} \tag{2.27}$$

This function, also called instantaneous rate of death, is positive definite but it does neither have to be increasing, nor to be decreasing and can even have discontinuities. It can sometimes be convenient to work with the cumulative hazard function Λ defined by

$$\Lambda(t) = -\log S(t) \tag{2.28}$$

The name of this function comes from the following relationship

$$\Lambda(t) = \int_0^t \lambda(u) du \tag{2.29}$$

which can be poetically described as the accumulation of hazard over time.

Our aim is to compute, for every pair $[SNP_{lj}, SNP_{rj}]$, a number representing the strength of its association with the survival trait (consisting of a time and a censoring variable). This enables to sort the pairs by decreasing chances of regulating the disease. The variable N_1 (N_2) defines the number of possible values for SNP_{lj} (SNP_{rj}). Categorical environment variables can also be handled.

2.4.1 Without correction for main effects

Conceptually, the starting point is similar to what was done for a continuous trait in Figure 2.7. The subjects are split into groups depending on their genotypes (nine groups in the bi-allelic case) and the risk of each group is assessed ("H", "L" or "O"), as illustrated in Figure 2.9. Note that the word "risk" is used loosely in this section, hazard ratios are commonly used when presenting results in clinical trials involving survival traits and are strictly speaking not the same as relative risk ratios.

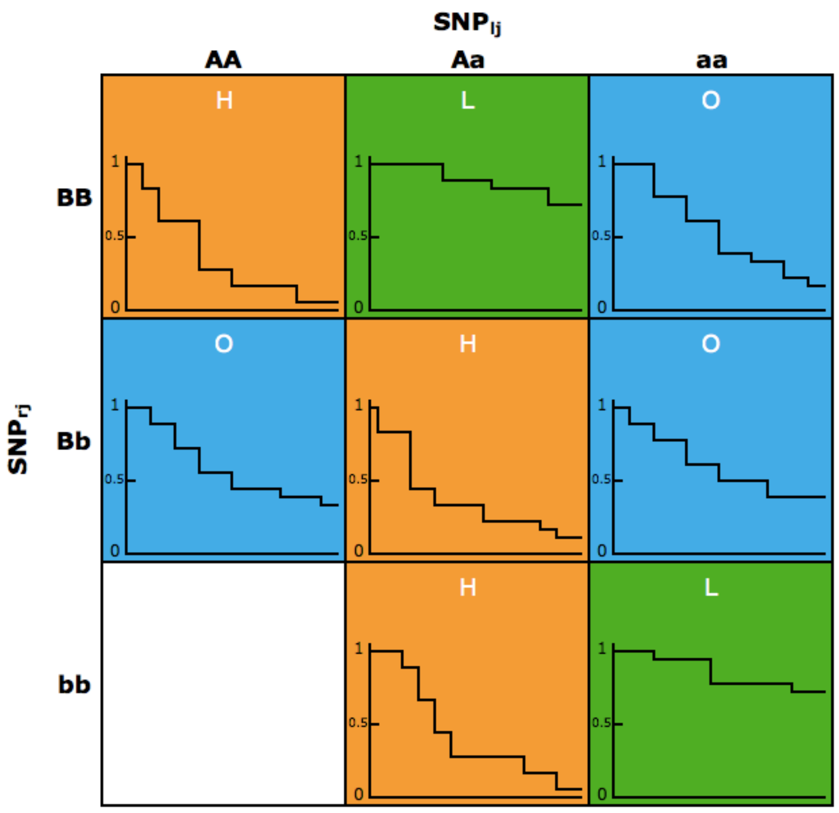


Figure 2.9: The subjects are still split into cells depending on their genotypes, in the same way as in Figure 2.7. However, here the cells contains an estimation of the probability of survival over time. In each cell, the x-axis represents time and the y-axis the probability of not experiencing the event. A cell is immediately assigned to the "O" category (no-evidence for any risk change) if it consists of less than ten individuals. Otherwise, a logrank test is performed to decide if there is a statistically significant difference between the group of subjects belonging to the cell and the group consisting of all other subjects. If there is, the cell is assigned to either the "H" or "L" category, depending on which group has the fastest decreasing curve. Otherwise, it is assigned to the "O" category.

To assign a category to a multilocus genotype combination cell, the subjects are split into two groups: those belonging to the cell of interest and all the other subjects. The null hypothesis is that the risk of event is the same in the two groups. There are several tests to test this hypothesis. Log-rank and Wilcoxon are two nonparametric ones, in that they do not make assumptions about the distributions of survival estimates. In the absence of censorship (e.g. loss to follow up, alive at end of study) the methods reduce to a Mann-Whitney (two sample Wilcoxon) test for two groups of survival times and a Kruskal-Wallis test for more than two groups of survival times. This directly links to the nonparametric tests implemented for the continuous case, as shown in [79]. Peto's log-rank test is generally the most appropriate method for censored outcome, hence our choice [94]. Log-rank is more sensitive than the Wilcoxon test to differences between groups in later points in time. But the Prentice modified Wilcoxon test is more sensitive when the ratio of hazards is higher at early survival times than at late ones [59]. This is important when insensible results are obtained with an MBMDR run for survival data without correction for main effects. Perhaps the assumption that the ratio of hazards is not higher at early survival times than at late ones may be violated.

The log-rank test statistic compares estimates of the hazard functions of the two groups at each observed event time . It is constructed by computing the observed and expected number of events in one of the groups at each observed event time and then adding these to obtain an overall summary across all time points where there is an event [17]. More precisely, the following procedure is used: find out the different observed event times from either group and sort them. For each event time $j = 1, \ldots, J$, let N_{1j} and N_{2j} be the number of subjects in group 1 and 2 respectively, which did not experience the event yet or are censored, at the start of the period j. Let O_{1j} and O_{2j} be the observed events in group 1 and 2 respectively, occurring precisely at time j. Compute $N_j = N_{1j} + N_{2j}$ and $O_j = O_{1j} + O_{2j}$. The *logrank* statistic compares each O_{1j} to its expectation E_{1j} under the null hypothesis that the two groups have identical survival and hazard functions:

$$Z = \frac{\sum_{j=1}^{J}(O_{1j} - E_{1j})}{\sqrt{\sum_{j=1}^{J} V_j}} \tag{2.30}$$

where E_{1j} and V_j are computed by taking into account the fact that under the null, O_j events are supposed to happen across both groups at time j, so that O_{1j} follows an hypergeometric distribution with parameters N_j, N_{1j} and O_j. Therefore, the aforementioned expectation and variance are given by

$$E_{1j} = \frac{O_j}{N_j} N_{1j} \tag{2.31}$$

and

$$V_j = \frac{O_j \frac{N_{1j}}{N_j}(1 - \frac{N_{1j}}{N_j})(N_j - O_j)}{N_j - 1} \tag{2.32}$$

From a programming point of view, some care needs to be taken to reach a fast implementation. First, in order to avoid the computation of the square root from equation 2.30, Z^2 is computed instead of Z. Second, when the data is read at the start of the program, the subjects are sorted once for all in increasing event times. Third, the matrices represented in Figure 2.9 should be stored as vectors, the group index being obtained in the same way as for binary and continuous traits. Figure 2.10 sketches the computation of a statistic t_j^2 representing the strength of the association between the pair $[SNP_{lj}, SNP_{rj}]$ and the trait $\mathcal{T}$.

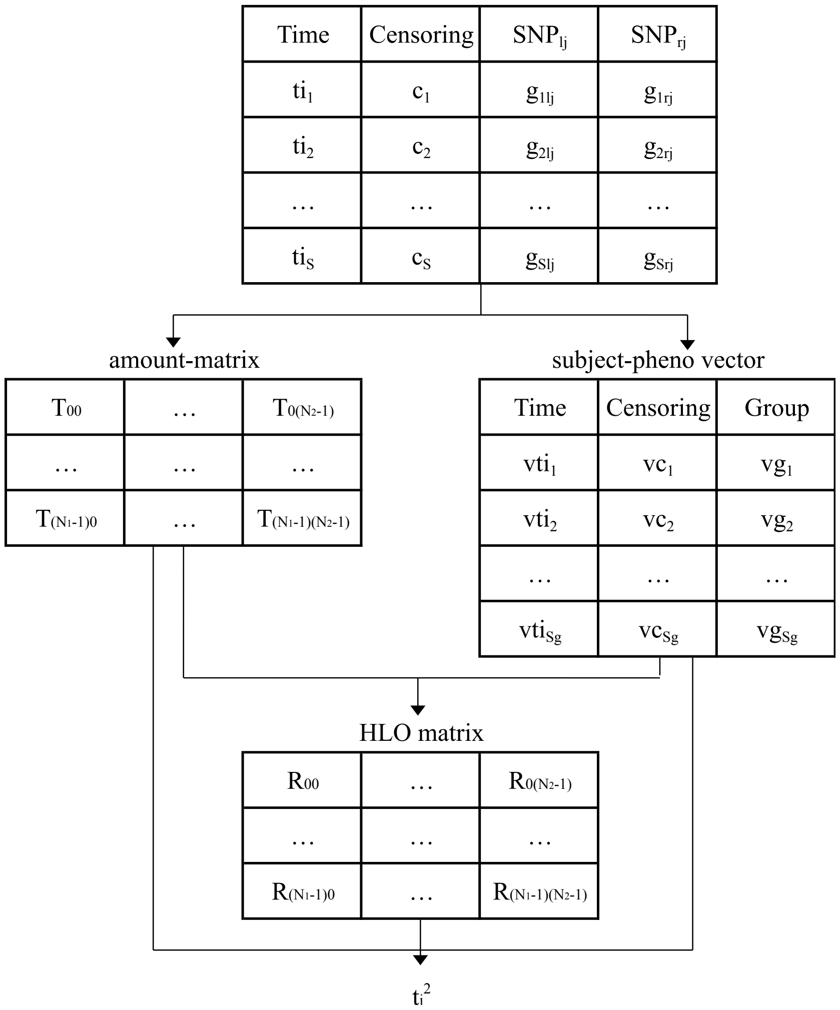


Figure 2.10: Computation of an MB-MDR statistic for a survival trait, without correction for main effects. Input: ti_s is the event time of subject s, c_s its censoring and g_{slj} and g_{srj} its genotypes for SNP_{lj} and SNP_{rj} respectively. The computation can be decomposed in three steps. First, the amount-matrix and subject-pheno vector are constructed. T_{mn} is the number of subjects whose genotype is $g_{slj} = m$ and $g_{srj} = n$. subject-pheno is a vector sorted by increasing event times, containing the time, censoring and group index $g_{slj} * N_2 + g_{srj}$ of the subjects. Second, the HLO matrix is constructed. R_{mn} is either “H” if the subjects whose genotype is m for SNP_{lj} and n for SNP_{rj} have a high statistical risk of disease, “L” if they have a low statistical risk and “O” if there is no statistical evidence. Third, the final t_j^2 value is computed from the objects constructed at steps 1 and 2.

The three steps of the computation of the number t_j^2 capturing the degree of association between the pair $[SNP_{lj}, SNP_{rj}]$ and the trait $\mathcal{T}$ are as follows:

1) Generation of the amount-matrix T (coded as vector for efficiency reasons) and subject-pheno vector V. The former is created with a size of $N_1 \times N_2$ and initialized with zero values, the latter is created empty. Since the subjects are sorted in increasing event times, we can assume that $\forall a < b$ we have $ti_a \leq ti_b$. This allows to construct the two objects in linear time. Perform a loop over the subjects of the dataset: for $s = 1, \ldots S$, if no value is missing for subject s, increment $T_{g_{slj} \times N_2 + g_{srj}}$ by one and add an object containing ti_s, c_s and $g_{slj} * N_2 + g_{srj}$ at the back of vector V. At the end of this process, the size of the latter is equal to the amount S_g of subjects without missing values and the vector is sorted, i.e. $\forall a < b$ we have $vti_a \leq vti_b$. Note that empty groups are not removed from T, because doing so would require an update of all the group indexes stored in vector V.

2) Generation of the HLO-matrix from the two objects generated at step 1, but again coded as vector. The value of each R_h elements depends on a logrank test between the group of subjects having a genotype satisfying $(SNP_{lj} \times N_2 + SNP_{rj} = h)$ and the group composed of all other subjects. As already mentioned, the actual value to compute is the square of equation 2.30. The algorithm is based on a loop, computing the two sums in the numerator and denominator of this equation, using variables called num_Z and den_Z respectively, initialized to zero. Variable N_{1j} is initialized to T_h and variable N_j to S_g. If N_{1j} is below a threshold that is a parameter of the program (default value 10) then Z^2 is not computed at all, since the logrank test would not be statistically significant. In this case the value of R_h is automatically set to “O”. Otherwise, for $s = 1, \ldots S_g$, find out the amount k of subjects having the exact same event time as subject s and the group they belong to. Compute the O_{1j} and O_j values. If $O_j \geq 1$, update num_Z and den_Z as defined in equation 2.30. Before starting the next iteration[8], update N_{1j} and N_j. When the final computed Z^2_{obs} value is below the critical value[9] (based on a liberal significance threshold of 0.1) $Z^2_{\text{crit}} = 2,705543$, the value of R_h is set to “O”, to indicate that we cannot reject the independence hypothesis. Otherwise, R_h is set to either “H” if $num_Z > 0$, to indicate that the population whose genotype satisfies $(SNP_{lj} \times N_2 + SNP_{rj} = h)$ has a high risk of having the trait, or to “L” to indicate a low risk for this event.

3) Computation of t_j from the three objects generated at steps 1 and 2. It consists in performing two logrank tests and returning the maximum of both. The first one compares the subjects belonging to the “H” category versus those belonging to the “L” or “O” category. The second one compares the subjects belonging to the “L” category versus those belonging to the “H” or “O” category. Computing this can be easily achieved in a similar way as in step 2. Note that in many case, there are only cells associated to the “O” category and t_j^2 is readily equal to zero.

The detailed algorithm computing the t_j^2 statistic is reported in Box 2.11.

[8] Note that when $k \neq 0$, $k + 1$ subjects are handled at once and the next iteration is at $s + 1 + k$.

[9] 90th percentile of the χ^2 distribution with 1 degree of freedom

Box 2.11. Survival MB-MDR statistic computing (no 1st order correction)

```
(1) Create a vector $T$ of size $dim = N_1 \times N_2$ and vector $V$ empty. Fill $T$ with 0's. Define $Z^2_{crit} = 2,705543$.
(2) For $s = 1, \ldots, S$: if $g_{slj}$ and $g_{srj}$ are not missing do $T_{g_{slj} \times N_2 + g_{srj}}{++}$ and add an object with values $(ti_s, c_s, g_{slj} * N_2 + g_{srj})$ at the back of $V$. Let $S_g$ be the final size of vector $V$. For facility, we note $vti_k$, $vc_k$ and $vg_k$ respectively the time, censoring and group index of the kth element of $V$.
(3) Create HLO vector $R$ of size $dim$. For $h = 0, \ldots, dim - 1$:
    (a) If ($T_h < 10$) set $R_h$ = "O" and skip steps (b), (c), and (d).
    (b) Define $n_1 = T_h$, $n_t = S_g$ and $num_Z = den_Z = k = 0$.
    (c) While $k < S_g$ do
        i. Define $t_{\text{now}} = vti_k$, $o_1 = o_t = m_1 = 0$ and $ik = k$.
        ii. While ($k < S_g$ and $vti_k = t_{\text{now}}$) do
            A. if ($vc_k$ is not censored) do $\{o_t{++}$ ; if ($vg_k = h$) do $o_1{++}\}$
            B. if ($vg_k = h$) do $m_1{++}$
            C. k++
        iii. Execute $num_Z \mathrel{+}= o_1 - o_t\frac{n_1}{n_t}$ and $den_Z \mathrel{+}= \frac{n_1}{n_t}(1 - \frac{n_1}{n_t})\frac{n_t - o_t}{n_t - 1}$
        iv. Compute $n_1 = n_1 - m_1$ and $n_t = n_t - (k - ik)$.
    (d) If ($den_z \leq 0$) set $R_{mn}$ = "O". Else compute $Z^2_{obs} = \frac{num_Z^2}{den_Z}$.
    (e) If $Z^2_{obs} < Z^2_{crit}$ set $R_{mn}$ = "O", else if $num_Z > 0$ set $R_{mn}$ = "H", else set $R_{mn}$ = "L".
(4) If there is no "H" and no "L" in the R matrix, return $t_j^2 = 0$.
(5) If there is at least one "H":
    (a) Initialize $n_1 = 0$, $n_t = S_g$, $num_Z = den_Z = k = 0$.
    (b) For $h = 0, \ldots, dim - 1$: if $R_h$ = "H" do $n_1 \mathrel{+}= T_h$.
    (c) While $k < S_g$ do
        i. Define $t_{\text{now}} = vti_k$, $o_1 = o_2 = m_1 = m_2 = 0$.
        ii. While ($k < S_g$ and $vti_k = t_{\text{now}}$) do
            A. If ($R_{vg_k}$ = "H") do $\{m_1{++}$ ; if ($vc_k$ is not censored) do $o_1{++}\}$,
               else do $\{m_2{++}$ ; if ($vc_k$ is not censored) do $o_2{++}\}$.
            B. Execute k++.
        iii. Compute $o_t = o_1 + o_2$
        iv. Execute $num_Z \mathrel{+}= o_1 - o_t\frac{n_1}{n_t}$ and $den_Z \mathrel{+}= \frac{n_1}{n_t}(1 - \frac{n_1}{n_t})\frac{n_t - o_t}{n_t - 1}$
        v. Compute $n_1 = n_1 - m_1$ and $n_t = n_t - (m_1 + m_2)$.
    (d) If ($den_Z > 0$) compute $Z^2_{HvsLO} = \frac{num_Z^2}{den_Z}$, else set it equal to zero.
```

```
(6) If there is at least one "L":
    (a) Initialize $n_1 = 0$, $n_t = S_g$, $num_Z = den_Z = k = 0$.
    (b) For $h = 0, \ldots, dim - 1$: if $R_h$ = "L" do $n_1{+}{=}T_h$.
    (c) While $k < S_g$ do
        i. Define $t_{\text{now}} = vti_k$, $o_1 = o_2 = m_1 = m_2 = 0$.
        ii. While ($k < S_g$ and $vti_k = t_{\text{now}}$) do
            A. If ($R_{vg_k}$ = "L") do $\{m_1{+}{+}$ ; if ($vc_k$ is not censored) do $o_1{+}{+}\}$,
               else do $\{m_2{+}{+}$ ; if ($vc_k$ is not censored) do $o_2{+}{+}\}$.
            B. Execute k++.
        iii. Compute $o_t = o_1 + o_2$.
        iv. Execute $num_Z \mathrel{+}= o_1 - o_t\frac{n_1}{n_t}$ and $den_Z \mathrel{+}= \frac{n_1}{n_t}(1 - \frac{n_1}{n_t})\frac{n_t - o_t}{n_t - 1}$
        v. Compute $n_1 = n_1 - m_1$ and $n_t = n_t - (m_1 + m_2)$.
    (d) If ($den_Z > 0$) compute $Z^2_{LvsHO} = \frac{num^2_Z}{den_Z}$, else set it equal to zero.
(7) Return $t^2_j = \max(Z^2_{\text{HvsLO}}, Z^2_{\text{LvsHO}})$.
```

2.4.2 With correction for main effects

For survival data, the correction for main effects is based on the Cox proportional hazards model [23]. The latter links the time that passes before the event to covariates, in our case the two SNPs and/or environmental variables. The assumption behind the Cox proportional hazards model is that the effect of a covariate is multiplicative with respect to the hazard rate. For instance, smoking may double the hazard rate for a lung cancer to occur.

More precisely, the model can be seen as consisting of two parts: the hazard function $\lambda_0(t)$ and the effect parameters. The former defines how the risk of event changes over time (per time unit) at baseline levels of covariates and the latter defines the hazard variation in response to explanatory covariates [28]. In our context, the covariates are discrete ones, i.e. they are coded $0, 1, \ldots, N_1 - 1$ and $0, 1, \ldots, N_2 - 1$ respectively. Covariates expressed on a continuous scale are handled in Chapter 4, but note already that in such a case it is usually assumed that the hazards responds logarithmically. In this document, we present the Cox model in the classical way: first, by ignoring the issue of ties (i.e. several subjects experiencing an event at the exact same time) and second, by showing how to adapt the equations to take it into account.

Let Y_i be the observed time (censoring time or event time) and C_i the right censoring value (1 if the event occurred and 0 if it is censored) for subject i. The hazard function at time t for covariates SNP_{lj} and SNP_{rj} is given by

$$\lambda_0(t|SNP_{lj}, SNP_{rj}) = \lambda_0(t)e^{\beta_1 SNP_{lj} + \beta_2 SNP_{rj}} \tag{2.33}$$

where β_1 and β_2 are the parameters to fit.

Let β be a vector containing the latter two values and X the covariate matrix (rows correspond to individuals, the first column contains their values for SNP_{lj} and the second for SNP_{rj}). A partial likelihood can be constructed by taking into account all possible t times[23] and is given by

$$L(\beta) = \prod_{i:C_i=1} \frac{\theta_i}{\sum_{j:Y_j \geq Y_i} \theta_j} \tag{2.34}$$

where $\theta_j = e^{X_j \beta}$. The corresponding log partial likelihood is given by

$$l(\beta) = \sum_{i:C_i=1} (X_i \beta' - log \sum_{j:Y_j \geq Y_i} \theta_j) \tag{2.35}$$

The model parameters are then best estimated by maximizing equation 2.35 over β. This maximization can be obtained using the Newton-Raphson algorithm [19]. The score function is as usual the derivative of the log partial likelihood [9]:

$$l'(\beta) = \sum_{i:C_i=1} (X_i - \frac{\sum_{j:Y_j \geq Y_i} \theta_j X_j}{\sum_{j:Y_j \geq Y_i} \theta_j}) \tag{2.36}$$

For the regression coefficients, approximate standard errors are needed. These can be produced from the variance-covariance matrix. The latter can be approximated by the inverse of the Hessian matrix. The Hessian matrix itself can as usual be obtained by computing the second derivative of the log partial likelihood:

$$l''(\beta) = -\sum_{i:C_i=1} (\frac{\sum_{j:Y_j \geq Y_i} \theta_j X_j X_j'}{\sum_{j:Y_j \geq Y_i} \theta_j} - \frac{\sum_{j:Y_j \geq Y_i} \theta_j X_j \times \sum_{j:Y_j \geq Y_i} \theta_j X_j'}{[\sum_{j:Y_j \geq Y_i} \theta_j]^2}) \tag{2.37}$$

This procedure works well in practice, but is know to lead to sub-optimal results in the case of ties, where Efron's approach should be preferred [28]. Let t_j represent the unique times and H_j the set of indexes i such that $Y_i = t_j$ and $C_i = 1$. Let m_j be the number of events at time j. Efron's method consists in maximizing the following partial likelihood:

$$L(\beta) = \prod_j \frac{\prod_{i \in H_j} \theta_i}{\prod_{l=0}^{m-1} (\sum_{i:Y_i \geq t_j} \theta_i - \frac{l}{m} \sum_{i \in H_j} \theta_i)} \tag{2.38}$$

The corresponding log partial likelihood can be computed as

$$l(\beta) = \sum_j (\sum_{i \in H_j} X_i \beta' - \sum_{l=0}^{m-1} log(\sum_{i:Y_i \geq t_j} \theta_i - \frac{l}{m} \sum_{i \in H_j} \theta_i)) \tag{2.39}$$

This leads to a score function used for the Newton-Raphson method given by

$$l'(\beta) = \sum_j (\sum_{i \in H_j} X_i - \sum_{l=0}^{m-1} \frac{\sum_{i:Y_i \geq t_j} \theta_i X_i - \frac{l}{m} \sum_{i \in H_j} \theta_i X_i}{\sum_{i:Y_i \geq t_j} \theta_i - \frac{l}{m} \sum_{i \in H_j} \theta_i}) \tag{2.40}$$

and a Hessian matrix defined by

$$l''(\beta) = \sum_j \sum_{l=0} m - 1(\frac{\sum_{i:Y_i \geq t_j} \theta_i X_i X_i' - \frac{l}{m} \sum_{i \in H_j} \theta_i X_i X_i'}{\phi_{j,l,m}} - \frac{Z_{j,l,m} Z_{j,l,m}'}{\phi_{j,l,m}^2}) \tag{2.41}$$

where

$$\phi_{j,l,m} = \sum_{i:Y_i \geq t_j} \theta_i - \frac{l}{m} \sum_{i \in H_j} \theta_i \tag{2.42}$$

$$Z_{j,l,m} = \sum_{i:Y_i \geq t_j} \theta_i X_i - \frac{l}{m} \sum_{i \in H_j} \theta_i X_i \tag{2.43}$$

In this thesis, speed of the software is the major issue. For this reason, most of the source code has been written from scratch, in order to be able to customize every single operation and win orders of magnitude of precious computing time. However, the Cox model is by far more complex to implement than all the methods that have been presented so far in this chapter. For this reason, I was advocated by my promotor to avoid reinventing the wheel here and try instead to take advantage of the fact that the (open) source code of the R function *coxfit6* is in fact written in C. This function is well established, has been highly tested by many users and is close to what we need.

As already mentioned earlier, calling an R function from mbmdr-4.4.1.out would lead to an important slow down of the software and force the user to install R on his machine. This would imply that mbmdr-4.4.1.out would not be a standalone software anymore, requiring all users to install R, even those who are not interested in a survival data analysis. Furthermore, since different versions of R uses different internal survival routines, the user having R already installed, may have to install an older/newer version than the one that they are using. To avoid this issue, our solution is to take the open source C code of the *coxfit6* function, remove all the parts that are related to *R* (for instance conversions to S-PLUS format) and keep this purified code as a customized library in our software. In this way, we avoid calling R from C++, which would imply data conversions in C++ to call R, which itself would convert it back to call its own source code written in C. Obviously, this would lead to an important waste of computing time (and in practice also sometimes to conversion problems).

Note that the work consisting in making a standalone library from the C source code of an R function from the survival package is far from trivial. Indeed, the C code is usually relying on some R functions, whose source code is also written in C. The latter code is itself relying on further R functions, actually implemented in C and so on. Fortunately, this cascade does end pretty fast in the case of *coxfit6* and only five C source code of R functions had to be customized! At the end of this procedure, we end up with a *coxfit6* function that can be called within our C++ software:

$$coxfit6(time, censoring, covar) \Rightarrow [\beta, l(\beta)]$$

This function takes as arguments the time and censoring status of the subjects (sorted in increasing time order) and the covariate matrix (containing one line per individual and one column per covariate). It returns the aforementioned β vector and the log likelihood $l(\beta)$.

Partial likelihood ratio test

To make use of the fitted model, simple logrank tests can no longer be used as in steps (3), (5) and (6) of Box 2.11. In each of these three steps, the idea was to compare a selection of subjects S_1 versus the others S_0. At step (3), the subjects with one particular genotype versus the rest. At step (5) (respectively 6), the subjects belonging to the "H" (respectively "L") category versus the rest. These comparisons should now take account of the main effects and of the group belongings simultaneously. To achieve this, the model is extended by adding a column at the back of matrix X, whose value is 1 if the subject belongs to S_1 and 0 otherwise.

A partial likelihood ratio test enables to test if the precision of the model gets statistically significantly better by adding the extra column or not. The log partial likelihood is computed by *coxfit6* in both scenarios. To avoid confusion, we note $l(\beta)_{main}$ the one without the extra column (taking only into account the main effects) and $l(\beta)_{main+}$ the one with the group-belongings column. The test-statistic is defined by

$$D = 2(l(\beta)_{main+} - l(\beta)_{main}) \tag{2.44}$$

This test statistic follows a χ^2_1 distribution. Significance is assessed by comparing the final computed D_{obs} value to the critical value[10] (based on a liberal significance threshold of 0.1) $D_{\text{crit}} = 2,705543$. When a significant group is found during the HLO matrix construction, it is assigned to the "H" category when the β value corresponding to the extra column is positive and to "L" otherwise.

Algorithm

This section describes the algorithm computing a number t_j, capturing the degree of association between the pair $[SNP_{lj}, SNP_{rj}]$ and the trait $\mathcal{T}$, adjusted for the main effects of the two SNPs (or environmental variables), using either a codominant or an additive coding scheme. The variables N_1 and N_2 still define the number of possible values for SNP_{lj} and SNP_{rj} respectively. Categorical environment variables can again be handled, as long as they are coded $0, 1, \ldots, N_1 - 1$, resp. $0, 1, \ldots, N_2 - 1$. The computation follows again the general strategy of Figure 2.10, except that the subject-pheno vector no longer stores the group indexes and is here split into two different vectors V_t and V_c (storing respectively the time and censoring values of the individuals). The three steps are now as follows:

1) Generation of the amount-matrix T (coded as usual as a vector for efficiency reasons), time vector V_t, censoring vector V_c and covariate matrix X. The vector T is created with a size of $N_1 \times N_2$ and initialized with 0's and the other objects are created empty. Matrix X is composed of two columns in case of an additive coding scheme and $N_1 + N_2 - 2$ columns in case of a codominant one. All objects are build up by performing a loop over the subjects. For $s = 1, \ldots S$: if no value is missing for subject s: increment the cell of T whose index is given by $g_{slj} \times N_2 + g_{srj}$, add ti_s at the back of V_t, add c_s at the back of V_c and insert a new row at the end of X. In the case of additive coding, g_{slj} is put on the first column

[10] 90th percentile of the χ^2 distribution with 1 degree of freedom

of the new row and g_{srj} in the second one. Indeed, in the additive case, the coding is defined by the number of minor alleles, which is readily the code that has been used in this software to code the genotypes. In the case of codominant coding, $N_1 - 1$ columns are used to code g_{slj} and $N_2 - 1$ to code g_{srj}. The first column is equal to 1 if $g_{slj} = 1$ and 0 otherwise, the second column is equal to 1 if $g_{slj} = 2$ and 0 otherwise, etc. In other words, the first column indicates if the subject has a single minor allele for the SNP_{lj} or not, the second column if the subject has two minor alleles for SNP_{lj} or not and so on. The first column coming after the columns that have already been filled is equal to 1 if $g_{srj} = 1$ and 0 otherwise, the second one is equal to 1 if $g_{srj} = 2$ and 0 otherwise, etc. The codominant coding obviously allows *coxfit6* to build a more precise model, since more variable are available to fit it.

2) The second step still consists in generating the HLO-matrix, but again coded in vector format. First, the model taking into account only the main effects is fitted using *coxfit6* and the corresponding log likelihood $l(\beta)_{main}$ is stored. Then, a column containing only 0's is added to X and a loop is performed over the different genotype groups. For $h = 1, \ldots, dim - 1$, if the h^{th} element of the amount-matrix is less than 10, set the h^{th} element of the HLO-vector to "O", to indicate the absence of evidence that the subset of individuals with multilocus genotype satisfying $SNP_{lj} \times N_2 + SNP_{rj} = h$ has neither a high nor a low risk for disease. Otherwise, overwrite the last column of X with 0's, except for rows corresponding to subjects belonging to the group under investigation (i.e. those whose genotype satisfies $g_{slj} \times N_2 + g_{srj} = h$) which are set to 1. Then fit the new model with *coxfit6*, obtain the corresponding β vector and log likelihood $l(\beta)_{main+}$ value and compute $D_{obs} = 2(l(\beta)_{main+} - l(\beta)_{main})$. When this value is below the critical value $D_{\text{crit}} = 2,705543$, set the h^{th} element of the HLO-vector to "O". Otherwise, set it to either "H" if the last element of the β vector is greater than zero, to indicate a high risk for the disease, or to "L" to indicate a low risk for the disease.

3) If there are neither "H" nor "L" values in the HLO vector, return 0. Otherwise, compute the following D_{HvsLO} and D_{LvsHO} values and return the maximum. First, overwrite the last column of X with 0's, except for the subjects belonging to a genotype group whose corresponding HLO-vector cell is equal to "H", which are set to 1. Then fit the new model with *coxfit6*, obtain the corresponding β vector and log likelihood $l(\beta)_{main+}$ value and compute $D_{HvsLO} = 2(l(\beta)_{main+} - l(\beta)_{main})$. Second, do exactly the same except that the elements set to 1 are now those corresponding to a "L" value, to obtain $D_{LvsHO} = 2(l(\beta)_{main+} - l(\beta)_{main})$. Of course, if there are no "H" ("L") values, the first (second) number is not computed and the function readily returns the second (first) one.

The detailed algorithm computing the t_j statistic representing the strength of the association between the pair $[SNP_{lj}, SNP_{rj}]$ and the trait, corrected for the main effects of the SNPs, is reported in Box 2.12.

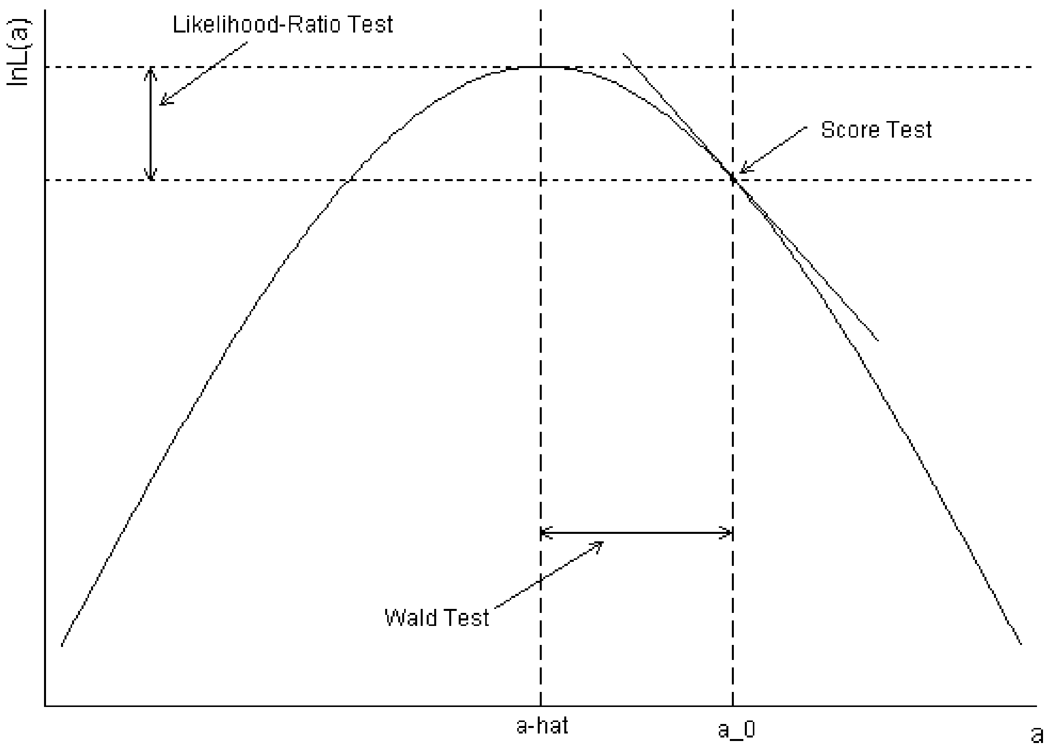
Box 2.12. Survival MB-MDR statistic computing (with 1st order correction)

```
(1) Create a vector $T$ of size $dim = N_1 \times N_2$ and fill it with 0's. Create empty
    vectors $V_t$ and $V_c$. Create empty matrix $X$ consisting of 2 columns in case of
    additive coding and $N_1 + N_2 - 2$ columns in case of codominant coding.
(2) For $s = 1, \dots, S$: if $g_{slj}$ and $g_{srj}$ are not missing do $T_{g_{slj} \times N_2 + g_{srj}}{++}$, add $ti_s$
    at the back of $V_t$, $c_s$ at the back of $V_c$ and insert a new row at the end of $X$:
    (a) If additive coding: put $g_{slj}$ in the $1^{st}$ column and $g_{slj}$ in the $2^{nd}$.
    (b) If codominant coding: perform two loops. For $a = 1, \dots, N_1 - 1$:
        if (a $= g_{slj}$) put 1 in the $a^{th}$ column, otherwise 0. For $b = 1, \dots, N_2 - 1$:
        if (b $= g_{srj}$) put 1 to the $(N_1 + b)^{th}$ column, otherwise 0.
(3) Call $coxfit6(V_t, V_c, X)$. Let $loglik_{main}$ be the returned log likelihood.
(4) Insert a new column at the end of $X$ filled with 0's. Define $D_{\text{crit}} = 2,705543$.
(5) Create HLO vector $R$ of size $dim$. For $h = 0, \dots, dim - 1$:
    (a) If ($T_h < 10$) set $R_h =$"O" and skips step (b), (c), and (d).
    (b) Initialize $a = 0$. Overwrite the last column of $X$. For $s = 1, \dots, S$:
        if $g_{slj}$ and $g_{srj}$ are not missing: do (if ($g_{slj} \times N_2 + g_{srj} = h$) overwrite
        the $a^{th}$ element with 1, else with 0) and $a{++}$.
    (c) Call $coxfit6(V_t, V_c, X)$. Let $\beta$ and $loglik_{main+}$ be the returned objects.
    (d) Compute $D_{\text{obs}} = 2(loglik_{main+} - loglik_{main})$.
    (e) If ($D_{\text{obs}} < D_{\text{crit}}$) set $R_{mn} =$ "O", else if the last element of the $\beta$ vector
        is greater than zero set $R_{mn} =$ "H", else set $R_{mn} =$ "L".
(6) If there is no "H" and no "L" in the R matrix, return $t_j = 0$.
(7) If there is at least one "H":
    (a) Initialize $a = 0$. Overwrite the last column of $X$. For $s = 1, \dots, S$:
        if $R_{g_{slj} \times N_2 + g_{srj}} =$ "H") overwrite the $a^{th}$ element with 1, else with 0) and
        $a{++}$.
    (b) Call $coxfit6(V_t, V_c, X)$. Let $loglik_{HvsLO}$ be the returned log likelihood.
    (c) Compute $D_{HvsLO} = 2(loglik_{HvsLO} - loglik_{main})$.
(8) If there is at least one "L":
    (a) Initialize $a = 0$. Overwrite the last column of $X$. For $s = 1, \dots, S$:
        if $R_{g_{slj} \times N_2 + g_{srj}} =$ "L") overwrite the $a^{th}$ element with 1, else with 0) and
        $a{++}$.
    (b) Call $coxfit6(V_t, V_c, X)$. Let $loglik_{LvsHO}$ be the returned log likelihood.
    (c) Compute $D_{LvsHO} = 2(loglik_{LvsHO} - loglik_{main})$.
(9) Return $t_j = \max(D_{\text{HvsLO}}, D_{\text{LvsHO}})$.
```

Most of the options for customizing the analysis that were proposed in the case of a trait expressed on a binary scale and on a continuous scale are also available for survival studies. An option that is specific for survival trait allows users to specify a different input format as the one used in this thesis. Indeed, many datasets contain the time value first and then the censoring one, but this convention is not always followed. The option *-if ISALIVE* allows users to specify that the input format contains the censoring variable first (coded as 0 if the time is censored and 1 otherwise) and the time value just after that. Adapting the code introduced in the previous two sections for main effects analysis or three-order interactions can be done in a similar way as for a continuous trait. It is just a degeneration/generalization of the existing algorithms.

2.5 Discussion

In this chapter, we first discussed the reasons that lead us to select a different type of statistic to correct for the main effects of the SNPs in Section 2.2.2 than the one selected in Section 2.3.2. Indeed, score statistics were computed in the former case (binary traits) and Wald statistics in the latter one (continuous traits). The score and Wald tests are asymptotically equivalent tests of the likelihood ratio (lr) one, i.e. as the sample size become infinitely large, the values of all these test-statistics becomes increasingly close to each other. "In finite samples, the three will tend to generate somewhat different test statistics, but will generally come to the same conclusion" [54]. In fact, the three tests address the same research question: does constraining a parameter to a particular value (zero in our case, i.e. leaving out this predictor variable) reduce the fit of the model? The difference between the tests is that they use different strategies to answer this question, as illustrated in Figure 2.11.



Source: http://www.ats.ucla.edu/stat/mult_pkg/faq/general/nested_tests.htm

Figure 2.11: In this Figure, the x-axis contains possible values of a parameter a and the y-axis the log likelihood corresponding to these values. The lr test compares the log likelihoods of a model for which a has a fixed value (zero in the case of MB-MDR) to a model where it is freely estimated. Graphically, this corresponds to comparing the height of the likelihoods for the two models. The Wald test compares the value $a - hat$ leading to the highest log likelihood to the value a_0 under the null (in MB-MDR $a = 0$). Graphically, the distance between $a - hat$ and a_0 indicates if freely estimating a significantly improves model fit or not. The score test is based on the slope of the log likelihood at a_0. Graphically, this corresponds to looking at how quickly the likelihood is changing at the null hypothesized value.

The advantage of the Wald and score tests over the lr one, is that they require only one model to be estimated and not two as for the lr test. An interesting relationship exists between the three tests when the model is linear: Wald $\geq$ lr $\geq$ score [57]. For a typical simple random sample, the Pearson chi-squared statistic is widely used and can be shown to equal the score test statistic for testing independence in an $J \times K$ contingency table with row and column variables that are jointly multinomial [75]. It is also equal to the score test statistic for no row (column) covariate effect in a multinomial logistic regression where the column (row) variable is considered the outcome. This motivates the choice of a chi-squared test in MB-MDR for binary traits (most often used). Since we cannot necessarily rely on asymptotic equivalence properties as we may have sparse pooled cells, having a test that is stable enough for smaller samples is advocated. Note that we are also using a score test when a correction is made for predictors other than the main effects of the SNPs, as shown in Chapter 4. Here, the main motivation is to obtain more stability in parameter estimations, when a large number of predictor variables are included in the analysis, for instance gender, age, body mass index (BMI), smoking habits, etc.

For continuous traits, we preferred the Wald statistic over the score one. Indeed, the Wald test can be used to test multiple parameters simultaneously. It is hence commonly used to perform multiple degrees of freedom tests to model categorical predictor variables in regression. As a consequence, it is more flexible than the score test in this sense. The advantage of the score test is that it can be used to search for omitted variables when the number of candidate variables is large. But we do not expect to have such large numbers of candidate variables, as our focus is on the interaction between two categorical variables only. When there are issues with small samples in our 9 cells or in two groups to be compared within MB-MDR, possibly jeopardizing the large sample distribution assumption of the Wald test statistic, then we have shown that we can always turn to non-parametric tests [79].

Another interesting subject discussed in this chapter is that even if MB-MDR is a non-parametric method (no assumptions are made about genetic modes of inheritance), this does not mean that the internal tests used for labeling the 9 cells have to be non-parametric too. Recall that a non-parametric test of statistical significance does by definition not make assumptions about the nature of the underlying distributions from which samples have been drawn, as opposed to a parametric test. In fact, for binary traits the default in `mbmdr-4.4.1.out` is a non-parametric test and for continuous traits a parametric one. For binary traits, we have already discussed above the benefits of choosing the non-parametric chi-square test.

For continuous traits, the test is a parametric student's t-test, assuming normality of the mean and variances of the trait values. This condition is usually met, leading to better results than with a non-parametric method, as it is well known that parametric methods have improved statistical power over non-parametric ones when all parametric assumptions are valid [35]. However, when the normality assumption is suspected not to be met, we recommend to rank-transform traits to normality prior MB-MDR analysis [79]. Note that even when non-parametric methods are selected, some issues may need to be checked. For instance, the Cox model used in Section 2.4.2 is non-parametric to the extent that no assumptions are made about form of the baseline hazard. However, two key assumptions need to be checked.

First and foremost, the issue of non-informative censoring. To satisfy this assumption, the design of the underlying study must ensure that the mechanisms giving rise to censoring of individual subjects are not related to the probability of an event occurring. For example, in clinical studies, care must be taken to ensure that continuation of follow-up does not depend on the participant's medical conditions. Violation of this assumption can invalidate just about any sort of survival analysis.

The second key assumption in the Cox model is that of proportional hazards. In a regression type setting this means that the survival curves for two strata (determined by the particular choices of values for the x-variables) must have hazard functions that are proportional over time (i.e. constant relative hazard). In that situation, and also for the Cox model, there are tests that can be applied to test proportionality.

The next discussion concerns an idea that has been brushed in Section 2.2.1. We have shown how to compute a statistic t_j representing the strength of the association between the pair $[SNP_{lj}, SNP_{rj}]$ and a trait $\mathcal{T}$ expressed on a binary scale, but have indicated that information theory proposes a more direct and natural way to discover the strength of this association, i.e. by computing their mutual information [18] defined by

$$t_j = \mathcal{I}(SNP_{lj}, SNP_{rj}; \mathcal{T}) \tag{2.45}$$

An obvious asset of this choice, is that it is easy and fast to compute. It can be obtained in different ways and a particularly intuitive one is

$$\mathcal{I}(SNP_{lj}, SNP_{rj}; \mathcal{T}) = H(SNP_{lj}, SNP_{rj}) + H(\mathcal{T}) - H(SNP_{lj}, SNP_{rj}, \mathcal{T}) \tag{2.46}$$

Indeed, adding the entropy of the pair of SNPs to the entropy of the trait leads to counting the intersection between these quantities twice, as illustrated in Figure 2.12. Removing the overall entropy (the entropy of the pair of SNPs and the trait) leaves us thus just with exactly what we are searching for: the intersection counted once.

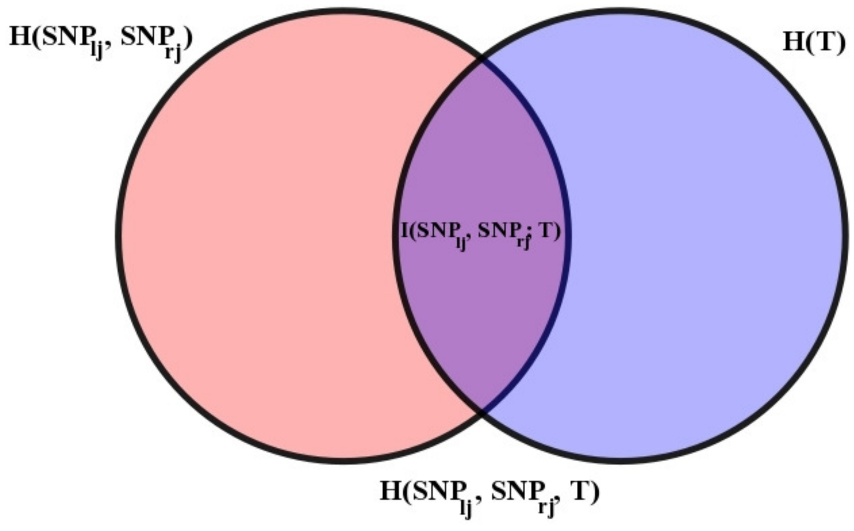


Figure 2.12: The mutual information between the pair $[SNP_{lj}, SNP_{rj}]$ and the trait T can be seen as the intersection between their entropies.

A nice property of the definition given in equation 2.46 is that it is "optimal" from an information theory point of view. Indeed, the principle of data processing inequality shows that if we take any function f to transform the information provided by the SNPs, we cannot gain information on the trait [22], i.e. that we have

$$\forall f: \ \mathcal{I}(SNP_{lj}, SNP_{rj}; \mathcal{T}) \geq \mathcal{I}(f(SNP_{lj}, SNP_{rj}); \mathcal{T}) \tag{2.47}$$

This approach was successfully applied in AMBIANCE and SYMPHONY, two information-theoretic method for gene–gene and gene–environment interaction analysis of disease syndromes and complex phenotypes [18, 64]. However, these methods are dedicated to binary traits. In general, adapting such methods to other type of traits is far from trivial and many such software focuses on one type of trait only, as opposed to mbmdr-4.4.1.out.

An alternative to the mutual information, is to work with the multiple mutual information defined by

$$\begin{aligned}\mathcal{I}(SNP_{lj}; SNP_{rj}; \mathcal{T}) = H(SNP_{lj}) + H(SNP_{rj}) + H(\mathcal{T}) - H(SNP_{lj}, SNP_{rj}) \\ -H(SNP_{lj}, \mathcal{T}) - H(SNP_{rj}, \mathcal{T}) + H(SNP_{lj}, SNP_{rj}, \mathcal{T})\end{aligned} \tag{2.48}$$

Graphically, this corresponds to the surface at the intersection of the three entropies of SNP_{lj}, SNP_{rj} and $\mathcal{T}$, as could easily be shown by sketching a figure similar to Figure 2.12. This measure is similar to the bivariate synergy introduced in [129]:

$$Syn(SNP_{lj}, SNP_{rj}; \mathcal{T}) = I(SNP_{lj}, SNP_{rj}; \mathcal{T}) - I(SNP_{lj}, \mathcal{T}) - I(SNP_{rj}, \mathcal{T}) \tag{2.49}$$

In fact, straightforward manipulations shows that the multiple mutual information equals to the bivariate synergy apart from a sign change. This approach has been investigated in [11] and describes the synergy as "the additional contribution provided by the *whole* compared with contributions provided by the *constituents*". Furthermore, they propose to normalize the metric by dividing it by $H(\mathcal{T})$, such that its values are between -1 and 1. This metric can either be used as a measure of direct association between SNP pairs and the trait, or as a measure of association between the trait and multilocus genotype combinations pooled in two groups. In both cases, the caveats is again that an implementation dedicated to binary traits requires more work to adapt to other trait types. Note that it should also be possible to define conditional synergies, i.e. joint explications of the pairs of SNPs over their singular effects, in the presence of another variable.

We have seen that information-theoretic software usually comes at the cost of some flexibility loss. Therefore, in this work, we have decided to propose methods based on the MB-MDR methodology, to compute a number capturing the degree of association between a SNP pair and the trait. These methods depends on the type of trait and the fact that the user opts to correct for main effects or not. In the latter case, codominant correction should be the default choice, because on average this leads to the best practical results, in terms of balance between power and type I error. However, not correcting for main effects is the fastest method and a relative comparison of its speed compared to the other two approaches (additive and codominant corrections) should be done. Indeed, a dataset solved within a day without correction for main effects, may require weeks when the slow down represents one or several orders of magnitude.

When the trait is either expressed on a binary or a continuous scale, all the code has been written from scratch and every single operation optimized to avoid doing the same computations twice. The matrix operations have been kept to a minimum. As a consequence, the slow down is acceptable. In the case of a binary trait, the computing time observed when no correction is performed is approximately multiplied by three when an additive correction is selected and by four in case of a codominant one. In the case of a continuous trait, the computing time obtained without correction for main effects is about multiplied by two and a half when an additive correction is performed and by three and a half in case of a codominant one. For this reason, codominant correction is the default option in the software for these type of traits.

In the case of a survival datasets, the main effects correction relies on the C source code of the *coxfit6* function from R. In Box 2.12, this function is called several times (in the bi-allelic case for instance, 12 times) and some computations are the same every time! Indeed, only the last column of the X matrix changes from one call to another. This can unfortunately not be avoided as long as this function is used as a library, leading to a slow down of about one order of magnitude. Trying to call R functions directly from the software (as I was advocated to do many times in this thesis) would even amplify this problem, since in addition to the aforementioned issue, unnecessary data conversions would take place. Another difference between the survival case and the binary and continuous cases is the size of the X matrix. In the case of a trait expressed on a binary or continuous scale, this matrix is very small (in the bi-allelic case for instance, it consists of up to 9 rows). However, in the case of a survival datasets, it consists of as many rows as subjects, i.e. usually hundreds or even thousands of rows. As a consequence, the model fitting is at least one order of magnitude slower than before. As an example, a very small dataset composed of 400 subjects and 20 SNPs is solved within 9 seconds by mbmdr-4.4.1.out when no correction for main effects is performed, but takes 9 minutes 25 seconds in case of additive correction and 32 minutes 21 seconds in case of codominant correction. The fact that the latter is significantly slower than additive correction is not a surprise. Indeed, in the bi-allelic case, the X matrix consists of two columns when additive correction is performed and four when codominant correction is chosen. Since fitting the model involves a lot of matrix multiplications, the computing time is therefore approximately multiplied by four. For this reason, not correcting for main effects is the default choice in the case of a survival dataset, to avoid that users starts very long analysis without realizing it.

In summary, the fact that almost all the C++ code of mbmdr-4.4.1.out has been written from scratch and optimized was worth it. Without this work, reaching GWAIs would have been impossible in practice. This demonstrate that writing a dedicated software for the task was a necessary work. A simple R program relying on general R functions would have been orders of magnitude slower and could therefore not be used to analyze large-scale datasets. Our approach allows to solve three-order interaction problems, which has been shown to be useful in practice in the context of pulmonary tuberculosis [21]. Note that the biology of complex diseases is often very complex and the odds are high that higher-order interactions are taking place. However, these higher-order interactions can be seen as a big network to be discovered, but impossible to find directly by testing for all possible combination of nodes. For this reason, finding edges between two nodes or link between three nodes can already be a good starting point. That is a foreseen usage of our software, actually.

Chapter 3

Multiple-Testing correction

3.1 Outline

In Chapter 2, we have shown different methods for computing a statistic t_j representing the strength of the association between a pair $[SNP_{lj}, SNP_{rj}]$ and the trait $\mathcal{T}$. This enables to sort the m pairs of SNPs by decreasing t_j values. However, these numbers are difficult to interpret. Is the highest value much higher than what would be expected by chance? If yes, how much values are significantly higher than what would be obtained fortuitously? The purpose of this chapter is to make it possible to answer such questions in a proper mathematical way. Ideally, the pairs of SNPs discovered by our software would then be validated in independent samples (gold standard in genetic association studies) and/or by a biologist in the lab as illustrated in Figure 3.1 and the pharmaceutical industry would produce drugs targeting the identified SNPs.

Source: http://www.ouestfrance-emploi.com/metiers/biologiste-medical

Figure 3.1: The aim of mbmdr-4.4.1.out is to discover pairs of SNPs having high statistical chances to be linked to the disease. Such findings could be investigated by a biologist. Being able to find out which pairs are significant and which not in an efficient way is the main issue handled in this PhD thesis. In practice, minimizing the work of the biologist in the lab is essential, since his task is expensive and time consuming. Furthermore, biologists would soon lose interest in a software that tends to suggest wrong hypothesis too often.

A first attempt to find out if a pair of SNPs is significantly associated with the disease or not is to make a simple hypothesis test. The null hypothesis H_0 is that the pair is not associated to the trait. The goal is to compute the probability of obtaining an MB-MDR statistic equal to or more extreme than the observed t_j value, under the null. Such a probability is called a *p-value*. The latter can then be compared to a liberal significance threshold α, usually 5%. If below, the null hypothesis can be rejected, i.e. intuitively one can say that finding such a high t_j value would be very surprising if the null hypothesis would be true. Otherwise, the null hypothesis cannot be rejected and in practice the corresponding pair of SNPs is not kept in the set of interesting pairs to be investigated further. Note that this procedure is not a mathematical demonstration, one cannot prove that the null hypothesis is true or false in this way, just get a strong indication. Two types of errors can be made in such tests, as illustrated in Table 3.1. First, a false positive, or type I error, occurs when a pair of SNPs is declared to be linked to the disease, when it is not in reality. Second, a false negative, or type II error, occurs when the hypothesis test fails to identify a pair of SNPs that is in fact truly linked to the disease.

Table 3.1: Type I and type II errors in GWAIs

	H_0 not rejected	H_0 rejected
Pair of SNPs not linked to the disease		TYPE I
Pair of SNPs truly linked to the disease	TYPE II	

The problem of the naive approach presented so far is that it does not take into account the fact that a lot of such tests are performed simultaneously. Let's note $p_1, ..., p_m$ the p-values corresponding to the m hypothesis tests performed, $\mathcal{M}_0$ the unknown subset of true null hypotheses and m_0 the unknown size of this subset (in practice, most of the investigated pairs of SNPs of a GWAIs are not linked to the disease and m_0 is close to m). Intuitively, since every test from $\mathcal{M}_0$ has a 5% chance to lead to a rejection and since $m_0 \approx m$, about 5% of all performed tests lead to a rejection wrongly. Since our software is meant to investigate billions of pairs of SNPs, we cannot afford making a mistake for 5% of them! This is called the multiple-testing problem and is the core business of this PhD thesis. A lot of papers have been dedicated to this subject, but GWAIs is a challenging area of application for multiple testing procedures because of the huge amount of computations involved.

Most solutions to the multiple-testing problem from the literature aim either at controlling the *family-wise error rate* (FWER) or the *false-discovery rate* (FDR). The former is defined as the probability of at least one type I error across the m hypothesis tests, the latter as the expected proportion of type I errors among the rejected null hypotheses. It can easily be shown that FDR $\leq$ FWER. The latter is thus the most difficult to keep low and as a consequence the most interesting one regarding our aim to minimize the amount of false-positives. In this thesis, our goal is therefore to control the FWER at 5%. It is important to note that in order to control the FWER exactly, it would be necessary to know $\mathcal{M}_0$. Since this cannot be the case, we introduce the notions of *weak* and *strong* control. The former controls the FWER under the complete null hypothesis, i.e. the scenario where all null hypothesis are true ($m_0 = m$). The latter controls the FWER for every possible choice of $\mathcal{M}_0$. Obviously, strong control implies exact and weak control.

3.2 Bonferroni correction

An easy way to reach strong control of the FWER is to use the Bonferroni correction [26]. The idea is to divide the significance threshold of all performed tests by the amount of tests, i.e to use $\frac{\alpha}{m}$ as threshold. It is straightforward to prove that the Bonferroni correction guarantees a FWER below $\alpha = 5\%$ using Boole's inequality [36]:

$$FWER = P(\cup_{j \in \mathcal{M}_0}(p_j \leq \frac{\alpha}{m})) \leq \sum_{j \in \mathcal{M}_0}(P(p_j \leq \frac{\alpha}{m})) \leq m_0\frac{\alpha}{m} \leq m\frac{\alpha}{m} = \alpha \qquad (3.1)$$

Implementing the Bonferroni correction requires to estimate the $p_1, \ldots, p_m$ values. Indeed, computing the exact value would require to know the exact distribution of the $t_1, \ldots, t_m$ statistics under the null, i.e. under the assumption that the observed trait values have been assigned randomly to the S subjects. Finding out the exact distribution would imply to generate the $S!$ possible permutations of the subject's trait values, which is in practice not feasible since typical datasets involves hundreds or thousands of patients. As a consequence, the p_j values are estimated based on the observed data and a random subset of $B = 999$ permutations (default value), as illustrated in Figure 3.2 in the particular case of a trait expressed on a binary scale.

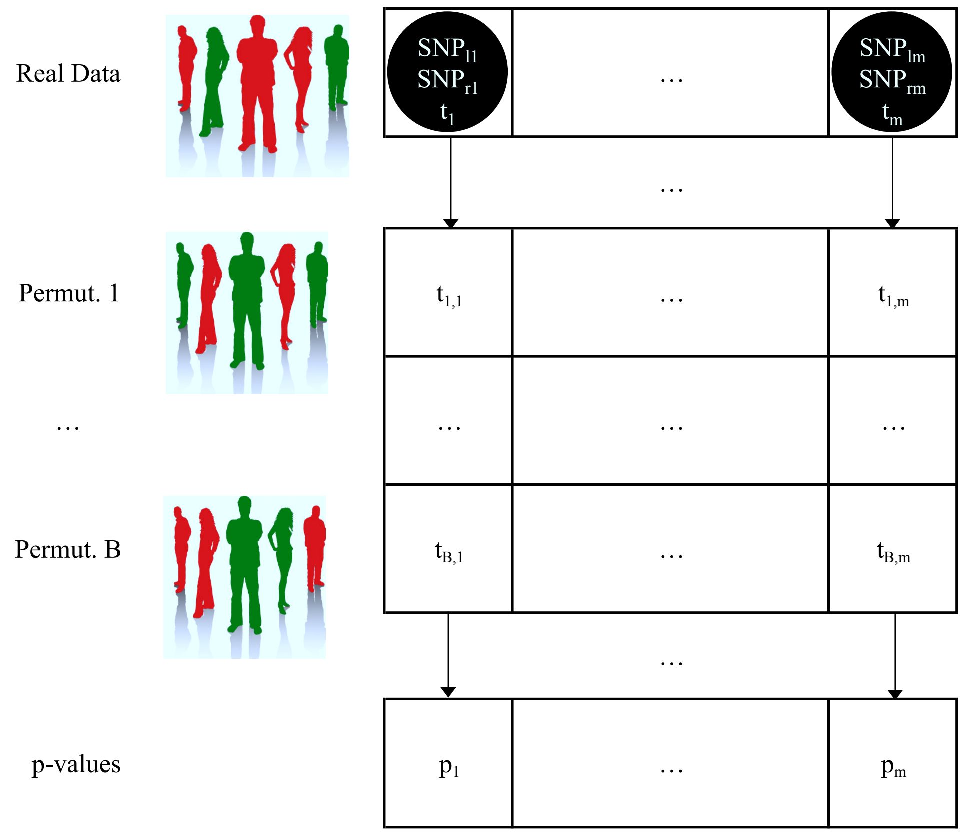


Figure 3.2: To estimate the p_j values ($j = 1, \ldots, m$), B random permutations of the trait values of the subjects are generated. Each time, the original t_j value is compared to the $t_{i,j}$ values obtained on the permuted data ($i = 1, \ldots, B$) and p_j is estimated by $\frac{a_j+1}{B+1}$, where a_j is the amount of $t_{i,j} \geq t_j$ values. The null hypothesis is rejected for the SNP pairs for which $p_j \leq \frac{\alpha}{m}$.

3.2.1 Classical implementation

A straightforward way to implement the Bonferroni correction algorithm can be decomposed into the following three steps:

1) Generation of the *Real Data* vector R. This vector is created empty with a preallocated size of m and filled by performing a loop over the pairs of SNPs. For $j = 1, \dots, m$: compute the test-statistic t_j corresponding to the j^{th} pair of SNPs[1] and store an object containing the computed test-statistic and the names of the corresponding SNPs in R_j.

2) Generation of the test-statistics *Permutation* vectors $T_1, \dots, T_B$. These vectors are created empty with a preallocated size of m and filled one by one. For $i = 1, \dots, B$: permute randomly the trait values of the subjects, without modifying their genotypes and perform a nested loop over the pairs of SNPs. For $j = 1, \dots, m$: compute the test-statistic $t_{i,j}$ corresponding to the j^{th} pair of SNPs and store it at the j^{th} index of the T_i vector.

3) Generation of the *p-values* vector P. This vector is again created empty with a preallocated size of m and filled by performing a loop over the pairs of SNPs. For $j = 1, \dots, m$: compute the amount a_j of $t_{i,j} \geq t_j$ values $(i = 1, \dots, B)$ and store the estimated p-value $\frac{a_j+1}{B+1}$ at the j^{th} index of vector P. For $j = 1, \dots, m$: if $P_j \leq \frac{\alpha}{m}$, reject the null hypothesis for the j^{th} pair of SNPs.

Three remarks are in place:

First, the $+1$ terms at the numerator and denominator of the computation of the estimated p-value are introduced to ensure a strictly positive result. Indeed, since a p-value represents the probability to obtain a test-statistic value at least as extreme as the observed one, it cannot be equal to zero: the observed value itself has been witnessed and has thus a non-zero probability of occurrence[2]. As a consequence, the minimal p-value that can be obtained is $\frac{1}{B+1}$. Hence the choice of a default value of $B = 999$, to obtain p-values that are multiples of $\frac{1}{1000}$.

Second, the method that was proposed so far is not optimal from a memory point of view, since it implies to store all *Permutation* vectors of Figure 3.2 in memory. This requires $O(Bm)$ memory, whereas $O(m)$ can be achieved by adopting a different approach. The idea is that the a_j values computed at step 3, can already be calculated on-the-fly. A vector *a-values* is created from scratch and initialized with 1's. Step 1 does not change, still requiring $O(m)$ memory. Then, at each iteration of step 2 $(i = 1, \dots, B)$, the trait values are still permuted randomly and a loop is performed over the pairs of SNPs. However, this loop does not store test-statistics in a vector anymore, but readily updates the *a-values* vector by incrementing the a_j values corresponding to the $t_{i,j} \geq t_j$ by one. Finally, steps 3 becomes trivial: dividing the *a-values* vector by $B + 1$ readily leads to the final *p-values* vector.

[1] The computation of such a test-statistic has been described in full details in Chapter 2 and depends for instance on the scale on which the trait is expressed, the type of correction for main effects, ...

[2] The classical way to present Bonferroni is to add a vector *Permutation 0* coinciding with the *Real Data* vector to the set of *Permutation* vectors. The number of such vectors is now $B + 1$ and the p-value is estimated by $\frac{a_j}{B+1}$, where a_j is the amount of $t_{i,j} \geq t_j$ values including the artificial permutation coinciding with the observed data $(i = 0, \dots, B)$, ensuring that $a_j \geq 1$. The version described above is slightly more efficient, avoiding a comparison whose outcome is already known.

Third, note that the trait values can be permuted in-place at step 2, without harming the algorithm. Indeed, after step 1, the observed trait values are no longer of any use and loosing them has no drawback. Furthermore, the *Permutation* vectors of Figure 3.2 are filled row by row. Therefore, once a vector is filled, the trait values are again not of any use anymore. As a consequence, $O(S)$ memory is enough to handle the permutations of the trait values (recall that S is the number of subjects).

3.2.2 mbmdr-4.4.1.out's implementation

Performing a GWAIs with the classical implementation of Bonferroni generates about 10^{12} p-values. Producing a file containing so much results is pointless, no one can read them all. mbmdr-4.4.1.out's implementation (see Box 3.1) proposes to produce only the $n = 1000$ (default) lowest p-values, sorted in increasing order. To achieve this, the main idea of the *Sorting by insertion* algorithm [65] can be recycled. First, n p-values are computed and stored (with the names of the corresponding SNPs) in a vector P. After that, P is sorted in increasing p-values order using the *quicksort* algorithm [65]. Then, at each iteration, the next p-value is computed and compared with the highest p-value in P. If greater or equal, nothing is done. Otherwise, the highest value is removed and the current one inserted (with the names of the corresponding SNPs) at the right place in order to preserve the sorting. This insertion requires $\frac{n}{2}$ operations on average. This method works particularly well when $m >> n$. Intuitively, the probability of having to insert decreases at each iteration and tends to zero because the vector contains lower and lower p-values. This algorithm requires $O(m)$ computing time on average, but could degenerate in $O(nm)$. A benefit of this approach is that vector P requires only a memory of $O(n)$ instead of $O(m)$, dividing the memory usage by about 10^9 in the case of a GWAIs! A drawback is that the $t_{i,j}$ values from Figure 3.2 are now computed column by column, making in-place permutation of the trait values impossible, requiring a memory of $O(BS)$ instead of $O(S)$, i.e. 10^3 times more.

Box 3.1. mbmdr-4.4.1.out's implementation of Bonferroni

```
(1) Generate $B$ permutations of the trait values and store them in memory.
(2) If $m < n$ set $n = m$. Create empty vector $P$ of size $n$. For $j = 1, \dots, n$:
    (a) Set the trait value to the original ones and compute $t_j$.
    (b) Define $a_j = 1$. For $i = 1, \dots, B$: set the trait values to the ones of the
        $i^{\text{th}}$ permutation, compute $t_{i,j}$ and if ($t_{i,j} \geq t_j$) do $a_j$++.
    (c) Store an object ($a_j$, $SNP_{lj}$, $SNP_{rj}$) in $P_j$. We note $(P_j)_1$ the 1st element.
(3) Sort $P$ by increasing $a_j$ values. After that, for $j = n + 1, \dots, m$ do:
    (a) Set the trait values to the original trait ones and compute $t_j$.
    (b) Define $a_j = 1$. For $i = 1, \dots, B$: set the trait values to the ones of the
        $i^{\text{th}}$ permutation, compute $t_{i,j}$ and if ($t_{i,j} \geq t_j$) do $a_j$++.
    (c) If $a_j < (P_n)_1$ do {$k = n$ ; while ($k > 0$ and $a_j < (P_{k-1})_1$) do $P_k = P_{k-1}$
        and k−− ; store an object ($a_j$, $SNP_{lj}$, $SNP_{rj}$) in $P_k$.}
(4) For $j = 1, \dots, n$: divide $(P_j)_1$ by B+1 to obtain the p-value to compare to $\frac{\alpha}{m}$.
```

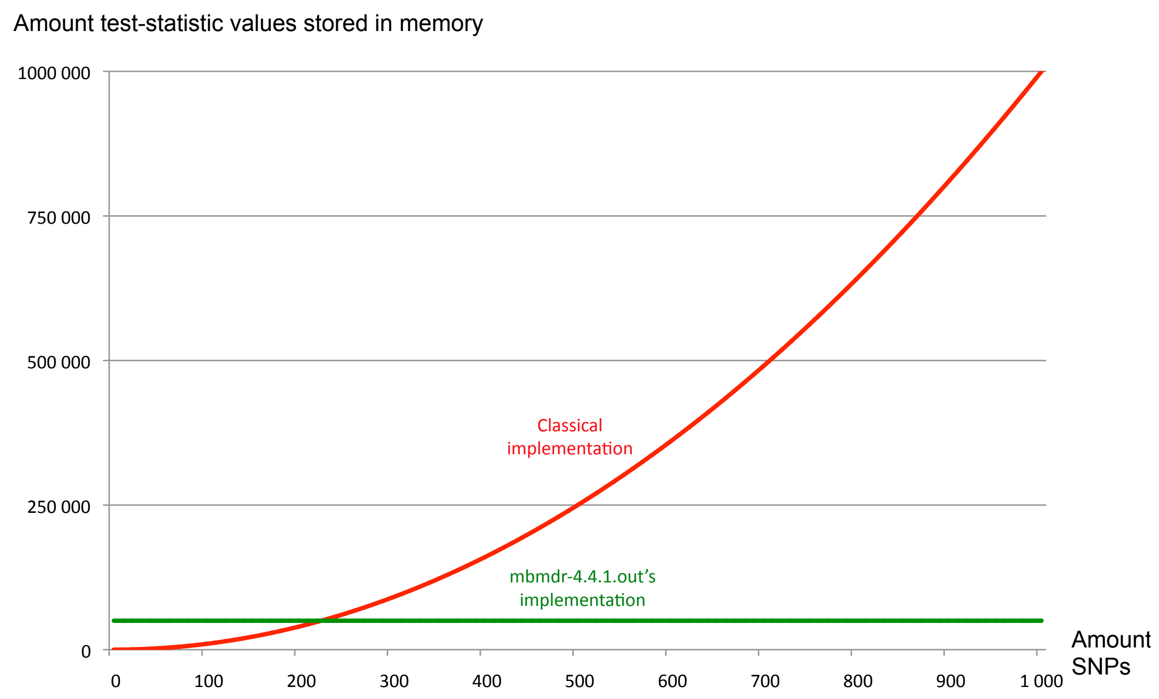


Figure 3.3: Memory usage comparison of classical Bonferroni implementation versus mbmdr-4.4.1.out's one. The memory usage of the former raises quadratically with the number of SNPs, whereas the memory usage of the latter is independent from the number of SNPs.

The fact that so much memory can be saved for a GWAIs is in fact just a consequence of the following observation. The main benefit of mbmdr-4.4.1.out's implementation of Bonferroni is to make the memory usage independent from the amount $M$of SNPs! Indeed, only objects of size n are stored. The classical implementation on the other hand implies to store test-statistics for all $m = \frac{M(M-1)}{2}$ pairs of SNPs. It raises thus quadratically in terms of SNPs as illustrated in Figure 3.3. A GWAIS would require several Terabytes of memory with the classical implementation!

Note that this algorithm is trivial to adapt for users interested in performing a main effects analysis. The implementation from Box 3.1 needs only a minuscule adaptation: at steps (2)(c) and (3)(c), an object (a_j, SNP_j) should now be stored instead of the $(a_j, SNP_{lj}, SNP_{rj})$ one. It comes without saying that the variable m represents the number of SNPs here and not the number of pairs anymore. Similarly, the code of Box 3.1 is easy to adapt for three-order interactions: at step (2)(c) and (3)(c), an object $(a_j, SNP_{lj}, SNP_{mj}, SNP_{rj})$ should be stored instead of the $(a_j, SNP_{lj}, SNP_{rj})$ one and m now represents the number of triplets of SNPs.

The Bonferroni correction is a popular method because of its simplicity. However, this method is know to be very conservative. In other words, it controls the FWER so strongly, that the type II error rate increases. The art in writing multiple-testing correction algorithms is to achieve a good balance between type I and type II errors. The purpose of the methods presented in the next sections, is to achieve a higher power than the Bonferroni correction, while still controlling the FWER at 5%.

3.3 MaxT

The single-step and step-down maxT adjusted p-values are two algorithms introduced by Westfall & Young to handle the multiple-testing problem [136]. Since the former tends to be conservative for the control of the FWER, we focus on the latter in this thesis and call it simply maxT. In the previous section, the $p_1, \ldots, p_m$ values have been computed without taking care of the multiple-testing issue and only after that action was taken, i.e. by adapting the thresholds of the hypothesis tests. In this section the idea is the opposite: the thresholds stay at $\alpha = 5\%$, while the p-values are adjusted for the fact that multiple-testings are performed. To avoid confusion in the rest of this document, we call *marginal empirical p-value* the traditional (unadjusted) p-value associated with a single hypothesis test and *adjusted p-value* the adjusted ones.

In the context of this PhD thesis, I was often in the situation of explaining maxT to a broad audience. I found out that starting with an intuitive explanation works best. Beforehand, note that the maxT algorithm naturally starts by computing test-statistics for all pairs of SNPs and sorting them in decreasing order. Without loss of generality, let's note $t_1 \geq t_2 \geq \ldots \geq t_m$ these sorted test-statistics and $(SNP_{l1}, SNP_{r1}), \ldots, (SNP_{lm}, SNP_{rm})$ the corresponding pairs of SNPs. Intuitively, to estimate the probability of observing a maximum value that is at least as extreme as t_1 under the null, t_1 should not be compared to the statistics $t_{1,1}, \ldots, t_{B,1}$ as was the case for the Bonferroni correction in Figure 3.2. Instead, it should be compared to the test-statistics that are also the highest just by chance, for each permutation $i = 1, \ldots, B$, as in Figure 3.4.

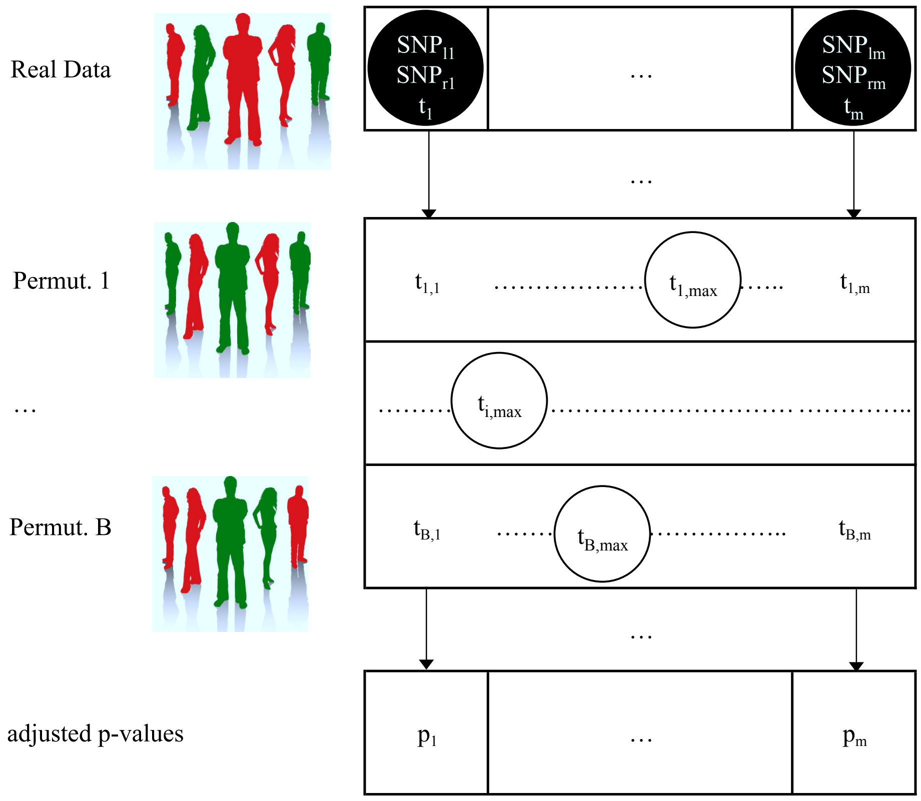


Figure 3.4: The intuitive idea for estimating p_1 in maxT, is that t_1 should be compared to its alter egos, i.e. the maximums $t_{1,max}, \ldots, t_{B,max}$ of each row.

First, suppose that the complete null hypothesis is true, i.e. that no pair of SNPs is linked to the disease. In this scenario, t_1 is the highest value just by chance[3]. Comparing t_1 to its alter egos, i.e. the maximums of each rows in Figure 3.4, comes down to looking where t_1 lies in an approximation of the distribution of the maximums under the null. As a consequence, comparing p_1 to α =5% is sound, since t_1 has obviously a probability of 5% to land by chance in the top 5% of that distribution under the null.

Once p_1 has been computed, an intuitive way to understand how p_2 is calculated is to imagine that since the case of the pair (SNP_{l1}, SNP_{r1}) is closed, it is now removed from the dataset, regardless of the fact that p_1 is below $\alpha = 5\%$ or not. As a consequence, (SNP_{l2}, SNP_{r2}) is here the pair leading to the highest test-statistic and the whole procedure can be repeated to compute p_2. This idea can be extended to successively compute $p_3, \ldots, p_m$. From a programming point of view, implementing it like this would be dramatically inefficient.

To avoid it, the actual implementation of maxT is based on the following three observations. First, note that the $t_{i,m}$ value is guaranteed to be useful: when p_m is computed, (SNP_{lm}, SNP_{rm}) is the only pair remaining in the dataset and $t_{i,m}$ is readily the maximum. Second, note that the value $t_{i,m-1}$ is useful if and only if it is higher than $t_{i,m}$. Indeed, if it is higher it is at least the maximum in the computation of p_{m-1}, but if it is lower it cannot be the maximum in any computation. For this reason, in the latter case, the value of $t_{i,m-1}$ is overwritten by the value of $t_{i,m}$, without harming the algorithm. Third, note that this reasoning can be carried on for $t_{i,m-2}, \ldots, t_{i,1}$. After this procedure, the permutation vectors contains decreasing values. This allows to compute any p_j value easily, by using the $t_{1,j}, \ldots, t_{B,j}$ values readily[4]. By the way, observe that the values on the left of $t_{1,max}, \ldots, t_{B,max}$ are now all equal to the $t_{1,max}, \ldots, t_{B,max}$ values themselves. As a final remark, note that a p_2 value lower than p_1 would be too optimistic, i.e. the p-value of a less significant pair should not be lower than the one of a more significant pair. For this reason, whenever $p_2 < p_1$, maxT replaces p_2 by p_1 and so on for $p_3, \ldots, p_m$. Figure 3.5 illustrates this whole procedure.

So far, we have supposed that the complete null hypothesis is true. In this case, we have shown that the probability that p_1 is less or equal than 5% is actually equal to 5%. Furthermore, we just imposed that $p_1 \leq p_2 \leq \ldots \leq p_m$. This implies that 95% of the time, there is no adjusted p-value below 5%. In other words, we have proved that maxT controls the FWER weakly at 5%. When the complete null hypothesis is not true, a lot of scenario are possible depending on how many pairs of SNPs are truly linked to the disease. Nevertheless, Westfall & Young have proved that in any case, maxT guarantees strong control of the FWER under the subset pivotality hypothesis [136]. By definition, the latter is true if for all subsets $\mathcal{K}$ of $\{1, \ldots, m\}$ the joint distributions of the sub-vecor $\{p_i : i \in \mathcal{K}\}$ are identical under the restrictions $H_{\mathcal{K}} = \cap_{i \in \mathcal{K}} \{H_i = 0\}$ and $H_{\mathcal{M}} = \cap_{i=1}^{m} \{H_i = 0\}$ [39].

[3] To be very accurate, note that the probability of landing on top is not always distributed uniformly among the pair of SNPs. Some may have a particular structure increasing/decreasing this probability. However, the maxT algorithm is constructed to take this fact into account.

[4] They are now guaranteed to be greater or equal than the values on their right in Figure 3.4. Furthermore, we can still imagine that the values on the left have been removed, but we don't actually need to remove them to make the algorithm work.

A lot has been made of this subset pivotality condition, that has been portrayed in the literature as too stringent [103]. However, Westfall & Troendle have shown that this condition is hardly restrictive [135]. Note that a drawback of maxT, is that when the test statistics are not identically distributed unbalanced adjustments can be observed because not all tests contribute equally to the computed adjusted p-values [80]. If this is an issue for a particular experiment, then users can turn to the minP algorithm presented in Section 3.5. This can for instance be very useful when there are a lof of SNPs with a low minor allele frequencies (MAF). However, the drawback is that minP is significantly slower than maxT and more memory demanding.

3.3.1 Classical implementation

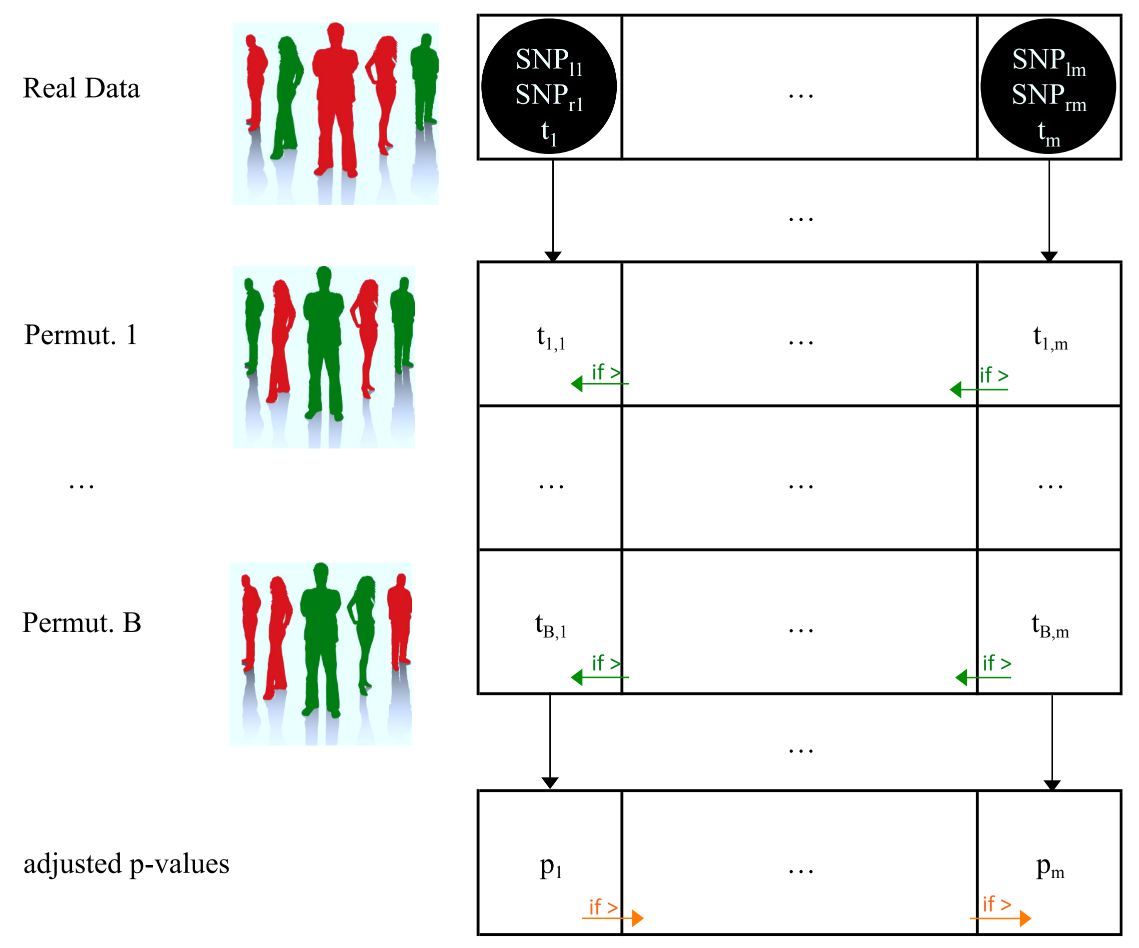


Figure 3.5: Classical maxT implementation. First, test-statistics are computed for all pairs of SNPs and stored with the corresponding SNP names in the sorted *Real data* vector. We can assume that $t_j \geq t_{j+1}, \forall j = 1, \ldots, m-1$. Then, B permutations of the trait values are generated and $t_{i,j}$ values are computed and stored in the *Permutation* vectors, $\forall i = 1, \ldots, B$, $\forall j = 1, \ldots, m$. Then, $t_{i,j}$ is overwritten by $t_{i,j+1}$ whenever $t_{i,j+1} > t_{i,j}, \forall i = 1, \ldots, B, \forall j = m-1, \ldots, 1$. Then, the adjusted p_j values are estimated by $\frac{a_j+1}{B+1}$, where a_j is the amount of $t_{i,j} \geq t_j$ values. Finally, p_{j+1} is overwritten by p_j whenever $p_j > p_{j+1}, \forall j = 1, \ldots, m-1$.

The different steps of the classical maxT algorithm can be decomposed as follows:

1) Generation of the sorted *Real Data* vector R. This vector is created empty with a preallocated size of m and filled by performing a loop over the pairs of SNPs. For $j = 1, \ldots, m$: compute the test-statistic t_j corresponding to the j^{th} pair of SNPs and store an object containing the computed test-statistic and the names of the corresponding SNPs in R_j. After this loop, sort the vector in decreasing test-statistic order using the *quicksort* algorithm. Without loss of generality, let's note $t_1 \geq t_2 \geq \ldots \geq t_m$ these test-statistics and $(SNP_{l1}, SNP_{r1}), \ldots, (SNP_{lm}, SNP_{rm})$ the corresponding pairs of SNPs.

2) Generation of the test-statistics *Permutation* vectors $T_1, \ldots, T_B$. These vectors are created empty with a preallocated size of m and filled one by one. For $i = 1, \ldots, B$: permute randomly the trait values of the subjects, without modifying their genotypes and perform a nested loop over the pairs of SNPs. For $j = 1, \ldots, m$: compute the test-statistic $t_{i,j}$ corresponding to the j^{th} pair of SNPs and store it at the j^{th} index of the T_i vector. Force the monotonicity of the rows: for $j = m-1, \ldots, 1$ replace $T_{i,j}$ by $T_{i,j+1}$ if $T_{i,j} < T_{i,j+1}$.

3) Generation of the *adjusted p-values* vector P. This vector is again created empty with a preallocated size of m and filled by performing a loop over the pairs of SNPs. For $j = 1, \ldots, m$: compute the amount a_j of $T_{i,j} \geq t_j$ values $(i = 1, \ldots, B)$ and store the value $\frac{a_j+1}{B+1}$ at the j^{th} index of vector P. Force the monotonicity of vector P: for $j = 1, \ldots, m-1$ replace P_{j+1} by P_j if $P_{j+1} < P_j$.

Three remarks are in place:

First, the $+1$ terms at the numerator and denominator of the computation of the estimated adjusted p-value are again introduced to ensure a strictly positive result. Indeed, observing a test-statistic at least as extreme as the observed one cannot have a probability of zero since the observed one has been witnessed.

Second, from a memory point of view, it is again best to implement the aforementioned algorithm in a slightly different way. The classical implementation implies all *Permutation* vectors of Figure 3.5 to be in memory at the same time, requiring $O(Bm)$ memory. A memory of $O(m)$ can be achieved by adopting a similar approach as in Section 3.2 (Bonferroni correction). A vector P of all a_j values can be created from scratch and initialized with 1's. In this way, the elements of the P vector can be updated at the end of each iteration $i = 1, \ldots, B$ of step 2 by incrementing the P_j values corresponding to the $T_{i,j} \geq t_j$ by one. As a consequence, all i^{th} $t_{i,j}$ values are no longer of any use at the end of the i^{th} iteration and can be overwritten. Hence, only a single test-statistics *Permutation* vector T whose values are overwritten at each iteration can be used instead of B ones. After that, step 3 comes down to dividing all elements of P by $B+1$ and performing the aforementioned monotonicity enforcing procedure.

Third, note that the trait values can again be permuted in-place at step 2. Indeed, losing the observed trait values after step 1 has no drawback, since they are no longer of any use after that step. Furthermore, once a *Permutation* vector of Figure 3.5 has been filled (in practice de P vector as mentioned above), the corresponding permuted trait values used in the computation are again not of any use anymore and can be lost without harming the algorithm. As a consequence, if S is the number of subjects, $O(S)$ memory is enough to handle the permutations of the trait values.

3.3.2 Van Lishout's implementation

To avoid producing a file with about 10^{12} results when performing a GWAIs, we propose Van Lishout's implementation of maxT. This important contribution has been accepted for oral presentation at the *BBC 2011* conference in Luxembourg [122] and published in *BMC Bioinformatics* in 2013 [125]. The main benefit of this algorithm is to save memory, but significant computing time is also saved. Figure 3.6 illustrates the idea.

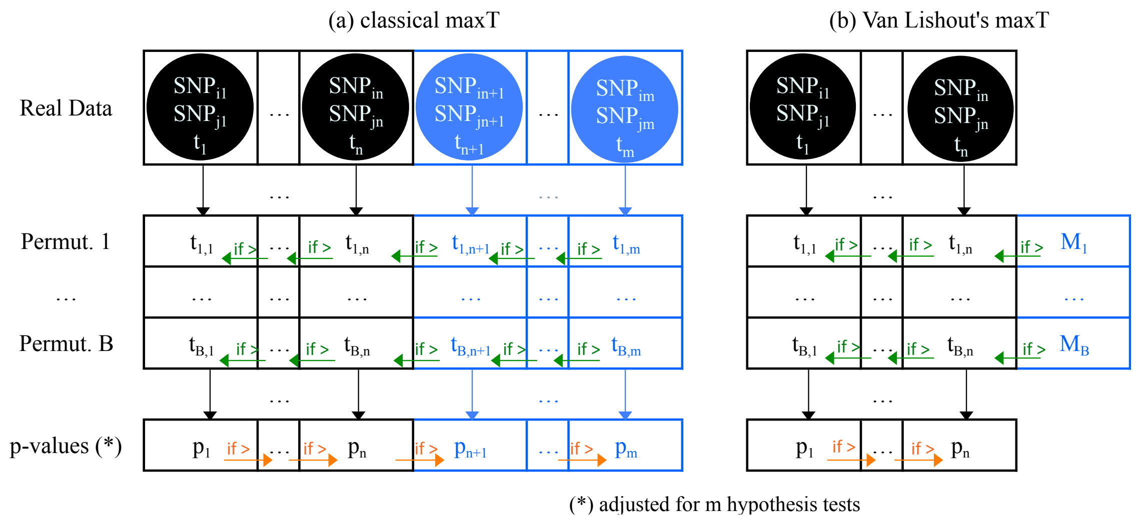


Figure 3.6: Classical implementation of maxT versus Van Lishout's one. In the classical case, test-statistics are computed for all pairs of SNPs and stored with the SNP names in the sorted *Real data* vector ($t_j \geq t_{j+1}, \forall j = 1, \ldots, m-1$). In Van Lishout's implementation, the same computations are performed, but only the top n test-statistics and corresponding SNP names are stored. In the classical implementation, all $t_{i,j}$ values are computed and stored $\forall i = 1 \ldots B$, $\forall j = 1 \ldots m$. In Van Lishout's maxT, all these $t_{i,j}$ values are still computed, but only the $[t_{i,1}, \ldots, t_{i,n}]$ values are stored and the maximum M_i of the $[t_{i,n+1}, \ldots, t_{i,m}]$ ones. In the classical implementation, $t_{i,j}$ is overwritten by $t_{i,j+1}$ whenever $t_{i,j+1} > t_{i,j}$, $\forall j = m-1 \ldots 1$. In Van Lishout's implementation, $t_{i,n}$ is overwritten by M_i if $M_i > t_{i,n}$ and $t_{i,j}$ is overwritten by $t_{i,j+1}$ whenever $t_{i,j+1} > t_{i,j}$, $\forall j = n-1 \ldots 1$. Finally, p_{j+1} is overwritten by p_j whenever $p_j > p_j + 1$, $\forall j = 1 \ldots m-1$ in the classical case and $\forall j = 1 \ldots n-1$ in Van Lishout's implementation. It is easy to prove that the final $p_1, \ldots, p_n$ adjusted p-values are the same in both cases.

The essence of the MB-MDR methodology is to identify sets of gene-gene interactions via a series of association tests, which may or may not be fully non-parametric, while reducing dimensionality. In practical applications, there is an abundance of adjusted p-values close or equal to 1 and only a few adjusted p-values points towards interesting multi-locus genotype combination to pursue. With this in mind, we adapt the maxT algorithm so that it still calculates the test-statistics for all SNP pairs, but only calculates the adjusted p-values of the n best pairs, i.e. the ones with the n lowest p-values. We show that our method produces the exact same top n adjusted p-values as with the original maxT implementation, however using many fewer resources. In other words, despite the fact that only n adjusted p-values are produced, they are still adjusted at the overall level, i.e. for the m association tests.

When interaction signals are expected to be strong in the light of an improved study design (for instance, an increased sample size, a pathway-driven study design, the use of expression traits derived from co-expression networks) or in the context of replicating earlier epistasis findings, the value of n should be set sufficiently large by the user, in order not to lose signals in the final output. However, when epistasis is tested for in a hypotheses-free way, it is highly unlikely that more than 1000 significant epistatic pairs can be identified ($n = 1000$, default value).

Van Lishout's implementation of maxT exploits all ideas presented so far and in particular the three remarks of Section 3.3.1. The different steps are given by:

1) Generation of the sorted *Real Data* vector R. This vector is created empty with a preallocated size of n and initialized by performing a loop over the first n pairs of SNPs. For $j = 1, \ldots, n$: compute the test-statistic t_j corresponding to the j^{th} pair of SNPs and store an object containing t_j and the names of the corresponding SNPs in R_j. After this loop, sort the vector in decreasing test-statistic order using the *quicksort* algorithm. Then, perform a loop over the remaining $m - n$ pairs of SNPs. For $j = n+1, \ldots, m$: compute the test-statistic t_j corresponding to the j^{th} pair of SNPs. If it is higher than the test-statistic value stored in R_n, overwrite R_n with R_{n-1}. If it is also higher than the test-statistic value stored in R_{n-1}, overwrite R_{n-1} with R_{n-2}. Continue this process until a higher value than t_j is reached (or the beginning of R). Then create a new object containing t_j and the corresponding SNP names and store it in the one but last visited cell of the R vector (or the first one). Let's note $t_1 \geq t_2 \geq \ldots \geq t_m$ the test-statistics obtained at the end of the loop and $(SNP_{l1}, SNP_{r1}), \ldots, (SNP_{lm}, SNP_{rm})$ the corresponding pairs of SNPs.

2) Generation of the test-statistics *Permutation* vector T and the *adjusted p-values* vector P. These vector are created with a preallocated size of n. The T vector is created empty and the P one initialized with 1's. For $i = 1, \ldots, B$: permute randomly the trait values of the subjects, without modifying their genotypes and perform four loops. First, loop over the top n pairs of SNPs. For $j = 1, \ldots, n$: compute $t_{i,j}$ and store it in T_j. Second, initialize M_i to 0 and loop over the remaining $m - n$ pairs of SNPs. For $j = n+1, \ldots, m$: compute $t_{i,j}$ and if it is higher than M_i overwrite M_i with $t_{i,j}$. If the final M_i is higher than T_n, replace T_n by M_i. Third, force the monotonicity of vector T: for $j = n-1, \ldots, 1$ replace T_j by T_{j+1} if $T_j < T_{j+1}$. Fourth, update vector P: for $j = 1, \ldots, n$: if $T_j \geq t_j$ increment P_j by 1.

3) Finalization of the *adjusted p-values* vector. For $j = 1, \ldots, n$: divide P_j by $B+1$. Force monotonicity: for $j = 1, \ldots, n-1$: if $P_{j+1} < P_j$ overwrite P_{j+1} by P_j.

Three remarks are in place:

First, note that at the first step, instead of saving the names of the SNPs in the *Real Data* vector, we store their indexes, i.e. 1 for the first SNP appearing in the input file, 2 for the second one and so on[5]. This obviously saves spaces but more importantly allows to access quicker to the SNPs values when computing a test-statistic value.

[5] Recall that in the actual implementation of the software, we use zero-based numbering for efficiency reasons, i.e. assigning index 0 to the initial element of a sequence.

Second, note that the $t_{i,j}$ values are filled row-by-row and not column-by-column as in Section 3.2.2. This allows in-place permutation of the trait values, requiring a memory of $O(S)$ instead of $O(BS)$ (S is the number of subject). The modern version of the Fisher-Yale shuffle algorithm [31], introduced by Durstenfeld [27] and popularized by Knuth [65], enables to permute the trait values in-place in linear time. A simple loop permutes the values of the trait vector[6] randomly. For $g = 2, \ldots, S$, get a random integer a such that $1 \geq a \geq g$ and exchange the a^{th} and g^{th} elements of the vector. It is easy to prove that if we pick any cell of the trait values vector, at the end of the loop, any element has a probability of $\frac{1}{S}$ to land in that cell.

Three, note that the renaming of the test-statistics at the end of step 1 implies a subtlety that has been hidden so far, to avoid making the core presentation of the algorithm unnecessary complicate. At the first step, the pairs of SNPs are browsed in an order dictated by the input file, whereas at the second step, they are browsed in decreasing test-statistics order. In the classical implementation, this is not an issue. The sorted *Real Data* vector can be used at step 2, to find out the indexes of the SNPs for which a test-statistic value should be computed. However, in Van Lishout's implementation, the sorted *Real Data* vector contains only the names of the top n SNPs[7]. To solve this issue, we create a hash table containing the names of the SNPs belonging to the top n and their corresponding rankings, resolving collision by separate chaining [65]. In this way, at step 2, the pairs can be browsed in the same order as at step 1 and at each iteration, the hash table is used to decide (almost instantaneously) if the current value corresponds to one of the n best pairs or not and act accordingly, i.e. either store the value in T or update M_i.

The main benefit of this new implementation is the same one as for mbmdr-4.4.1.out's implementation of Bonferroni. Indeed, since only objects of size n are again stored, the new implementation makes the memory usage independent from the number of SNPs, whereas the classical implementation raises quadratically with that amount. The illustration of Figure 3.3 still holds here. A GWAIS would require several Terabytes of memory with the classical implementation! Note that Figure 3.3 may be misleading, since it only represents the memory required by Van Lishout's implementation of maxT itself. This does not mean that the memory usage is independent from the number of SNPs globally. Indeed, the dataset itself has to be read at the start of the analysis and stored in memory. This implies a memory proportional to the number of SNPs. For a GWAis with thousands of subjects, this usually represents several Gigabytes. Compared to that, the memory usage required by Van Lishout's implementation of maxT is negligible. In short, one can say that the memory needed by our software is roughly equal to the size of the dataset. As a consequence, the remaining concern is now computing time. At this point, it is thus necessary to construct synthetic data in order to estimate the computing time, for different dataset sizes and try to see whether a GWAIs is already realistic or not. Otherwise, more advanced methods needs to be put in place.

The final implementation of Van Lishout's maxT is reported in Box 3.2.

[6] Whose elements are indexed $1, \ldots, S$.

[7] As already mentioned, in practice, instead of saving the names, we save the indexes of the SNPs

Box 3.2. Van Lishout's implementation of maxT

```
(1) If $m < n$ set $n = m$. Create empty vector $R$ of size $n$.
    (a) For $j = 1, \dots, n$: compute $t_j$ and store object ($t_j$, $SNP_{lj}$, $SNP_{rj}$) in $R_j$.
    (b) Sort $R$ in decreasing test-statistics order using the quicksort algorithm.
    (c) For $j = n + 1, \dots, m$: compute $t_j$ and if $t_j > (R_n)_1$ do
        i. Set k=n. While ($k > 0$ and $t_j > (R_{k-1})_1$) do $\{R_k = R_{k-1}$ ; k−−$\}$.
        ii. Store an object ($t_j$, $SNP_{lj}$, $SNP_{rj}$) in $R_k$.
    (d) Create a hash table containing an entry for each element of $R$, enclosing
        the indexes of the corresponding SNPs and the ranking.
(2) Create empty vectors $T$ and $P$ of size $n$. Fill $P$ with 1's. For $i = 1, \dots, B$:
    (a) For $g = 2, \dots, S$: compute $a$= modulo(rand(), g+1) and exchange the
        $a^{\text{th}}$ and $g^{\text{th}}$ elements of the trait values vector.
    (b) Set $M_i = 0$. For $j = 1, \dots, m$: compute $t_{i,j}$. Search if the $j^{th}$ pair of
        SNPs is in the hash table. If found, retrieve the ranking $r$ and store $t_{i,j}$
        in $T_r$. If not found, overwrite $M_i$ by $t_{i,j}$ if $M_i < t_{i,j}$.
    (c) Replace $T_n$ by $M_i$ if $T_n < M_i$.
    (d) For $j = n - 1, \dots, 1$: replace $T_j$ by $T_{j+1}$ if $T_j < T_{j+1}$.
    (e) For $j = 1, \dots, n$: increment $P_j$ by one if $T_j \geq t_j$.
(3) For $j = 1, \dots, n$: divide $P_j$ by $B + 1$. For $j = 1, \dots, n - 1$: replace $P_{j+1}$ by
    $P_j$ if $P_{j+1} < P_j$.
```

Note that this algorithm is trivial to adapt for users interested in performing a main effects analysis. The implementation from Box 3.2 needs only a minuscule adaptation: at steps (1)(a) and (1)(c)(ii), an object (t_j, SNP_j) should now be stored instead of the (t_j, SNP_{lj}, SNP_{rj}) one. It comes without saying that the variable m represents the number of SNPs here (not the number of pairs anymore) and that single SNP indexes are stored in the hash table (instead of pairs). Similarly, the code of Box 3.2 is easy to adapt for three-order interactions by using an object (t_j, SNP_{lj}, SNP_{mj}, SNP_{rj}) at steps (1)(a) and (1)(c)(ii) instead of the (t_j, SNP_{lj}, SNP_{rj}) one. Obviously, m now represents the number of triplets of SNPs an the hash tables stores the three SNP indexes.

Performances

In order to assess the speed performances we create 4 different datasets with 1,000 individuals each, of respectively 10 SNPs, 100 SNPs, 1,000 SNPS and 10,000 SNPs. To assess significance, the number of permutations is set to the default $B = 999$ value. Each dataset is constructed to contain a strong signal for the functional pair $[SNP_5, SNP_{10}]$. Table 3.2 states the two-locus penetrance table used to generate the data.

Table 3.2: Two-locus penetrance table used to create the simulated data

	AA	Aa	aa
BB	0	0.1	0
Bb	0.1	0	0.1
bb	0	0.1	0

A balanced number of cases and controls is sampled. The minor allele frequencies of the functional SNPs are fixed at 0.5 and those of the non-functional SNPs are generated randomly from a uniform distribution on [0.05, 0.5]. This corresponds to the first of six purely epistatic models discussed in [101]. All experiments are performed on an Intel Xeon CPU E5520@2.27 GHz 5.81 GT/s.

Before this thesis, the only available software implementing the MB-MDR methodology was an R-package based on the classical implementation of maxT [12]. In the context of the accepted oral presentation at the *BBC 2011* conference in Luxembourg, we compared the computing times of this R package with mbmdr-2.6.2.out, one of the earliest versions of our software containing Van Lishout's implementation of maxT. This comparison does obviously cover more than the fact that the R version is based on the classical implementation of maxT and our software on Van Lishout's implementation of maxT. Other major differences includes the fact that the former is in R whereas the latter in C++ and the fact that the computation of the test-statistics in the R version are based on general R functions, whereas in mbmdr-2.6.2.out they are customized as explained in full details in Chapter 2. Table 3.3 summarizes the results.

Table 3.3: Execution times of R package and mbmdr-2.6.2.out

SNPs	pairs of SNPs	R-package	mbmdr-2.6.2.out
10	45	1 h 2 min	1 sec
10^2	4,950	4 days 3h	1 min 3 sec
10^3	499,500	$\approx$ 1.1 year	1 hour 43 min
10^4	49,995,000	$\approx$ 110 years	$\approx$ 7 days 4h

All tests are performed on an Intel Xeon CPU E5520@2.27 GHz 5.81 GT/s. The results prefixed by the symbol "$\approx$" are extrapolated.

One can see that mbmdr-2.6.2.out is approximately 5,500 times faster than its R counterparts. This is in part a consequence of the new algorithm but probably more importantly a repercussion of the fact that the C++ implementation is customized for its task (as also demonstrated in Chapter 2). The fact that C++ is a compiled language does of course also play an important role, since a lot of optimizations could be done while compiling. A benefit of this approach is that the software is standalone. The user does not need to install (and potentially learn) R before starting to use the program. Another conclusion that can be made from Table 3.3 is that this version is very far from being able to perform a GWAIs. A further extrapolation of the computing times shows that a GWAIs would take about 200 years to finish! The purpose of the next subsection is to show how to parallelize the algorithm, in order to save significant computing time.

3.3.3 Parallel version of Van Lishout's implementation

This contribution is also part of the accepted oral presentation at the *BBC 2011* conference in Luxembourg [122] and of the *BMC Bioinformatics 2013* paper [125]. Since all iterations of step 2 of Box 3.2 are independent from each other, it is possible to simultaneously run the computations of the permutations on different machines, as illustrated in Figure 3.7.

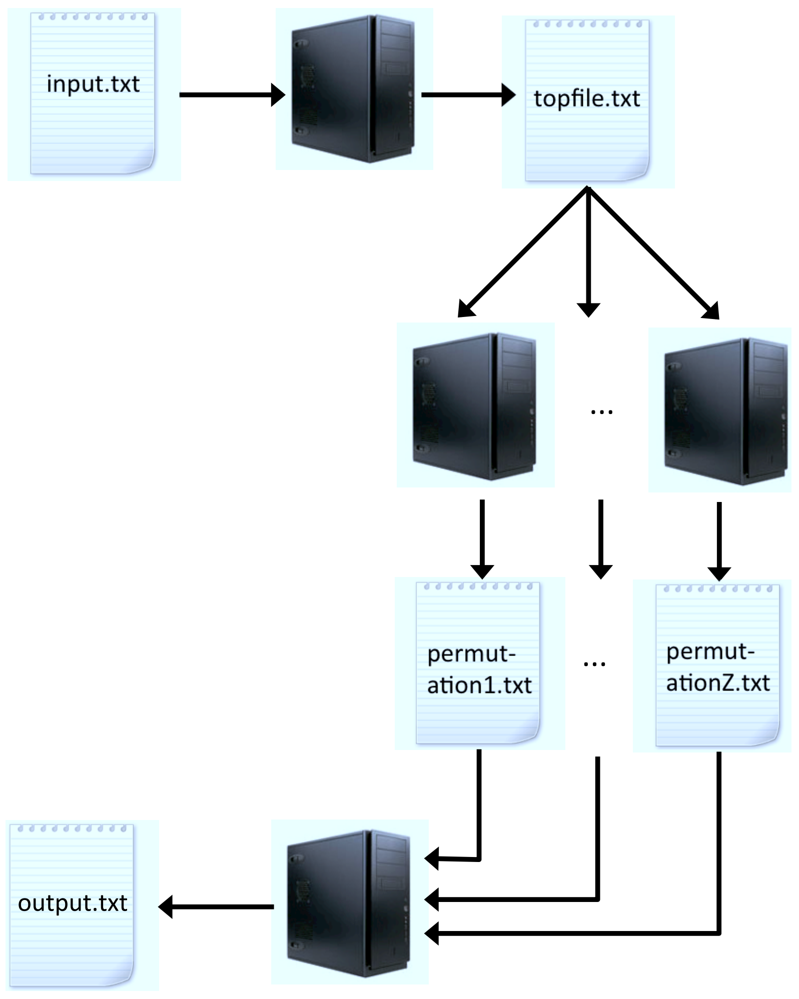


Figure 3.7: Parallel workflow of Van Lishout's implementation of maxT. Step 1 is first performed on the input file on a single machine. This produces the file *topfile.txt*, containing the ranked top pairs of SNPs and their corresponding test-statistics. Then, the computation of the permutations is split between the Z available machines. Finally, the produced *permutation_x.txt* files are read and the final output file is created.

To avoid communication between the machines, which would have as a consequence to slow down the execution, we run several instances of the software and each of them produces partial results that are written in short files. In this way, we avoid having to actually use parallel programming, with multiple threading and such advanced methods. Since the purpose of this implementation is to run on a cluster, on which the different jobs are queued, this choice is perfectly fine.

The three steps of the parallel workflow that we use to solve large-scale problems are the following:

1) Compute the test-statistics for all pairs on one machine and save the n highest ones into a file *topfile.txt*. This file should be saved at a location on which each machine has read access. Its purpose is to save the information of the *Real Data* vector of Figure 3.6. The file is thus tiny, i.e. it has a size that is $O(n)$.

2) Split the computation of the permutations homogeneously between the Z machines. On each machine $z = 1 \dots Z$, perform the following operations:

   (a) Read the file *topfile.txt*

   (b) Initialize a vector P of size n with 0's.

   (c) Execute step 2 (a), (b), (c), (d) and (e) of Van Lishout's implementation of maxT for each permutation assigned to z.

   (d) Save the P vector into a file $permutation_z.txt$.

3) When all machines have terminated their work, sum all vectors of the files $permutation_1.txt \dots permutation_Z.txt$ and add 1, to obtain the final P vector. Perform step 3 of Van Lishout's implementation of maxT on this vector.

Performances

In order to assess the speed performances of the parallel workflow, we construct 4 different datasets with 1,000 individuals in the same way in Section 3.3.2, except that they contain 100 SNPs, 1,000 SNPS, 10,000 SNPs and 100,000 SNPs respectively here. This experiment was done in the context of the *BMC Bioinformatics 2013* paper [125] with mbmdr-3.0.3.out. A similar strategy was used to construct another set of 4 datasets, containing the same number of individuals and SNPs as before, but expressing the trait on a continuous scale instead of a binary one. The software finds again the strong signal in all datasets. Table 3.4 gives the execution times, using either the sequential version of Van Lishout's implementation of maxT on a single core of a cluster composed of 10 blades containing each four Quad-Core AMD Opteron(tm) Processor 2352 2.1 GHz or the parallel workflow on all the available cores. In the latter case, the computation of the permutations is split homogeneously between $10 \times 4 \times 4 = 160$ cores.

Table 3.4: Execution times of the sequential and parallel versions of mbmdr-3.0.3.out

SNPs	Sequential version Binary trait	Sequential version Continuous trait	Parallel workflow Binary trait	Parallel workflow Continuous trait
10^2	45 sec	1 min 35 sec	<1sec	<1sec
10^3	1 hour 16 min	2 hours 39 min	38 sec	1 min 17 sec
10^4	5 days 13 hours	11 days 19 hours	1 hour 3 min	2 hours 14 min
10^5	$\approx$ 1.5 year	$\approx$ 3 years	4 days 9 hours	$\approx$ 9 days

The parallel workflow was tested on a cluster composed of 10 blades, containing each four Quad-Core AMD Opteron(tm) Processor 2352 2.1 GHz. The sequential executions were performed on a single core of this cluster. The results prefixed by the symbol "$\approx$" are extrapolated.

Table 3.4 shows that our software is about twice faster for solving datasets for which the trait is expressed on a binary scale, compared to datasets where the trait is expressed on a continuous one. Furthermore, the table also shows that the execution time is approximately multiplied by 100 when the number of SNPs is multiplied by 10. This is logical, since the computation time mainly depends on how many test-statistics are computed, which in turn depends on the quantity of pairs of SNPs, itself proportional to the squared number of SNPs.

If we further extrapolate the times, we can conclude that for a trait expressed on a binary scale, a GWAIs analysis would take about one year. For a trait expressed on a continuous scale, about two years. This is a big improvement compared to Section 3.3.2, but still not enough to make a GWAIs possible in practice. The purpose of the next subsection, is to find out the main weakness of Van Lishout's implementation of maxT, in order to open the door for a new algorithm speeding up the computations.

Bottleneck

Van Lishout's implementation of maxT still leaves room for improvement. In what follows, we identify its main bottlenecks, in order to improve the overall performance on large-scale data. In Table 3.5 we report the number of test-statistics computed within the different steps of Van Lishout's implementation of maxT (with the default parameters of the software $n = 1000$ and $B = 999$) on a dataset containing 1000 subjects and 10^6 SNPs, which is equivalent to $m \approx 5\text{x}10^{11}$ pairs of SNPs.

Table 3.5: Number of test-statistics computed within the different steps of Van Lishout's implementation of MaxT on a dataset containing 10^3 subjects and 10^6 SNPs.

	theoretical value	numerical value
Step 1 (a)	$O(n)$	1 000
Step 1 (b)	$O(n \log n)$	9 966
Step 1 (c)	$O(m)$	499 999 500 000
Step 1 (d)	$O(n)$	1 000
Step 2 (a)	$O(BS)$	999 000
Step 2 (b)	$O(Bm)$	499 499 500 500 000
Step 2 (c)	$O(B)$	999
Step 2 (d)	$O(Bn)$	999 000
Step 2 (e)	$O(Bn)$	999 000
Step 3	$O(n)$	1 000

Since step 1 (a) consists in computing only n test-statistics, it takes obviously a time proportional to n. Step 1 (b) can be executed using the *quicksort* algorithm, which is known to lead to a computing time proportional to $n \log n$ [65]. Step 1 (c) is the most demanding part of step 1: it implies as many elementary computations as there are pairs of SNPs to test. This is easy to show. First, note that the loop involves $m - n$ iterations and since $m >> n$, this is approximately equivalent to m. Furthermore, steps 1 (c) i. and ii. are executed only when t_j is greater than the lowest test-statistic value in R. Intuitively, the probability of having to insert decreases at each iteration and tends to zero because the vector contains higher and higher values. As a consequence, this algorithm requires $O(m)$ computing time on average, but could degenerate in $O(nm)$. Step 1 (d) consists in n insertions in the hash table.

Since step 2 is performed B times, all computing times are multiplied by B. Step 2 (a) consists in S very quick operations and is not problematic. Step 2 (b) is the most demanding part of step 2: it implies m test-statistic computations. Step 2 (c) is trivial. Steps 2 (d) and (e) both involve a loop consisting of about n quick operations. Step 3 is performed only once and implies two loops consisting of about n iterations. This leads to a computing time proportional to n.

In summary, steps 1 (c) and 2 (b) are the two most demanding steps. Although significance assessment can be based on fewer SNP pairs, this first step of computing the original test-statistics cannot be avoided nor simplified. For this reason, we focus on step 2 (b). With 10^6 inputted SNPs, the number of elementary computations required is proportional to 10^{14}! Therefore, any improvement at this stage leads to better overall performances. In the next section, we introduce a novel algorithm for multiple testing, based on the idea of changing step 2(b) of Van Lishout's implementation of maxT and keeping the rest as it is. This algorithm, called gammaMAXT, is implemented in the software mbmdr-4.4.1.out and resolves remaining concerns about computation time in GWAIs using the MB-MDR framework. A detailed explanation of how we perform such an improvement is the subject of the next section.

3.4 GammaMAXT

This work is the main contribution of this PhD thesis. It has been accepted for oral presentation at the *ERCIM 2014* conference in Pisa [123] and published in *BioData Mining* in 2015 [124]. The core idea of gammaMAXT is that the M_i value computed at step 2 (c) should now be estimated from a sample from $[t_{i,n+1}, \dots, t_{i,m}]$ rather than calculated exactly, as illustrated in Figure 3.8.

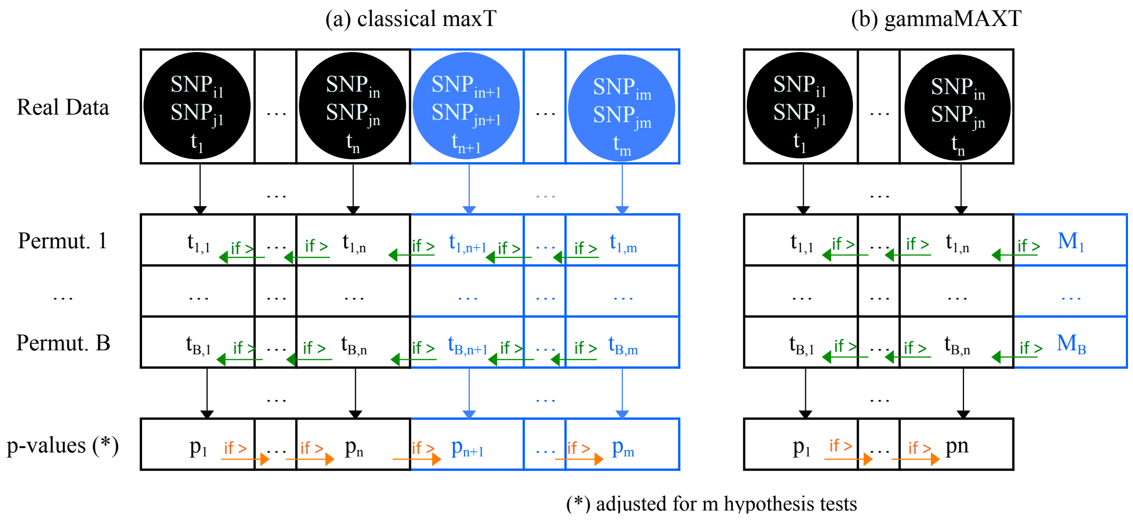


Figure 3.8: Classical implementation of maxT versus gammaMAXT. In the classical implementation, test-statistics are computed for all pairs of SNPs and stored with the corresponding SNP names in the sorted *Real data* vector. In gammaMAXT the same computations are performed, but only the top n test-statistics and corresponding SNP names are stored. In the classical implementation, all $t_{i,j}$ values are computed and stored $\forall i = 1 \dots B$, $\forall j = 1 \dots m$. In gammaMAXT, the maximum M_i of $[t_{i,n+1}, \dots, t_{i,m}]$ is estimated from a sample from $[t_{i,n+1}, \dots, t_{i,m}]$ rather than calculated exactly. In the classical implementation, $t_{i,j}$ is overwritten by $t_{i,j+1}$ whenever $t_{i,j+1} > t_{i,j}$, $\forall j = m-1 \dots 1$. In gammaMAXT $t_{i,n}$ is overwritten by M_i if $M_i > t_{i,n}$ and $t_{i,j}$ is overwritten by $t_{i,j+1}$ whenever $t_{i,j+1} > t_{i,j}$, $\forall j = n-1 \dots 1$. Finally, p_{j+1} is overwritten by p_j whenever $p_j > p_j + 1$, $\forall j = 1 \dots m-1$ in the classical case and $\forall j = 1 \dots n-1$ in gammaMAXT.

This idea emerged from the intuition that computing all the numbers in $[t_{i,n+1}, \dots, t_{i,m}]$ to keep just one, the maximum M_i, implies a huge computing time that could probably be invested better than this. This represents about 10^{11} test-statistics computations in the context of a GWAIs. In fact, the $t_{i,n+1}, \dots, t_{i,m}$ numbers follow an unknown distribution and basically what we are doing is fitting this distribution very precisely. Actually, more precisely than intuitively needed and that's the starting point of the concept. The distribution has been fit so precisely that instead of searching the maximum in $[t_{i,n+1}, \dots, t_{i,m}]$ directly, we could almost predict it from the fitted distribution. Even better, we actually do not need to compute so much numbers to fit the distribution, a sample of size $A = 10^6$ of non-zero values should suffice. This size is a tradeoff between computing time and precision of the prediction. The main remaining question is therefore which probability distribution to fit and is addressed in the next section.

3.4.1 Distributational assumptions

We have indicated before that MB-MDR offers a flexible framework to test for SNP-SNP interactions. The software in which the framework is implemented has a modular built-up that allows a flexible choice of association test, depending on the input data. For instance, for quantitative traits, t-tests or non-parametric equivalents can be carried out. For binary traits, chi-squared test of independence can be chosen. The association test that best reflects the data at hand is used in both stage 1 and stage 2 of the MB-MDR framework [16, 82]. After the data manipulation of combining cells using trait information, MB-MDR's final test statistic no longer follows the theoretical sample distribution of the initially chosen test statistic. In fact, earlier work has shown that such sequential pooling may lead to permutation-based distributions of MB-MDR test-statistics that depend on the number of multi-locus genotype cells pooled [11] or on the minor allele frequencies (MAFs) of the SNP pair under consideration [79].

Rather than looking at the null distribution of the test statistic linked to a SNP-pair, we are now interested in the distribution of a number of test values over several SNP-pairs, from which to derive the maximum value M_i. We hypothesize that test values in $[t_{i,n+1}, \ldots, t_{i,m}]$, follow a mixture distribution of a shifted gamma distribution [67] and a point mass at zero. Note that zero test values are induced by scenarios for which the MB-MDR test statistic cannot be computed. We have seen in Chapter 2 that whenever a group of subjects (e.g., in a 2-SNP interaction study, those subjects having two copies of the minor allele at each locus) is compared to the remaining subjects with respect to the trait under study and by using an appropriate association test statistic, this group can either be associated to a higher "risk" ("H" category), a lower "risk" ("L" category) or indecisive "risk" (nor "H", nor "L"; "O" category) for the trait. Here, "risk" is used loosely. For instance for continuous traits, the "risk" categories above may rather refer to increased ("H" category), decreased ("L" category) mean trait values. Also, in the MB-MDR methodology, risk scales can be refined to incorporate multiple risk categories [16]. It is important to realize that if all subjects are assigned the same label (in this scenario, most probably the "O" label), then MB-MDR returns an exact zero. It is not surprising that lack of power of GWAIs (which depends on sample size but also true effect size) induces such technical zeros for a significant proportion of the tested SNP pairs.

In order to take this important amount of zeros into account, we use the approach described in [49]. We assign a discrete probability mass to the exact zero value. Hence, if $\mathcal{X}_i$ is a random variable returning a random value from $[t_{i,n+1}, \ldots, t_{i,m}]$, we can define the probabilities $\pi = P(\mathcal{X}_i > 0)$ and $1 - \pi = P(\mathcal{X}_i = 0)$. Therefore, the distribution of $\mathcal{X}_i$ is semi-continuous with a discontinuity at zero. This implies that the probability density function is

$$f_{\mathcal{X}_i}(x) = (1 - \pi)\delta(x) + \pi g_{\mathcal{X}_i}(x)\mathbb{1}_{(x>0)} \tag{3.2}$$

where $\delta(x)$ is a point probability mass at $x = 0$, $g_{\mathcal{X}_i}(x)$ is the distribution of the strictly positive values and $\mathbb{1}_{(x>0)}$ is an indicator function taking the value 1 if $x > 0$ and 0 otherwise. The parameter π depends on the data at hand and can be estimated with the Maximum Likelihood Estimation (MLE) method [7] from the observed frequency in a sample from $[t_{i,n+1}, ..., t_{i,m}]$.

Due to the fact that our main goal consists in predicting a maximum, we are not particularly interested in fitting the distribution of $g_{\mathcal{X}_i}(x)$ on the entire set of strictly positive values. As a matter of fact, fitting the tail of $g_{\mathcal{X}_i}(x)$ should suffice. We show in the next section that focusing on the top 10% strictly positive values is an acceptable practical choice. We consider this a good tradeoff between fitting on a large and a smaller range of positive values. The former might lead to a poor fit of the tail, because many samples might not belong to that range. The latter might lead to a poor fit of the tail due to an insufficient number of samples. The amount of values belonging to the top 10% strictly positive values in $[t_{i,n+1}, \dots, t_{i,m}]$ is given by $q = \frac{(m-n)\pi}{10}$.

Assumption 1

We assume that the shifted gamma distribution is a good fit to the tail of $g_{\mathcal{X}_i}(x)$. Hence, if $\mathcal{Y}_i$ is a random variable returning a value from the aforementioned top 10% of strictly positive values, we postulate that its cumulative distribution function (CDF) is given by

$$F_{\mathcal{Y}_i}(y) = \frac{\gamma(k, \frac{y-y_0}{\theta})}{\Gamma(k)} \tag{3.3}$$

where γ is the lower incomplete gamma function, y_0 is the location parameter, k is the shape parameter and θ the scale parameter. Some authors discourage the use of the gamma distribution for model fitting due to the difficulty of parameter estimation [2]. However, in the specific case of fitting the tail of the distribution of the MB-MDR statistics, we believe that simpler models would be consistently inaccurate. Moreover, the lack of knowledge regarding the shape of a plausible distribution and the diversity of the data we are performing our computations on, make a versatile distribution function like the gamma, a reasonable assumption. Note that the choice of shifting the gamma distribution comes naturally due to the fact that the smallest strictly positive value should not be in the neighborhood of zero. Indeed, a small value would represent a low-significant association between the interaction of the two loci and the phenotype. As previously mentioned, this would lead to the "O" category for all subjects and an exact zero. The CDF of the random variable $\mathcal{Z}_i$ returning the maximum of the q values belonging to the top 10% strictly positive values in $[t_{i,n+1}, \dots, t_{i,m}]$ is given by

$$F_{\mathcal{Z}_i}(z) = [\frac{\gamma(k, \frac{z-y_0}{\theta})}{\Gamma(k)}]^q \tag{3.4}$$

Indeed, if we take q independent and identically distributed (i.i.d.) values $y_1, y_2, ..., y_q$, then $P[(y_1 \leq y_t) \wedge (y_2 \leq y_t) \wedge ... \wedge (y_q \leq y_t)] = [F_{\mathcal{Y}_i}(y_t)]^q = F_{\mathcal{Z}_i}(z)$.

Assumption 2

We postulate that the parameters π, y_0, k and θ are stable from one permutation to another. This assumption is a plausible one, given the results in Table 3.7, which show low variance of these parameters across 999 permutations. An analogous observation has been noticed in a similar work [92], based on hypothesis testing with an extreme value distribution. In order to reduce the computational burden of the fitting, we estimate the parameters once every 20 permutations. We consider this a compromise between robustness and performance.

Simulations supporting assumption 1

In this section, we investigate the hypothesis that the tail of $g_{\mathcal{X}_i}(x)$ follows a shifted gamma distribution and that fitting the top 10% of strictly positive values is an acceptable choice. We use the following datasets for this experiment:

- A simulated dataset D_1 expressed on a binary scale, composed of 1000 SNPs and 1000 individuals. Table 3.6 states the two-locus penetrance table used to generate it. A balanced number of cases and controls is sampled. The minor allele frequencies of the functional SNPs are fixed at 0.5 and those of the non-functional SNPs are randomly generated from a uniform distribution on [0.05, 0.5]. This corresponds to the first of six purely epistatic models discussed in [101]. Furthermore, any value in the dataset had a 5% chance to be missing.
- A simulated dataset D_2, with the same properties as D_1, except that the trait is expressed on a continuous scale.
- A simulated dataset D_3, with the same properties as D_1, except that the MAF's are on average lower, i.e. the non-functional SNPs were randomly generated from a uniform distribution on [0.05, 0.1].
- A real-life dataset D_4 on Crohn's disease, for which the trait is expressed on a binary scale [73, 5], reduced to 12471 SNPs and 1687 subjects as in [125].

Table 3.6: Two-locus penetrance table used to create the simulated datasets D_1, D_2 and D_3

	AA	Aa	aa
BB	0	0.1	0
Bb	0.1	0	0.1
bb	0	0.1	0

For each of the aforementioned datasets, we first carry out the original Van Lishout's implementation of maxT based on 10^4 permutations to generate a reference distribution for M_i. We second execute step (2)(b) of the gammaMAXT algorithm based on 10^4 permutations, with different values for the internal parameter defining the percentage of strictly positive values belonging to the tail of $g_{\mathcal{X}_i}(x)$. Figure 3.9 is generated in R and shows the results for dataset D_1. We observe that focusing on respectively 25%, 20%, 15%, 5% and 1% of the strictly positive values leads to a good fit, but that 10% is the optimal alternative. The curves of subfigure (d) are indeed close and the Kolmogorov-Smirnov (KS) distance is the lowest among these choices. This supports the hypothesis that the gammaMAXT algorithm produces accurate predictions of the M_i values. Figures 3.10, 3.11 and 3.12 show that 10% is consistently a good option, although not always the most optimal one.

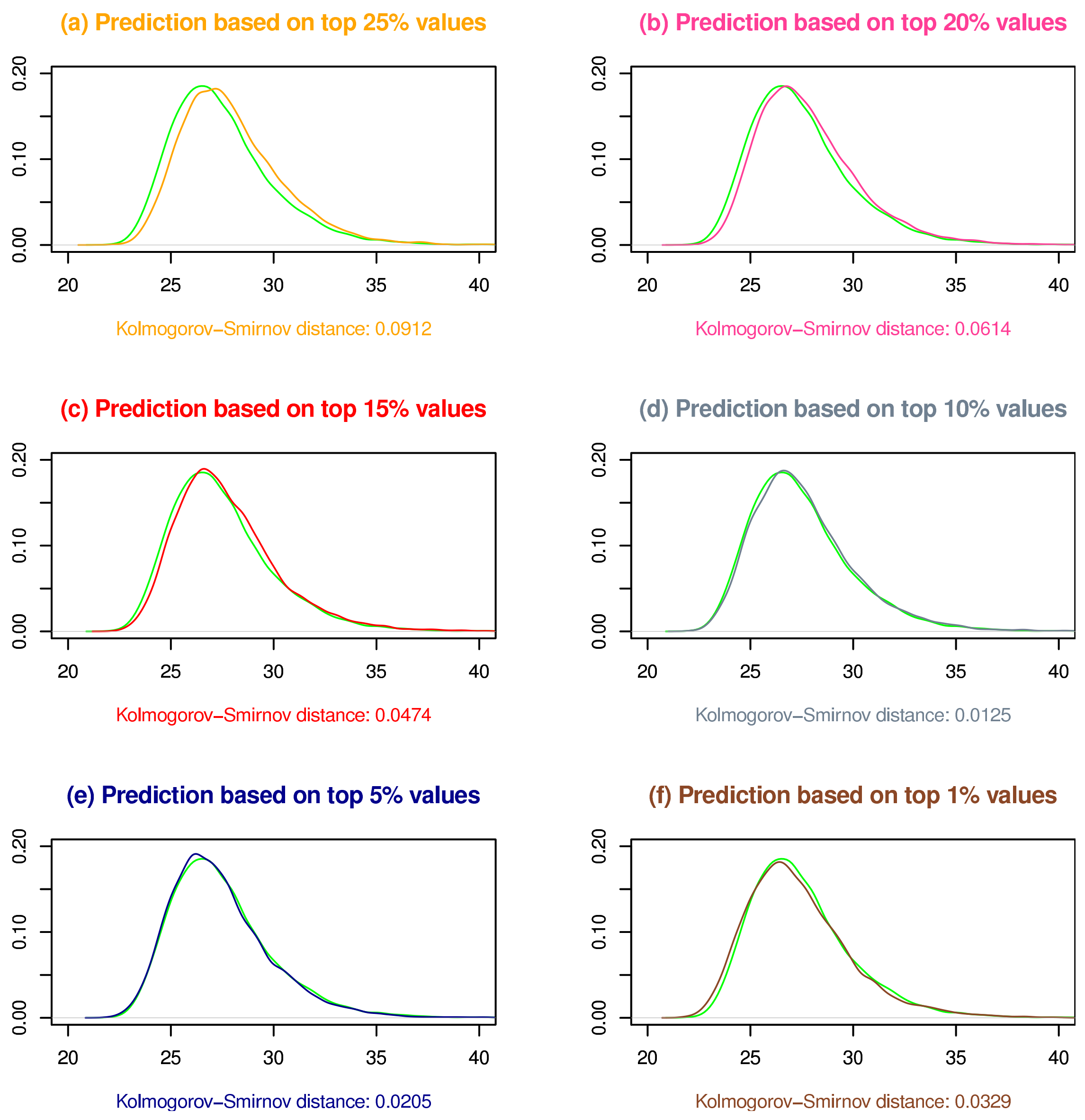


Figure 3.9: Theoretical (green) versus predicted M_i values for dataset D_1. The curve in Grey, corresponding to 10% is the optimal choice, leading to the lowest Kolmogorov-Smirnov distance.

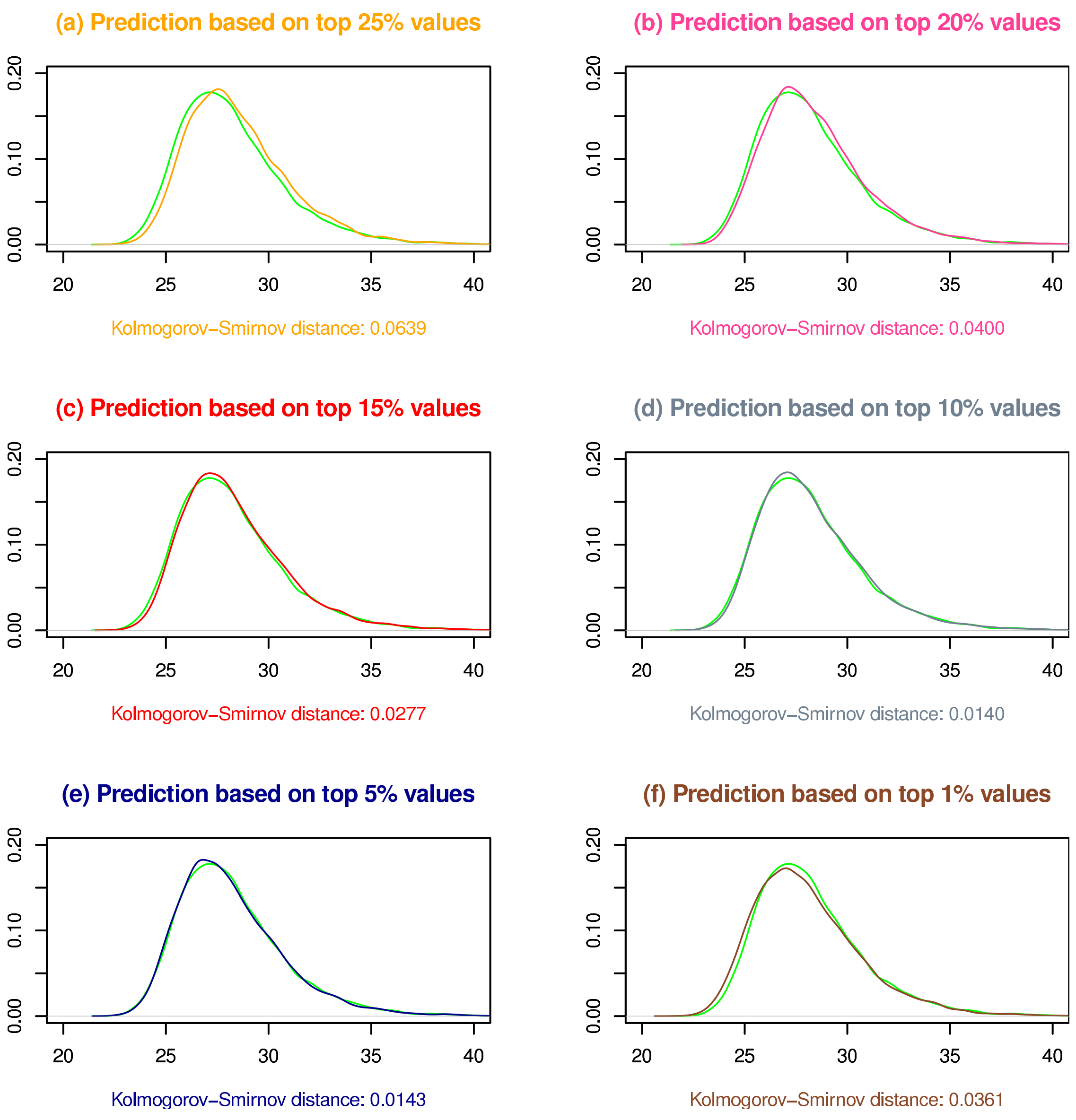


Figure 3.10: Theoretical (green) versus predicted M_i values for dataset D_2. The curve in Grey, corresponding to 10% is the optimal choice, leading to the lowest Kolmogorov-Smirnov distance.

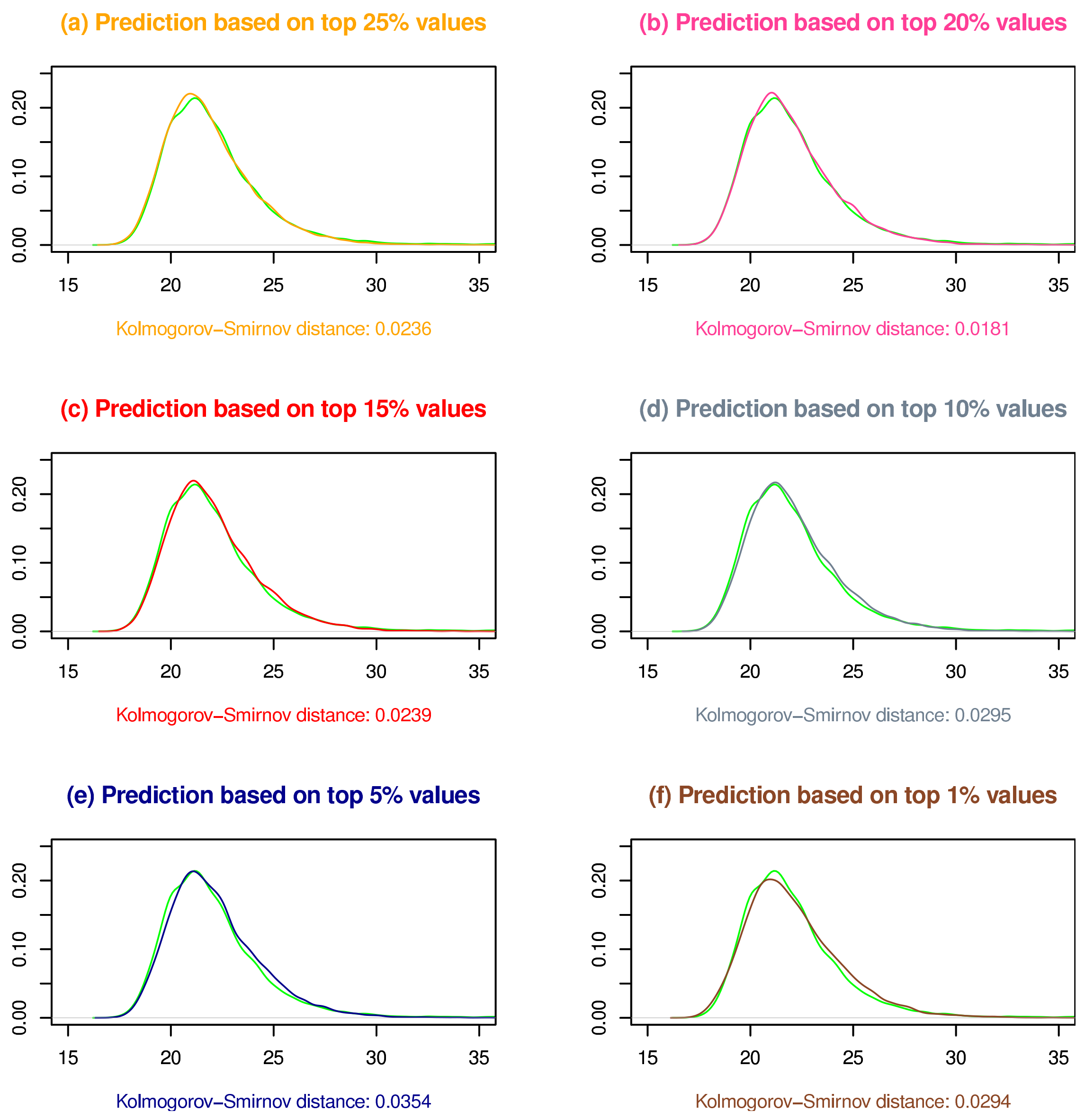


Figure 3.11: Theoretical (green) versus predicted M_i values for dataset D_3. The pink curve, corresponding to 20% is the optimal choice, but the curve in grey corresponding to 10% still leads to a low Kolmogorov-smirnov distance and remains a good practical choice.

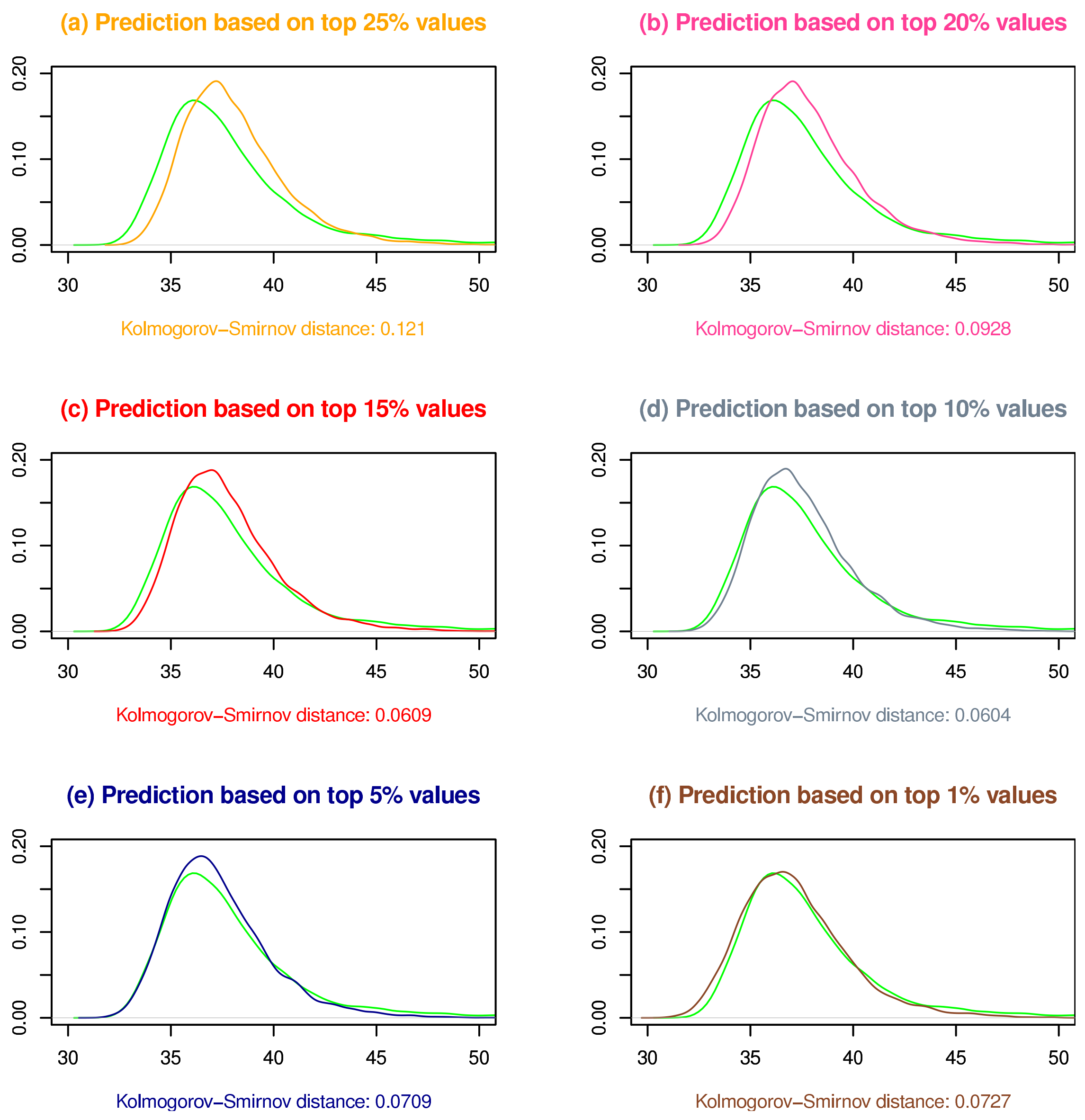


Figure 3.12: Theoretical (green) versus predicted M_i values for dataset D_4. The curve in Grey, corresponding to 10% is the optimal choice, leading to the lowest Kolmogorov-Smirnov distance.

Simulations supporting assumption 2

In this section, we show results indicating that the parameters π, y_0, k and θ are stable across permutations. We perform analyzes on datasets D_1 to D_4, using the default settings. For this experiment, we modified the gammaMAXT algorithm such that it fits new parameters for each of the 999 permutations (not only once every 20 as previously mentioned) and saves these into a file. We report their means and variances in Table 3.7. We observe that the variances are very low across all scenarios.

Table 3.7: Mean and variance of the fitted parameters for datasets $D_1 - D_4$

	$\mathbf{D_1}$ **Mean**	$\mathbf{D_1}$ **Var**	$\mathbf{D_2}$ **Mean**	$\mathbf{D_2}$ **Var**	$\mathbf{D_3}$ **Mean**	$\mathbf{D_3}$ **Var**	$\mathbf{D_4}$ **Mean**	$\mathbf{D_4}$ **Var**
π	0.337	1.247x10^{-6}	0.335	3.815x10^{-6}	0.137	4.948x10^{-7}	0.366	9.356x10^{-7}
$\mathbf{y_0}$	7.742	5.566x10^{-4}	7.825	8.778x10^{-4}	6.189	6.472x10^{-4}	7.788	3.805x10^{-4}
k	1.017	2.612x10^{-4}	1.012	2.534x10^{-4}	0.990	3.580x10^{-4}	1.017	1.725x10^{-4}
θ	1.917	1.462x10^{-3}	1.974	1.532x10^{-3}	1.694	1.829x10^{-3}	1.917	9.695x10^{-4}

3.4.2 Implementation

As mentioned earlier, the gammaMAXT algorithm only differs from Van Lishout's implementation of maxT (Box 3.2) with respect to step 2(b). In the novel implementation the maximum M_i is estimated from a sample of size $A = 10^6$ of strictly positives values in $[t_{i,n+1}, \ldots, t_{i,m}]$ rather than calculated directly. The parameter π is estimated on the fly using a variable z, counting the number of zeros encountered during the sampling process. The new step 2(b), described in Box 3.3, assumes that $m >> n$.

Box 3.3. Step 2(b) of gammaMAXT

```
(1) For $j = 1, \ldots, n$: compute the test-statistic value corresponding to the pair
    stored in $R_j$ and save it in $T_j$.
(2) If (i modulo 20 = 1) estimate $\pi, y_0, k$ and $\theta$: (otherwise use the latest estimates)
    (a) Set $z = 0$. Create empty vector $V$ of size A=$10^6$.
    (b) Randomly select an integer $r$ in $[1, m]$. If the $r^{\text{th}}$ pair of SNPs is in the
        hash table, restart this step.
    (c) Compute test-statistic $t_{i,r}$. If $t_{i,r} = 0$ do z=z+1, else store $t_{i,r}$ in $V$.
    (d) Repeat steps (b) and (c) until $V$ is full.
    (e) Sort $V$. Remove the 90% lowest values. The new size of $V$ is $N = \frac{A}{10}$.
    (f) Estimate $\pi = \frac{A}{z+A}$.
    (g) Estimate $y_0$ by the minimum of $V$.
    (h) Estimate $k$: see below.
    (i) Estimate $\theta = \frac{1}{kN} \sum_{i=1}^{N} (V_i - y_0)$.
(3) Sample $M_i$ from the distribution of the maximum, whose CDF is given by
    $F_{\mathcal{Z}_\rangle}(z) = [\frac{\gamma(k, \frac{z-y_0}{\theta})}{\Gamma(k)}]^{\frac{(m-n)\pi}{10}}$. The detailed procedure is described in Box 3.4.
```

First, note that the probability of restarting step (2) (b) of this procedure is very low. Indeed, since $m >> n$, the numbers in $[1, \ldots, m]$ are almost all indexes of SNPs that do not belong to the top n, i.e. are not in the hash table. As a consequence, this step cannot loop infinitely in practice. As an example, for a GWAIs, the probability that this step restarts ten times or more is approximately equal to 10^{-80}.

Second, note that whereas estimates in steps (2)(f), (2)(g) and (2)(i) are obtained via Maximum Likelihood, the estimation of the parameter k requires more elaboration. According to [86], an acceptable initial guess being within 1,5% of the correct value is

$$k = \frac{3 - s + \sqrt{(s-3)^2 + 24s}}{12s} \tag{3.5}$$

with

$$s = ln(\frac{1}{N}\sum_{i=1}^{N}(v[i] - y_0)) - \frac{1}{N}\sum_{i=1}^{N} ln(v[i] - y_0) \tag{3.6}$$

This initial guess is updated iteratively via the Newton-Raphson method [19]. In particular, in every iteration, k is updated as

$$k = k - \frac{ln(k) - \psi(k) - s}{\frac{1}{k} - \psi'(k)} \tag{3.7}$$

until the difference between the new and the old value of k is lower than the desired precision (default: 0.000001). $\psi(k)$ and $\psi'(k)$ are respectively the digamma and trigamma functions.

Finally, Box 3.4 describes the procedure used at step (3) to obtain the final M_i estimation. Note that it is important to sample in the distribution of the maximum, rather than take the expectation of the maximum. This is the proper way to mimic the original maxT algorithm correctly. Indeed, taking the expectation would lead to about the same predicted maximum for each of the B permutations, so that most p-values would be equal to either $\frac{1}{B}$ or 1 (the observed maximum is compared to about the same value each time and is thus either always lower or always bigger). By sampling, we intuitively give back to maxT what it needs: several maximum values from which it can approximate the distribution of the maximums to compare the observed one to. The idea of the sampling method of Box 3.4 is to take a random p-value and search the corresponding maximum value in the CDF using a Newton-Raphson strategy.

Box 3.4. Algorithm for sampling M_i when CDF is given by $F_{\mathcal{Z}_i}(z)$

(a) Take a too high initial guess of M_i (default: 1000). Initialize variable b to half of this value.

(b) Randomly select a real number $r_n \in [0, 1]$.

(c) If $F_{\mathcal{Z}_i}(M_i)$ is lower than r_n do $M_i = M_i + b$, otherwise do $M_i = M_i - b$. Divide b by 2.

(d) Repeat step (c) until b is below the desired precision (default: 0.000001).

3.4.3 Parallel version

We have presented a parallel workflow for Van Lishout's implementation of maxT in Section 3.3.3. The idea was to parallelize only the computations related to the permuted data, not those on the observed data, as illustrated in Figure 3.7. This did make perfect sense: we have seen in Table 3.5 that for a GWAIs, most of the computation time is spent on permuted data rather than observed one (10^{14} versus 10^{11} computations). In Van Lishout's maxT, running the computations related to the observed data in parallel would not make a big difference on the global computing time. This is totally different for gammaMAXT, since step (2)(b) has been reduced to about 10^7 computations here. As a consequence, the bottleneck is now the computation of the *Real Data* vector, which should be split between all available cores. The computations of the permutations can anyway happen in parallel since it still takes significant time in practice. The final parallel workflow is illustrated in Figure 3.13.

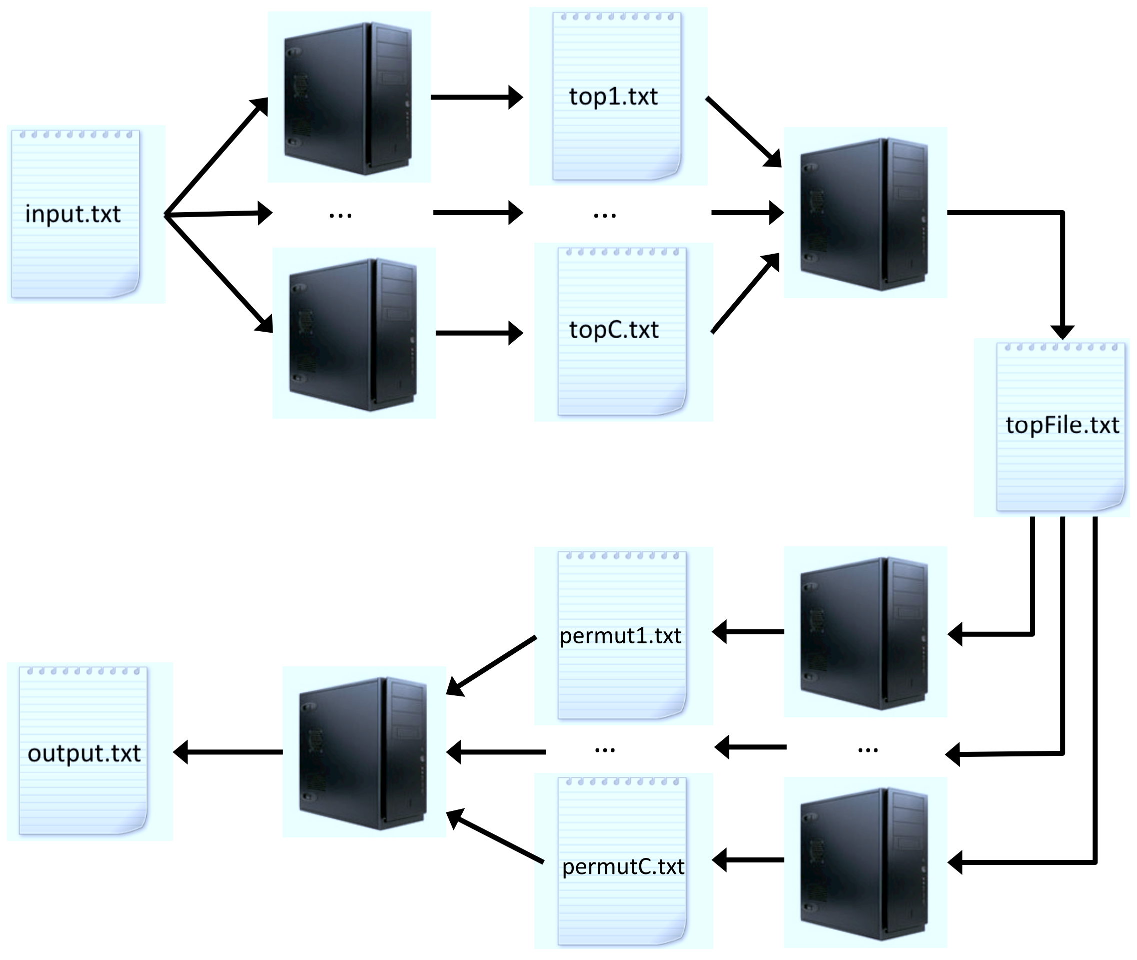


Figure 3.13: First, each cluster node performs a fair proportion of the $t_1, \ldots, t_m$ values computations from Figure 3.8 and saves the n highest into file *top_ c.txt*. Second, a node aggregates all *top_ c.txt* files and retrieves the overall n highest values, saved in *topfile.txt*. Third, each cluster node reads *topfile.txt* and performs an equitable fraction of the B permutations of Figure 3.8, saving results into file *permut_ c.txt*. Finally, a cluster node aggregates all *permut_ c.txt* and produces the final output file.

The four steps of the parallel workflow used to perform GWAIs are the following:

1) Each cluster node $c = 1 \dots C$ performs an equitable fraction of the computations of the $t1, \dots, t_m$ values from Figure 3.8. The n highest values (and corresponding SNP pair indexes) from each node are saved into file *top_ c.txt*.

2) Upon termination of all computations at the previous step, a cluster node aggregates all *top_ c.txt* files and retrieves the overall n highest values (and corresponding SNP pair indexes). Results are saved into *topfile.txt*.

3) Each cluster node reads *topfile.txt*, creates an empty vector T of size n and a vector P of size n filled with 0's and performs an equitable fraction of the B permutations of Figure 3.8. For each permutation i attributed to node c:

   (a) Generate a random permutation of the trait column.

   (b) Execute step (2)(b) of the gammaMAXT algorithm.

   (c) Replace T_n by M_i if $T_n < M_i$.

   (d) For $j = n - 1, \dots, 1$: replace T_j by T_{j+1} if $T_j < T_{j+1}$.

   (e) For $j = 1, \dots, n$: increment P_j by one if $T_j \geq t_j$.

   Upon completion of all computations on node c, save P into file *permut_ c.txt*.

4) A cluster node sums all vectors from the *permut_ c.txt* files to obtain the final vector P. All elements of P are incremented by 1 and divided by $B + 1$. The monotonicity is forced: for $j = 1, \dots, n - 1$, replace P_{j+1} by P_j if $P_{j+1} < P_j$.

Performances

In order to assess the speed performances of the parallel workflow, we create 4 different datasets with 1000 individuals each, of respectively 10^3, 10^4, 10^5 and 10^6 SNPs. This experiment was done in the context of the *BioData Mining 2015* publication [124] with mbmdr-4.2.2.out, the first public version based on gammaMAXT. All datasets were generated using GAMETES, a fast, direct algorithm for generating pure epistatic models with random architectures [120]. Another set of 4 datasets, containing the same number of individuals and SNPs, but expressing the trait on a continuous scale, was created using a similar strategy as for D_2 in Section 3.4.1. Table 3.8 shows the results.

Table 3.8: Execution times of the sequential and parallel versions of mbmdr-4.2.2.out

SNPs	Sequential version Binary trait	Parallel workflow Binary trait	Sequential version Continuous trait	Parallel workflow Continuous trait
10^3	13 min 33 sec	20 sec	13 min 18 sec	18 sec
10^4	52 min 15 sec	1 min 05 sec	56 min 14 sec	53 sec
10^5	64 hours 35 min	22 min 15 sec	70 hours 03 min	20 min 28 sec
10^6	$\approx$ 270 days	25 hours 12 min	$\approx$ 290 days	24 hours 06 min

The parallel workflow was tested on a 256-core computer cluster (Intel L5420 2.5 GHz 1333 MHz FSB). The sequential executions were performed on a single core of this cluster. The results prefixed by the symbol "$\approx$" are extrapolated.

We observe that mbmdr-4.2.2.out outperforms the computing times of mbmdr-3.0.3.out from Section 3.3.3 reported in Table 3.4. For instance, solving a continuous dataset of 10^4 SNPs on a single core takes about 56 minutes with mbmdr-4.2.2.out and almost 12 days with mbmdr-3.0.3.out, i.e. about 300 times less. Solving a continuous dataset of 10^6 SNPs on a 256-core cluster takes about one day with mbmdr-4.2.2.out and would take about 10^4 longer with mbmdr-3.0.3.out.

Note that the computing times reported in Table 3.4 are based on runs without any correction for the main effects of the SNPS. In this case, the times corresponding to a binary trait are about twice faster than those based on a continuous case. In Table 3.8, a codominant correction for the main effects of the SNPs has been performed, implying a regression framework. Since the latter is comparable in the binary and continuous case, we logically observe similar computing times.

The theoretical computing time of step (2)(b), which was $O(Bm)$ with Van Lishout's implementation of maxT according to Table 3.5, is now independent from m. Indeed, once every 20 permutations, this step requires $A = 10^6$ computations to fit the shifted gamma distribution and 19 times out of 20 no computations at all since the previously computed parameters are reused. When the default parameter $B = 999$ is used, parameters are estimated 50 times and the total number of test-statistics computed is therefore about 10^7. When this algorithm is used in the context of a GWAIs, step (2)(b) becomes negligible compared to step 1, which requires about 10^{11} computations according to Table 3.5. In this context, the global computing time is therefore $O(m)$, a big improvement compared to $O(Bm)$ (winning three orders of magnitude!).

When m gets smaller, the computing time spent at step 1 goes down, while the computing time spent at step (2)(b) does not change. This means that at some point, the latter becomes the most demanding one. This is why the parallel workflow executes both of this steps in parallel, so that the user does not need to investigate which step is the most demanding one in his particular case. Note that if m gets very small, at some point Van Lishout's implementation of maxT gets even faster than the gammaMAXT algorithm. For this reason, our software automatically executes the former instead of the latter in such scenarios. In practice, the software turns to Van Lishout's implementation when $m < 15000$ or when $m < 3n$. The former condition is the result of speed tests showing that this is about the threshold for which both algorithm reaches approximately the same speed. The latter condition ensures that step (2) (b) of Box 3.3 does not loop too long (recall that Box 3.3 should only be used when $m >> n$ to ensure that this step is quick). By the way, the $m < 3n$ condition can only apply if the user did select a higher n value than the default one ($n = 1000$). The intuition behind these two conditions is that estimating the M_i values rather than computing them exactly does not make sense when the computing time that is saved is small or that the algorithm becomes even smaller!

We have demonstrated that the execution time of mbmdr-4.2.2.out is significantly lower than the one of mbmdr-3.0.3.out in the context of a large-scale analysis. In the next section, we investigate whether this speed increase comes at the cost of a significant increase/decrease of the FWER/power or not.

3.4.4 FWER and power analysis

FWER

To study the control of the FWER, we run mbmdr-4.2.2.out on four sets of datasets:

- A set S_1 of 1000 datasets, each composed of 1000 SNPs and 1000 individuals, containing null data generated randomly from a uniform distribution on [0.05, 0.5]. A balanced number of cases and controls is sampled.
- A set S_2 with the same properties as S_1, except that the trait is expressed on a continuous scale.
- A set S_3 of 200 datasets, each composed of 10^4 SNPs and 1000 individuals, constructed in the same way as S_1.
- A set S_4 with the same properties as S_3, except that the trait is expressed on a continuous scale.

We report the observed false-positive rates in Table 3.9.

Table 3.9: Observed FWER of mbmdr-4.2.2.out

set	amount datasets	observed FWER
S_1	1000	4.5%
S_2	1000	6.2%
S_3	200	7%
S_4	200	6.5%

In practice, these are computed as the percentage of datasets containing at least one pair of SNPs that gave rise to an adjusted p-value below 5%. On each set, we note that the estimated rates are within the interval $[2,5\% - 7,5\%]$ and satisfies thus Bradley's liberal criterion of robustness for the significance level $\alpha = 5\%$ [8]. This criterion specifies that the FWER are controlled for any significance level α, if the empirical rate $\hat{\alpha}$ is contained in the interval $0.5\alpha \leq \hat{\alpha} \leq 1.5\alpha$. Now that we have shown that the gammaMAXT algorithm still controls the FWER at 5% in practice, using it instead of Van Lishout's implementation of maxT can only have one drawback: power. The purpose of the next experiment is to study how strongly the power has been affected by the fact that the M_i values are now estimated rather than computed.

Power

To evaluate the power, we create nine sets of data with GAMETES. Each set consists in 1000 datasets, all composed of 1000 individuals (500 cases and 500 controls) and 200 SNPs (out of which exactly one pair is linked to the trait). The heritability varies across the datasets from 0.03 to 0.01. In this way, we provide a range of decreasing effect sizes showing the power reduction. Table 3.10 indicates the percentage of time that the pair linked to the trait gave rise to an adjusted p-value below 5%.

Table 3.10: Power comparison between the gammaMAXT and the MaxT algorithms

heritability	gammaMAXT	MaxT
0.0100	3.7%	4.2%
0.0125	17.9%	19.4%
0.0150	50.3%	51.5%
0.0175	67.0%	68.7%
0.0200	86.6%	87.9%
0.0225	94.3%	94.7%
0.0250	97.5%	97.8%
0.0275	99.2%	99.3%
0.0300	99.6%	99.6%

We observe that the original MaxT and the new gammaMAXT algorithm lead to very similar values for the power. By predicting the M_i values instead of computing them explicitly, we can of course not win power, so that the power of the gammaMAXT algorithm is obviously equal or lower than the one of MaxT. However, we observe that the difference is small, the largest power reduction being of 1,7%. We conclude that the huge computing time that is saved by gammaMAXT versus Van Lishout's maxT is worth giving up a very small power percentage.

Note that a drawback of maxT, is that when the test statistics are not identically distributed, unbalanced adjustments can be observed because not all tests contribute equally to the computed adjusted p-values. If this is an issue for a particular experiment, then users can turn to the minP algorithm presented in the next section. This can for instance be very useful when there are a lot of SNPs with a low MAF.

3.5 MinP

The single-step and step-down minP adjusted p-values are two algorithms introduced by Westfall & Young to handle the multiple-testing problem [136]. Since the former tends to be conservative for the control of the FWER, we focus on the latter in this thesis and call it simply minP in this document.

3.5.1 Classical implementation

In Section 3.3, we have shown that the first step of maxT consists in computing test-statistics for all pairs of SNPs and sorting them in decreasing order. In minP, these test-statistics are still computed but not sorted anymore. What is done is computing the corresponding marginal empirical p-values (computed in the exact same way as in the Bonferroni case, as described in Section 3.2.1) and sort these. Without loss of generality, let's note them $p_1^* \leq p_2^* \leq \ldots \leq p_m^*$. The sorting occurs of course in increasing order, since the top results are the lowest marginal empirical p-values in minP, rather than the highest test-statistics in maxT, hence the names of the methods.

In maxT, the adjusted p-value p_1 is given by the probability of observing a maximum value that is at least as extreme as the observed maximum t_1. By symmetry, in minP, the adjusted p-value p_1 is given by the probability of observing a minimum marginal empirical p-value that is at most as extreme as the observed one p_1^*. In maxT, t_1 is compared to the highest test-statistic $t_{1,max}, \ldots, t_{B,max}$ obtained on B permutation of the trait values (see Figure 3.4). In minP, p_1^* is compared to the smallest marginal empirical p-values $p_{1,min}^*, \ldots, p_{B,min}^*$ obtained on B permutations of the trait values. As a consequence, in the traditional implementation of minP, two sets of permutations are used: one for computing the marginal empirical p-values $p_1^*, \ldots, p_m^*$ and one for computing the adjusted ones $p_1, \ldots, p_m$.

To compute p_2, the intuitive idea from Section 3.3 can be recycled: one can imagine that the case of (SNP_{l1}, SNP_{r1}) is closed and that this pair has been removed from the dataset. As a consequence, (SNP_{l2}, SNP_{r2}) is now the pair leading to the lowest marginal empirical p-value and the whole procedure can be restarted. This reasoning can be extended for $p_3, \ldots, p_m$. Again, this would be a waste of time and the actual procedure is illustrated in Figure 3.14. This implementation is based on the following three observations. First, note that the m^{th} marginal empirical p-value is guaranteed to be useful: when p_m is computed, (SNP_{lm}, SNP_{rm}) is the only pair remaining in the dataset and this value is readily the minimum. Second, note that the $(m-1)^{\text{th}}$ marginal empirical p-value is useful if and only if it is lower than the m^{th} one. Indeed, if it is lower it is at least the minimum in the computation of p_{m-1}, but if it is higher it cannot be the minimum in any computation. For this reason, in the latter case, the $(m-1)^{\text{th}}$ marginal empirical p-value is overwritten by the m^{th} one, without harming the algorithm. Third, note that this reasoning can be carry on for the remaining $(m-2)^{\text{th}}, \ldots, 1^{\text{st}}$ marginal empirical p-values. After this procedure, the second set of permutation vectors contains increasing values. This allows to compute any p_j value easily, by using the corresponding $p_{1,j}^*, \ldots, p_{B,j}^*$ marginal empirical p-values readily. As a final remark, note that a p_2 value lower than p_1 would be too optimistic, i.e. the p-value of a less significant pair should not be lower than the one of a more significant pair. For this reason, whenever $p_2 < p_1$, minP replaces p_2 by p_1 and so on for $p_3, \ldots, p_m$.

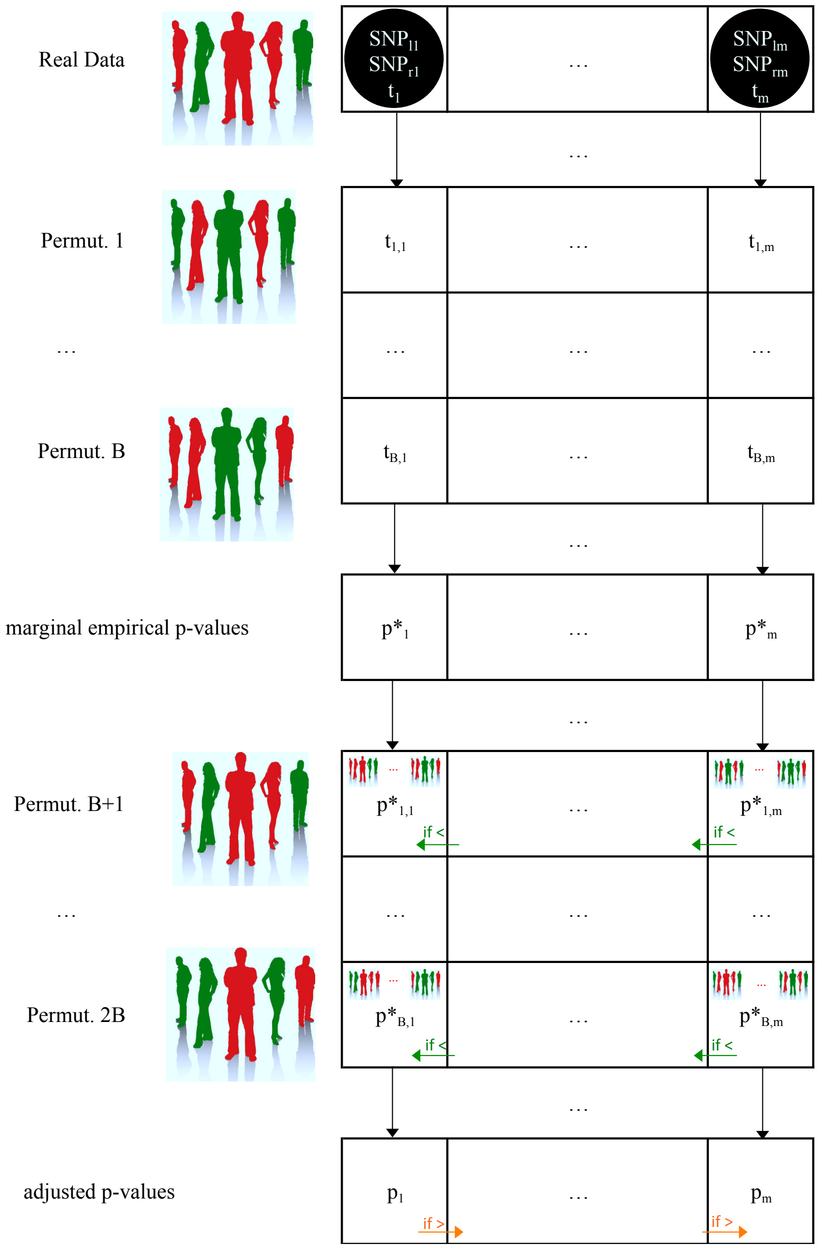


Figure 3.14: Classical minP. The estimation of the raw p_j^* values ($j = 1, \dots, m$) is based on B random permutations of the trait values ($i = 1, \dots, B$). Each time, the original t_j is compared to the $t_{i,j}$ values obtained on the permuted data and p_j^* estimated by $\frac{a_j+1}{B+1}$, with a_j the amount of $t_{i,j} \geq t_j$. Without loss of generality, we can assume $p_1^* \leq p_2^* \leq \dots p_m^*$ (the data can easily be reorganized). Another set of B random permutations of the trait values ($k = 1, \dots, B$) is used to estimate the raw $p_{k,j}^*$ values. Each time, a new $t_{k,j}$ value is computed (not represented in the figure), a temporary set of B permutations is generated ($l = 1, \dots, B$), temporary $t_{l,j}$ are computed and $p_{k,j}^*$ estimated by $\frac{b_j+1}{B+1}$, where b_j is the amount of $t_{l,j} \geq t_{k,j}$. Then, $p_{k,j}^*$ is overwritten by $p_{k,j+1}^*$ whenever $p_{k,j+1}^* < p_{k,j}^*, \forall k = 1, \dots, B, \forall j = m-1, \dots, 1$. Then, the adjusted p_j values are estimated by $\frac{c_j+1}{B+1}$, where c_j is the amount of $p_{k,j}^* \leq p_j^*$. Finally, p_{j+1} is overwritten by p_j whenever $p_j > p_{j+1}, \forall j = 1, \dots, m-1$.

A straightforward implementation of minP is given by the following five steps:

1) Generation of the *Real Data* vector R. This vector is created empty with a preallocated size of m and filled by performing a loop over the pairs of SNPs. For $j = 1, \ldots, m$: compute the test-statistic t_j corresponding to the j^{th} pair of SNPs and store an object containing the computed test-statistic and the names of the corresponding SNPs in R_j.

2) Generation of the test-statistics *Permutation* vectors $T_1, \ldots, T_B$. These vectors are created empty with a preallocated size of m and filled one by one. For $i = 1, \ldots, B$: permute randomly the trait values of the subjects, without modifying their genotypes and perform a nested loop over the pairs of SNPs. For $j = 1, \ldots, m$: compute the test-statistic $t_{i,j}$ corresponding to the j^{th} pair of SNPs and store it at the j^{th} index of the T_i vector.

3) Generation of the *Raw p-values* vector W. This vector is created empty with a preallocated size of m and filled by performing a loop over the pairs of SNPs. For $j = 1, \ldots, m$: compute the amount a_j of $t_{i,j} \geq t_j$ values $(i = 1, \ldots, B)$ and store the estimated p-value $\frac{a_j+1}{B+1}$ at the j^{th} index of vector W. Finally, sort vector W and ensure that vectors $R, T_1, \ldots T_m$ are rearranged accordingly.

4) Generation of the marginal empirical p-values *Permutation* vectors $V_1, \ldots, V_B$. These vectors are created empty with a preallocated size of m and filled one by one. For $k = 1, \ldots, B$: permute randomly the trait values of the subjects and perform a nested loop over the pairs of SNPs. For $j = 1, \ldots, m$: compute the test-statistic $t_{k,j}$ corresponding to the j^{th} pair of SNPs, for $l = 1, \ldots, m$ permute randomly the trait values of the subjects and compute the corresponding $t_{l,j}$ values, compute the amount b_j of $t_{l,j} \geq t_{k,j}$ values and store the estimated marginal empirical p-value $\frac{b_j+1}{B+1}$ at the j^{th} index of vector V_k. Force the monotonicity: $V_{k,j}$ is overwritten by $V_{k,j+1}$ whenever $V_{k,j+1} < V_{k,j}, \forall k = 1, \ldots, B, \forall j = m-1, \ldots, 1$.

5) Generation of the *adjusted p-values* vector P. This vector is again created empty with a preallocated size of m and filled by performing a loop over the pairs of SNPs. For $j = 1, \ldots, m$: compute the amount c_j of $V_{k,j} \leq W_j$ values $(k = 1, \ldots, B)$ and store the value $\frac{c_j+1}{B+1}$ at the j^{th} index of vector P. Force the monotonicity of vector P: for $j = 1, \ldots, m-1$ replace P_{j+1} by P_j if $P_{j+1} < P_j$.

Note that from a memory point of view, it is again best to implement the aforementioned algorithm in a slightly different way. The straightforward implementation implies all *Permutation* vectors of Figure 3.14 to be in memory at the same time, requiring $O(Bm)$ memory. A memory of $O(m)$ can be achieved by working columns by columns in Figure 3.14 instead of rows by rows. This idea has been refined by Ge et al [39] to avoid the double permutation procedure described above and make the algorithm faster. The complexity of the classical implementation is indeed given by $O(mB^2 + m \log m)$, whereas the one of Ge's implementation is given by $O(mB \log B + m \log m)$. The drawback of working columns-by-columns is obviously that in-place permutation of the trait values is not possible anymore.

3.5.2 Ge's implementation

The idea of this implementation is to avoid producing the embedded temporary sets of B permutations ($l = 1, \ldots, B$) from Figure 3.14 and as a consequence sidestep the computation of the temporary $t_{l,j}$ values which is at the origin of the $O(mB^2)$ factor in the complexity of the algorithm. To do so, the concept is to fill the *Permutation* vectors $B+1, \ldots, 2B$ from Figure 3.14 column by column instead of row by row and use a single set $t_{1,j}, \ldots, t_{B,j}$ of test-statistics to estimate all $p^*_{1,j}, \ldots, p^*_{B,j}$ values of a column, instead of B sets of test-statistics values as before.

This scheme does not on its own reduce the complexity, since it still implies B comparisons to compute the amount b_j of $t_{l,j} \geq t_{k,j}$ values in order to asses the mB raw $p^*_{k,j}$ values, i.e. still $O(mB^2)$ operations. However, the idea can be refined by sorting the $t_{1,j}, \ldots, t_{B,j}$ values, which takes only $O(B \log B)$ operations using the *quicksort* algorithm. Let's note $t_{s1,j} \geq \ldots \geq t_{sB,j}$ the result, which is obviously different from one column to the other. From a programming point of view, working like this implies to keep track of the actual index before sorting, to be able to tell which $t_{si,j}$ value corresponds to which permutation.

When there are no ties, the marginal empirical p-values are readily given by

$$p^*_{si,j} = \frac{i}{B} \qquad \forall i = 1, \ldots, B \tag{3.8}$$

When there are ties, the situation is a bit more complicated. Let's organized the values as follows:

$$t_{s1^1,j} = \ldots = t_{s1^{k_1},j} > t_{s2^1,j} = \ldots = t_{s2^{k_2},j} > \ldots > t_{sA^1,j} = \ldots = t_{sA^{k_A},j} \tag{3.9}$$

where A is the total amount of different values, k_1 the amount of equals top values, k_2 the amount of equals second highest values and so on. In this case, the marginal empirical p-values can be obtained by

$$p^*_{sa^1,j} = \ldots = p^*_{sa^k_a,j} = \frac{\sum_{x=1}^{a} k_x}{B} \qquad \forall a = 1, \ldots, A \tag{3.10}$$

The detailed implementation can be found in [39] and also optimizes the memory usage, to reach a space complexity of $O(m + B)$. However, this algorithm would produce a file with about 10^{12} results when performing a GWAIs. For this reason, we propose Van Lishout's implementation of minP, presented in the next section. This algorithm is based on the now usual idea to compute only the top n adjusted p-values, while still correcting for the m hypothesis tests performed. It can also be considered as a single permutation version of minP, since it recycles the idea of Ge's implementation avoiding the double permutation algorithm. The main benefit of this novelty is to reduce the memory usage, while the computing time complexity is still given by $O(mB \log B + m \log m)$.

3.5.3 Van Lishout's implementation

To save memory and also improve the computing time slightly, we propose Van Lishout's implementation of minP. It adapts Ge's implementation of minP to produce only the top n results. This contribution has not been published yet. Indeed, since maxT is known to be faster than minP, our attention was focused on the former in this thesis and finally lead to the new gammaMAXT algorithm. The following implementation is basically just the result of applying the core idea of Van Lishout's implementation of maxT to the case of minP. Anyway, recycling the ideas of Ge's implementation into this algorithm requires some additional care and this is certainly worth describing in details. Figure 3.15 illustrates the idea.

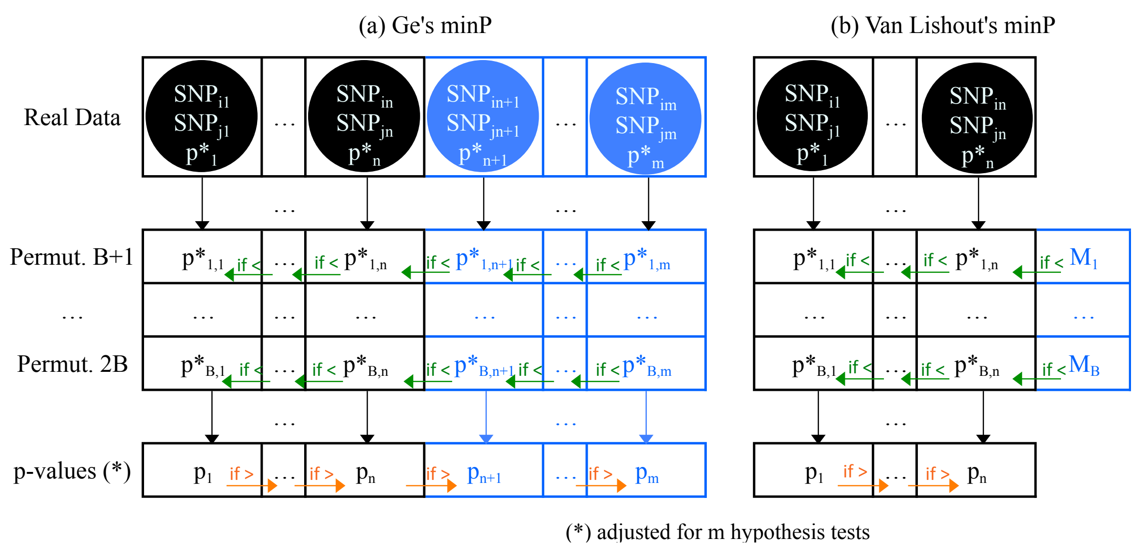

(a) Ge's minP
(b) Van Lishout's minP
Real Data
Permut. B+1
...
Permut. 2B
p-values (*)
SNP_i1 SNP_j1 p*_1
SNP_in SNP_jn p*_n
SNP_in+1 SNP_jn+1 p*_n+1
SNP_im SNP_jm p*_m
p*_1,1 if < ... if < p*_1,n if < p*_1,n+1 if < ... if < p*_1,m
p*_B,1 if < ... if < p*_B,n if < p*_B,n+1 if < ... if < p*_B,m
if < M_1
if < M_B
p_1 if > ... if > p_n if > p_n+1 ... if > p_m
(*) adjusted for m hypothesis tests

Figure 3.15: Ge's implementation of minP versus Van Lishout's one. In Ge's case, marginal empirical p-values are computed for all pairs of SNPs (in the same way as in Figure 3.2) and stored with the SNP names in the sorted *Real data* vector ($p^*_j \leq p^*_{j+1}, \forall j = 1, \ldots, m-1$). In Van Lishout's implementation, the same computations are performed, but only the top n marginal empirical p-values and corresponding SNP names are stored. In Ge's implementation, the $p^*_{i,j}$ values are computed column by column, from right to left, $\forall j = m, \ldots, 1$. Each time, the $p^*_{i,j}$ values that are smaller than the $p^*_{i,j+1}$ ones are overwritten by the latter $\forall i = 1, \ldots, B$ and the adjusted p-value p_j is computed by $\frac{a_j+1}{B+1}$, where a_j is the amount of $p^*_{i,j} \leq p^*_j$ values. Note that the $p^*_{1,j+1}, \ldots, p^*_{B,j+1}$ values can be removed from memory once p_j has been computed, $\forall j = m-1, \ldots, 1$. There are thus at most two columns in memory at the same time. In Van Lishout's implementation, in order to be able to recycle Ge's core idea, the $p^*_{i,j}$ values are also computed column by column, from right to left $\forall j = m, \ldots, 1$. However, each time, only the minimum M_i of the $[p^*_{i,n+1}, \ldots, p^*_{i,m}]$ is computed and stored, $\forall i = 1, \ldots, B$. Then, $p^*_{i,n}$ is replaced by M_i whenever $M_i < p^*_{i,n}$ and again $p^*_{i,j}$ replaced by $p^*_{i,j+1}$ whenever $p^*_{i,j+1} < p^*_{i,j}, \forall j = n-1, \ldots, 1$. Finally, p_{j+1} is overwritten by p_j whenever $p_j > p_j + 1$, $\forall j = 1 \ldots m-1$ in the classical case and $\forall j = 1 \ldots n-1$ in Van Lishout's implementation. It is easy to prove that the final $p_1, \ldots, p_n$ adjusted p-values are the same with both implementations of minP.

Van Lishout's implementation of minP exploits ideas from the maxT section and Ge's idea to avoid the double permutation algorithm. The different steps are given by:

1) Generation of the sorted *Real Data* vector R. Beforehand, store B permutations of the trait values. The R vector is created empty with a preallocated size of n and initialized by performing a loop over the first n pairs of SNPs. For $j = 1, \ldots, n$: compute the marginal empirical p-value p_j^* corresponding to the j^{th} pair of SNPs[8] and store an object containing p_j^* and the names of the corresponding SNPs in R_j. After this loop, use *quicksort* to sort the vector in increasing marginal empirical p-values order. Then, perform a loop over the remaining $m-n$ pairs of SNPs. For $j = n+1, \ldots, m$: compute the marginal empirical p-value p_j^* corresponding to the j^{th} pair of SNPs. If it is lower than the value stored in R_n, overwrite R_n with R_{n-1}. If it is also lower than the value stored in R_{n-1}, overwrite R_{n-1} with R_{n-2}. Continue this process until a lower value than p_j^* is reached (or the beginning of R). Then create a new object containing p_j^* and the corresponding SNP names and store it in the one but last visited cell of the R vector (or the first). Let's note $p_1^* \leq p_2^* \leq \ldots \leq p_m^*$ the marginal empirical p-values obtained after the loop and $(SNP_{l1}, SNP_{r1}), \ldots, (SNP_{lm}, SNP_{rm})$ the corresponding pairs of SNPs.

2) Generation of the *Minimum* vector M. This vector is created with a preallocated size of $B+1$, initialized with 1's[9] and build up by performing a loop over the pair of SNPs not belonging to the top n. For $j = n+1, \ldots, m$: create a vector T of size $B+1$ and construct it in three steps. First, for $i = 1, \ldots, B$: set the trait values to the ones of the i^{th} permutation, compute the test-statistic value $t_{i,j}$ and store an object $(t_{i,j}, i)$ in T_i. Second, compute the test-statistic t_j corresponding to the real data and store an object $(t_j, B+1)$ in T_{B+1}. Third, sort vector T in decreasing test-statistics order using the *quicksort* algorithm. Once T has been constructed, Ge's idea from equations 3.8 and 3.10 can be recycled to update M. First suppose that there are no ties. In this case, the marginal empirical p-value associated to any element T_c is readily given by $\frac{c}{B+1}$. The index i stored in T_c gives us the element of M to which this marginal empirical p-value should be compared to. If $\frac{c}{B+1} < M_i$ a new minimum has been found and M_i needs to be overwritten by that minimum. Second, suppose that there are ties. In this case, the marginal empirical p-value associated to any element T_c is given by $\frac{c'}{B+1}$, where c' is the highest number for which the test-statistics stored in T_c and $T_{c'}$ are equals. Since in practice, ties are possible, the best way to implement the update of M in linear time is to loop over the element of T from the right to the left. In this way, a variable can be initialized to $B+1$ and set to the index of the currently visited cell of T if and only if this cell contains an higher test-statistic value than its right neighbor. In this way, the variable always contains the correct numerator of the marginal empirical p-value corresponding to the currently visited cell of T.

[8] To obtain p_j^* proceed in three steps. First, compute the test-statistic value t_j corresponding to the j^{th} pair of SNPs on the real data and initialize a variable a_j to 1. Second, for $i = 1, \ldots, B$: set the trait values to the ones of the i^{th} permutation, compute $t_{i,j}$ and if $t_{i,j} \geq t_j$ increment the variable a_j by one. Third, divide a_j by $B+1$ to obtain p_j^*. Note that in practice, for efficiency reasons, the division does actually occur at the end of the algorithm. Conceptually, we wrote that it occurs here in order to be able to call the content a (correctly defined) probability.

[9] Note that in practice it is filled with $B+1$ values, since the aforementioned division by $B+1$ has been postponed to the end of the algorithm.

3) Generation of the *adjusted p-values* vector P. This vector is created empty with a preallocated size of n. For $j = n, \dots, 1$: create a vector T of size $B + 1$ and construct it in the exact same way as at step 2. Update the *Minimum* vector M as in step 2. After that, count the amount a_j of elements of M lower or equal than R_j. The j^{th} adjusted p-value is readily given by $\frac{a_j}{B+1}$. Finally, enforce the monotonicity of P. For $j = 1, \dots, n-1$: if $P_{j+1} < P_j$ overwrite P_{j+1} by P_j.

Box 3.5. Van Lishout's implementation of minP

(1) Generate B permutations of the trait values and store them in memory.

(2) If $m < n$ set $n = m$. Create empty vector R of size n. For $j = 1, \dots, n$:

- (a) Set the trait value to the original trait ones and compute t_j.
- (b) Define $a_j = 1$. For $i = 1, \dots, B$: set the trait values to the ones of the i^{th} permutation, compute $t_{i,j}$ and if $(t_{i,j} \geq t_j)$ do a_j++.
- (c) Store an object (a_j, SNP_{lj}, SNP_{rj}) in R_j. We note $(R_j)_1$ the 1st element.

(3) Sort R by increasing a_j values. After that, for $j = n+1, \dots, m$ do:

- (a) Set the trait values to the original ones and compute t_j.
- (b) Define $a_j = 1$. For $i = 1, \dots, B$: set the trait values to the ones of the i^{th} permutation, compute $t_{i,j}$ and if $(t_{i,j} \geq t_j)$ do a_j++.
- (c) If $a_j < (R_n)_1$ do {$k = n$; while ($k > 0$ and $a_j < (R_{k-1})_1$) do $R_k = R_{k-1}$ and k−− ; store an object (a_j, SNP_{lj}, SNP_{rj}) in R_k.}

(4) Create a hash table containing an entry for each element of R.

(5) Create a vector M of size $B+1$. Initialize it with $(B+1)$'s. For $j = 1, \dots, m$:

- (a) If the j^{th} pair of SNPs is in the hash table, ignore steps (b) and (c).
- (b) Create a vector T of size $B+1$. For $i = 1, \dots, B$: set the trait values to the ones of the i^{th} permutation, compute $t_{i,j}$ and store an object ($t_{i,j}$, i) in T_i. Set the trait to the original values, compute t_j and store an object (t_j, B+1) in T_{B+1}. Sort T in decreasing test-statistics order.
- (c) Set $c = B+1$. For $p = B-1, \dots, 0$: do {if $((T_p)_1 \neq (T_{p+1})_1)$ do c=p+1 ; if ($M_{(T_p)_2} >$ c) do $M_{(T_p)_2} =$ c}.

(6) Create empty vector P of size n. For $j = n, \dots, 1$:

- (a) Create a vector T of size $B+1$. For $i = 1, \dots, B$: set the trait values to the ones of the i^{th} permutation, compute $t_{i,j}$ and store an object ($t_{i,j}$, i) in T_i. Set the trait to the original values, compute t_j and store an object (t_j, B+1) in T_{B+1}. Sort T in decreasing test-statistics order.
- (b) Set $c = B+1$. For $p = B-1, \dots, 0$: do {if $((T_p)_1 \neq (T_{p+1})_1)$ do c=p+1 ; if ($M_{(T_p)_2} >$ c) do $M_{(T_p)_2} =$ c}.
- (c) Set $a_j = 0$. For $p = 1, \dots, B$: if $((M_p)_1 \leq (R_j)_1)$ do a_j++. Set $P_j = a_j$.

(7) For $j = 1, \dots, n$: divide P_j by $B+1$. For $j = 1, \dots, n-1$: replace P_{j+1} by P_j if $P_{j+1} < P_j$.

The final algorithm is given in Box 3.5 and is trivial to adapt for users interested in performing a main effects analysis. This implementation needs only a minuscule adaptation: at steps (2)(c) and (3)(c), an object $(a_j,\ SNP_j)$ should now be stored instead of the $(a_j,\ SNP_{lj},\ SNP_{rj})$ one. It comes without saying that the variable m represents the number of SNPs here and not the number of pairs anymore. Similarly, the code of Box 3.5 is easy to adapt for three-order interactions: at step (2)(c) and (3)(c), an object $(a_j,\ SNP_{lj},\ SNP_{mj},\ SNP_{rj})$ should be stored instead of the $(a_j,\ SNP_{lj},\ SNP_{rj})$ one and m now represents the number of triplets of SNPs.

3.6 Discussion

This chapter contains the two main contributions of this PhD thesis: Van Lishout's implementation of maxT and gammaMAXT. The figures from this chapter are not taken from the literature, they are all original[10]. All algorithms have been described in the context of GWAIs. However, the proposed algorithms and especially gammaMAXT, offer a general significance assessment and multiple testing approach, applicable to any context that requires performing hundreds of thousands of tests. It offers new perspectives for fast and efficient permutation-based significance assessment in large-scale (integrated) omics studies.

The reader may be wondering why there is no gammaMINP contribution, similar to the gammaMAXT one. There are several issues in adapting the gammaMAXT idea to the minP case. The probably most obvious one is that the minimum cannot follow a gamma distribution. Indeed, in the context of maxT, test-statistic values are sampled and supposed to follow a mixture distribution whose tail is modeled by a shifted gamma distribution. This idea works well because the test-statistic are expressed on continuous scale. In the case of minP, the sampled values would be the marginal empirical p-values instead of test-statistics. Since marginal empirical p-values are expressed as multiples of $\frac{1}{B+1}$, there are in fact only $B + 1$ possible values and the scale is now a discrete one. This obviously requires a different approach than what was done in the context of the gammaMAXT algorithm.

In Chapters 2 and 3, `mbmdr-4.4.1.out` is mostly presented in a basic setting, where the user investigates all available pairs of SNPs. However, the software also contains nice options to do customized analysis:

- The -f option allows users to restrain the attention to the pairs composed of exactly one marker from the given comma-separated list of marker names and one marker that is not in this list. This option is particularly useful for instance if the users wants to perform a GxE analysis. By passing the names of the environmental factors as argument to this option, the software automatically investigates all possible combination of one SNP and one environmental factor.

- The -F option is similar to the -f one, except that the markers are split into two input files. The software analyzes only the pairs composed of exactly one marker from the first input file and one marker from the second one. This option is useful for instance when we test SNPs from two different genes, we usually need to assess only intragenic associations and do not need intergenic ones.

- The -e and -E options allows to ignore some markers in the analysis, by either passing their names in a comma-separated list (-e option) or a file containing one marker name per line (-E option). This enables users to work with one big file containing all the available information and remove some information when doing some particular analysis. For instance, ignore the environmental factors for a while.

[10] Except Figure 3.1 which is obviously just an illustrative picture

- The -k and -K options are the mirror options from the -E and -E ones. This time, the users indicates the markers that he wants to keep instead of the ones he ones to get rid of. This option is not necessary from a conceptual point of view, but useful in practice to avoid long comma-separated lists in some scenarios.
- The -s option allows to perform the second stage of a two-stage analysis. The idea is to run a first analysis as usual without taking care of the fact that a second stage is planned. Then, once this first analysis is finished, use the output file that has been obtained as a list of interesting pairs of SNPs to investigate in the second round. The second stage can then be performed on another dataset, enclosing the same SNPs and environmental factors as at stage one, but containing data that is independent from the first file (for instance data coming from another hospital). This strategy is in fact called a discovery-replication analysis and is described in more details in Chapter 4. The -s option allows to pass the output file from the first stage as argument when performing the second stage. At stage two, only the pairs of SNPs included in the output file from stage one are investigated. Since the amount of pairs of SNPs remaining at stage two is typically very low, this stage is very quick from a computing time point of view, allowing to use more advanced algorithms like the ones presented in the next chapter.

Note also that the software contains automatic procedures to control the data integrity and if necessary correct it. For instance, a SNP with no variation at all could lead to a degenerated HLO matrix containing only a single group of individuals. To avoid this, such SNPs are detected just after the reading of the data is done and removed from the list of SNPs to investigate. Indeed, a variable with no variation at all cannot provide any information on the disease. Removing such variables from scratch allows to consider in the rest of the software that such a scenario cannot occur, facilitating the optimization of the source code.

Chapter 4

Population stratification and covariate adjustment

4.1 Outline

The algorithms proposed so far have been evaluated and discussed in the absence of population stratification. The latter refers to genetic similarities between individuals on the basis of shared genetic ancestry. This leads to a situation in which the population of interest includes subgroups of individuals that are on average more related to each other than to other members of the wider population [4]. Population stratification is in fact just a particular case of a confounding factor, as illustrated in Figure 4.1.

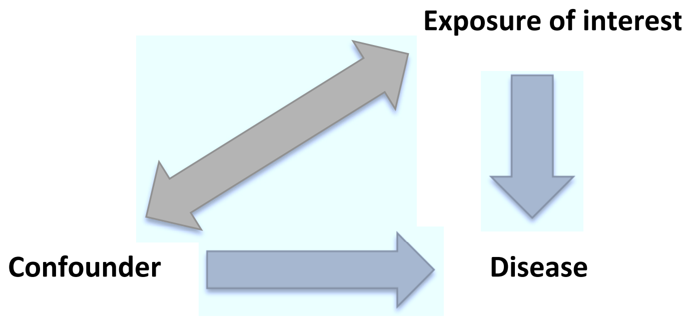


Figure 4.1: A confounder is correlated with the exposure of interest and is a risk factor even for the people that are not exposed by the exposure of interest. In the case of population stratification, the exposure of interest is the genotype. It is correlated with the confounder because both are correlated with ethnicity. If nothing is done to correct for population stratification, the genotype can be incorrectly regarded as associated with risk of disease.

The three conditions for a factor to be a confounder are the following:

- The confounder should associate with exposure, i.e. it should have imbalance distribution between the exposed and the non-exposed groups. For instance, age is a confounder if the distribution of age differs between exposed and non-exposed subjects.
- The confounding variable should be an independent risk factor for disease of interest. This inherent association must be present in both the exposed and the non-exposed groups.

- The association of confounder and disease should not be resulted via exposure, i.e. this association should not be an intermediate pathway relation between exposure and outcome.

If any of these three conditions is not satisfied, the mixing effect with exposure will not occur and the third variable is no longer a confounder. Figure 4.2 gives an example of how ignoring this issue can create an association between genetics and disease, even on simulated data clearly containing no such association.

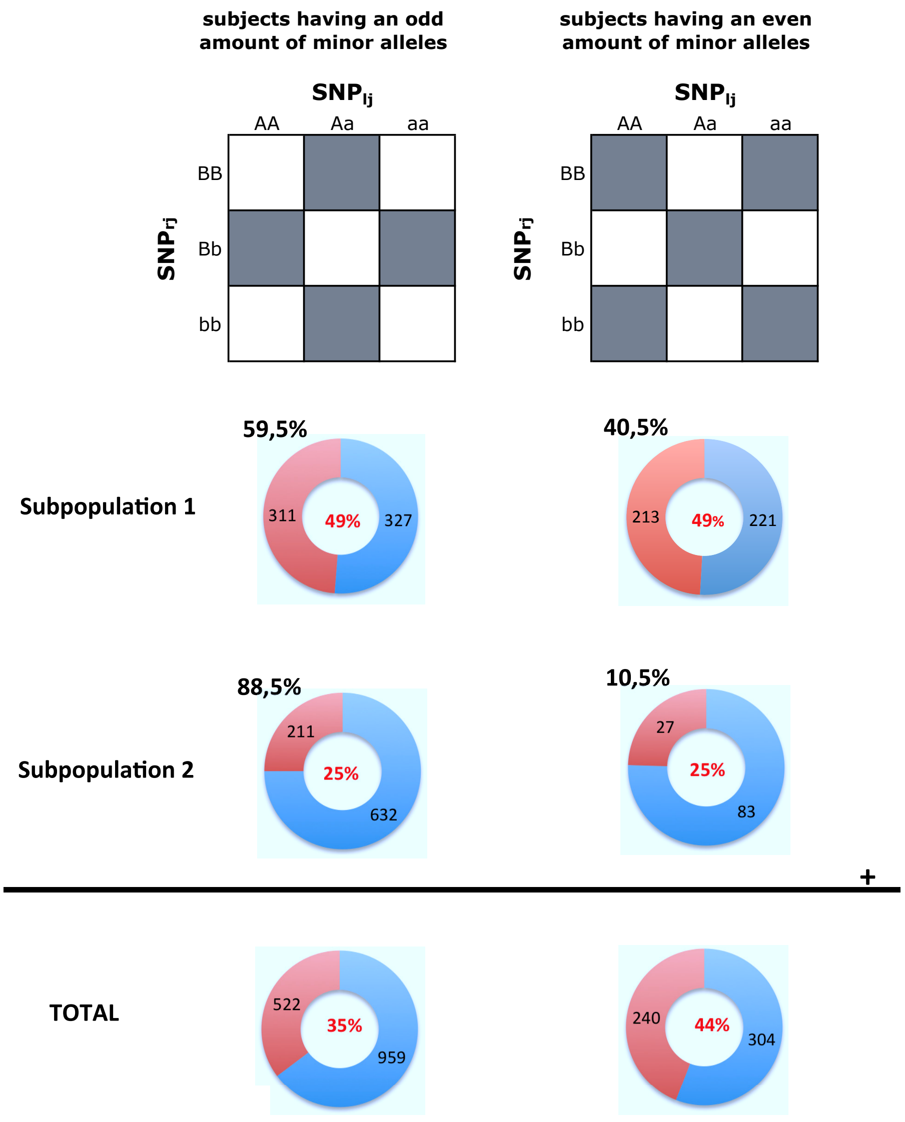


Figure 4.2: Suppose that a particular pair of SNPs has truly no effect on a disease, but that the latter is more prevalent in subpopulation 1 than 2 (49% of the people from subpopulation 1 are affected vs 25% in subpopulation 2, regardless of the genotype). Suppose that the rate of people having an odd amount of minor alleles is significantly lower in subpopulation 1 than 2 (59,5% vs 88,5%). Not taking the confounder into account would wrongly lead to conclude that having an odd amount of minor alleles decreases susceptibility to disease, since this is what is observed globally (35% of the people having an odd amount of minor alleles are affected in the dataset vs 44% for the people having an even amount).

This example is in fact an instance of Simpson's paradox [107]. In the context of GWAS, robust methods exists to correct for population stratification [51, 4, 76, 119]. Furthermore, confounding can potentially be dealt with by choosing another design than for instance case/control. Family-based strategies such as TDT (transmission disequilibrium test) [111], SDT (sibship disequilibrium test) [52] and FBAT (family based association test) [53] naturally guard against population stratification. In general, the effect of population stratification is well understood in the context of main effects, but not yet in the context of interactions [43]. In this prospect, mixed designs is a particularly interesting path to investigate, for instance in multi-center studies that selected different study designs [139].

In this thesis, we propose a new viewpoint. In Section 4.2, we introduce two new algorithms called STRAT1 and STRAT2 to automatically correct for population stratification in GWAIs. These algorithms are in fact more general, since they also correct for any other unmeasured confounding factors. In Section 4.3, we propose methods to handle known covariates (for instance age, gender, body mass index, ...) that may or may not be linked with the disease (or outcome).

In Chapter 3, the performances of the different algorithms have been evaluated on simulated data. In this chapter, taking the same approach would be problematic. Indeed, there is an increasing evidence that populations can be differentiated from each other by non-linear (rather than linear alone) genetic patterns [1]. Generating realistic synthetic data is far from trivial, since inducing the situation of confounding implies creating scenarios in which non-linear patterns are distributed differently between the populations and at the same time generating different disease prevalence between the populations. To our knowledge, no such tool exist and creating a simulation tool ourselves would represent a research on its own, out of the scope of this PhD thesis. Note that a method outside of the classic regression framework has been proposed in [90], in the context of gene-gene interactions. However, the data is simulated based on singular SNPs and differential genotype distributions (at a singular SNP level) between populations. Hence, it does not take into account joint SNP patterns and multilocus genotype distributions that may be different between populations.

For this reason, we validate the methods presented in this chapter on a real-life dataset that we have already used intensively in our research group. Since the data is well-known, sensible or insensible results should immediately be spotted. We use extensive data of the international Inflammatory Bowel Disease Genetic Consortium (IIBDGC) from 15 European countries typed on Immunochip. Inflammatory Bowel Disease describes a group of disorders in which the intestines become inflamed, affecting about 1 in 250 persons in Western Europe, North America and Australia. The data is described in more details in Section 4.2.2.

4.2 Correcting for unmeasured confounding factors

The root idea of the methods of this section is to adapt the maxT algorithm, to take account of the fact that an association between some pairs of SNPs and the trait may be confounded by unmeasured factors. As an example, suppose that no SNP pair is truly linked with the disease (complete null hypothesis) and that some SNP pairs are concerned with population stratification. The latter pairs will follow another distribution on the observed data than on the permuted one (permuting the trait values cuts the link between the confounder and the disease). As a consequence, the test-statistics corresponding to the SNP pairs concerned with population stratification will be higher on average on the observed data than on the permuted one, leading to lower p-values than it should and an important increase of the FWER. A classical way to handle such a situation in GWAs is to define an inflation factor λ, by which the observed test statistics can be adjusted [25]. The inflation factor is test dependent and defined by the ratio of the median of the empirically observed distribution of the test statistic to the expected median in case of a 2df test[4, 141]. This idea can be adapted to GWAIs and leads to the STRAT1 algorithm, which we describe in the next section.

4.2.1 STRAT1 algorithm

STRAT1 is a simple algorithm for correcting automatically for unmeasured confounding factors. The basic idea is illustrated in Figure 4.3 and can be decomposed as:

1) Generation of the sorted *Real Data* vector R. This vector is created empty with a preallocated size of m and filled by performing a loop over the pairs of SNPs. For $j = 1, \dots, m$: compute the test-statistic t_j corresponding to the j^{th} pair of SNPs and store an object containing the computed test-statistic and the names of the corresponding SNPs in R_j. After this loop, sort the vector in decreasing test-statistic order using the *quicksort* algorithm. Without loss of generality, let's note $t_1 \geq t_2 \geq \dots \geq t_m$ these test-statistics and $(SNP_{l1}, SNP_{r1}), \dots, (SNP_{lm}, SNP_{rm})$ the corresponding pairs of SNPs. Compute the median M_1 of the test-statistics stored in R.

2) Generation of the test-statistics *Permutation* vectors $T_1, \dots, T_B$, created empty with a preallocated size of m and filled one by one. For $i = 1, \dots, B$: permute randomly the trait values, without modifying the genotypes and perform a nested loop over the pairs of SNPs. For $j = 1, \dots, m$: compute the test-statistic $t_{i,j}$ corresponding to the j^{th} pair of SNPs and store it at the j^{th} index of the T_i vector. Compute the median M_2 of all the numbers stored in $T_1, \dots, T_B$. Force the monotonicity of the rows: for $i = 1, \dots, B$ perform a nested loop over the pairs of SNPs. For $j = m-1, \dots, 1$ replace $T_{i,j}$ by $T_{i,j+1}$ if $T_{i,j} < T_{i,j+1}$.

3) Inflation of the values of the *Real Data* vector R. Compute the inflation factor $\lambda = \frac{M_1}{M_2}$. For $j = 1, \dots, m$: divide the test-statistic value t_j stored in R_j by λ.

4) Generation of the *adjusted p-values* vector P. This vector is again created empty with a preallocated size of m and filled by performing a loop over the pairs of SNPs. For $j = 1, \dots, m$: compute the amount a_j of $T_{i,j} \geq t_j$ values $(i = 1, \dots, B)$ and store the value $\frac{a_j+1}{B+1}$ at the j^{th} index of vector P. Force the monotonicity of vector P: for $j = 1, \dots, m-1$ replace P_{j+1} by P_j if $P_{j+1} < P_j$.

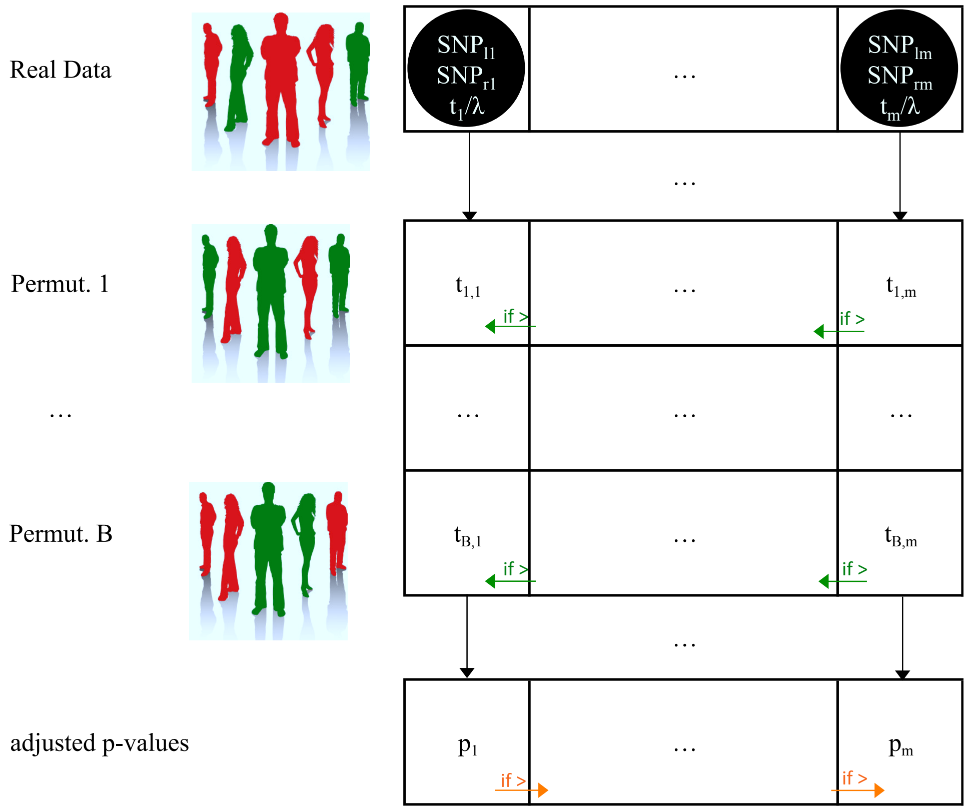


Figure 4.3: The basic idea of STRAT1 is the same as the one of the maxT algorithm described in Figure 3.5, except that the observed test-statistic values from the *Real data* vector are divided by the inflation factor λ. The latter is computed as $\frac{M_1}{M_2}$, where M_1 is the median of all the observed test-statistic values $t_1, \ldots, t_m$ and M_2 the median of all the test values obtained for the B permutations $t_{1,1}, \ldots, t_{B,m}$ (before the monotonicity enforcing step illustrated by green arrows was applied).

This basic idea is very straightforward. Nonetheless, some ameliorations need to be made to this initial idea, to make it work in practice, as we will motivate next:

1) In the previous chapter, we showed that MB-MDR final test statistics follow a mixture distribution of a shifted gamma distribution and a point mass at zero. Furthermore, Table 3.7 has shown that in practice, the parameter π denoting the proportion of strictly positive values is less than 50%. Hence, the zero values represent the majority of the test-statistic values. As a consequence, the median M_2 is equal to zero and the inflation factor cannot be defined by $\frac{M_1}{M_2}$, since this would imply a division by zero. The solution proposed in STRAT1 is to consider only the strictly positive numbers when computing M_1 and M_2.

2) The purpose of Van Lishout's implementation of maxT is mainly to save memory (to reach a memory usage independent from the number of SNPs instead of raising quadratically with it). Here, this idea cannot be recycled since all test-statistic values need to be stored in order to compute the median M_2. Analyzing a dataset composed of ten thousand SNPs would for instance require hundreds of Gigabytes. The solution proposed in STRAT1 is to use the algorithm for fast estimation of the median in linear time and constant space, as proposed in [30]. This algorithm is given in box 4.1.

Box 4.1. Fast Algorithm for Median Estimation

```
(1) Initialize $M_2$ to the first available value.
(2) Initialize $step$ to $\frac{M_2}{2}$ (or to a predefined minimal initial step $b$ if $\frac{M_2}{2} < b$)
(3) For each new value $v$:
    i. If $M_2 > v$ do ($M_2 = M_2 - step$) else if $M_2 < v$ do ($M_2 = M_2 + step$)
    ii. If ($|M_2 - v| < step$) do ($step = step/2$)
```

3) Computing time is the major issue, as was the case for the classical maxT implementation on which the basic idea of STRAT1 is based. The solution proposed in STRAT1 is to restrain our attention to the top one million pairs of SNPs leading to a strictly positive test-statistic value on the real data (if there are less than one million, focus on all available such pairs). This idea is similar to filtering the data, in order to focus on the most promizing pairs of SNPs. The drawback is that such a filtering increases the false positive rate significantly when the amount of dropped pairs is important, as discussed in details in the next paragraph.

The final version of STRAT1 is reported in Box 4.2. and illustrated in Figure 4.4.

Box 4.2. STRAT1 algorithm

```
(1) Create a vector $R$. For $j = 1, \ldots, m$: compute $t_j$ and if $t_j > 0$ push an object ($t_j$, $SNP_{lj}$, $SNP_{rj}$) at the back of $R$. We note $(R_k)_1$ the $k^{\text{th}}$ test-statistic.
(2) Sort $R$ in decreasing test-statistics order using the quicksort algorithm. Resize $R$ in order to keep only the top $10^6$ elements. Let $f$ be the final size of $R$.
(3) Find the median $M_1$. If $f$ is an odd number, $M_1$ is the test-statistic stored in $R_{\lceil \frac{f}{2} \rceil}$, otherwise the average of the test-statistics stored in $R_{\frac{f}{2}}$ and $R_{\frac{f}{2}+1}$.
(4) Generate $B$ permutations of the trait values and store them in memory.
(5) Find the median $M_2$. For $j = 1, \ldots, f$: do a nested loop over the permutations. For $i = 1, \ldots, B$: set the trait values to the ones of the $i^{\text{th}}$ permutation, compute $t_{i,j}$ and if $t_{i,j} \geq 0$ update $M_2$: if $t_{i,j}$ is the first encountered value execute steps (1) and (2) of Box 4.1, otherwise step (3).
(6) Define $\lambda = \frac{M_1}{M_2}$. For $j = 1, \ldots, f$: divide $(R_j)_1$ by $\lambda$.
(7) Create empty vectors $T$ and $P$ of size $f$. Fill $P$ with 1's. For $i = 1, \ldots, B$:
    (a) Set the trait values to the ones of the $i^{\text{th}}$ permutation.
    (b) For $j = 1, \ldots, f$: compute $t_{i,j}$ and store it in $T_j$.
    (c) For $j = f - 1, \ldots, 1$: replace $T_j$ by $T_{j+1}$ if $T_j < T_{j+1}$.
    (d) For $j = 1, \ldots, f$: increment $P_j$ by one if $T_j \geq (R_j)_1$.
(8) For $j = 1, \ldots, f$: divide $P_j$ by $B + 1$. For $j = 1, \ldots, f - 1$: replace $P_{j+1}$ by $P_j$ if $P_{j+1} < P_j$.
```

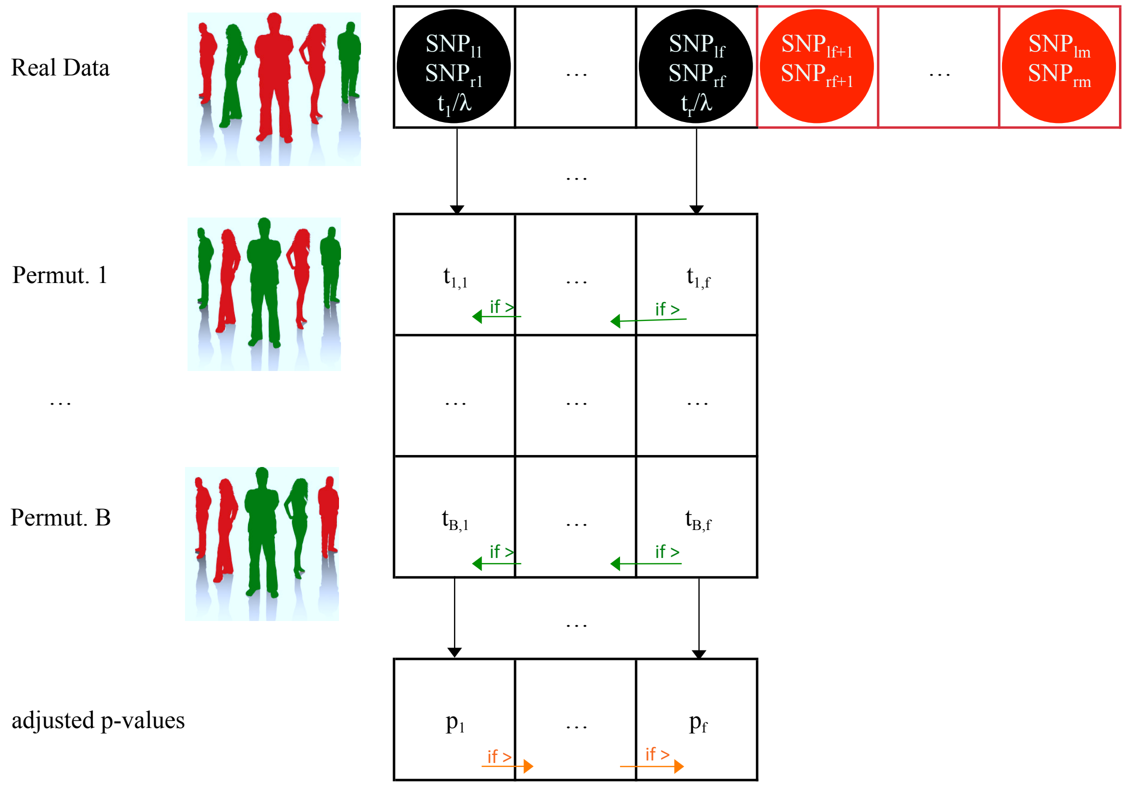


Figure 4.4: In the practical implementation of STRAT1, test-statistics are computed for all pairs of SNPs, but only the strictly positive ones are stored in the sorted Real Data vector, whose size is limited to one million elements. Let f be the final size of the vector. The pairs of SNPs that are given up are not stored and are represented in red in the Figure. These pairs of SNPs are not considered at all in the permutation vectors anymore, in opposition to the maxT algorithm.

In maxT and gammaMAXT, the multiple testing problem is handled properly, in the sense that the test statistics of all SNP pairs are taken into account both while constructing the *Real data* vector and each of the *Permutation* ones (directly or by prediction). This is also the case for the basic idea of STRAT1, but not for the practical implementation resulting from the three aforementioned ameliorations, required to make the algorithm work in practice in our particular context. Indeed, in the latter case, all pairs of SNPs are still considered while constructing the *Real data* vector, but only the ones leading to a strictly positive test-statistic value, limited to a maximum of one million pairs of SNPs, are kept. The pairs that have been dropped are no longer considered when constructing the *Permutation* vectors. As a consequence, when the complete null hypothesis is true and there are no unmeasured confounding factors, the maximums of the *Permutation* vectors may be smaller than the maximum of the *Real data* vector, just because the former vectors are smaller than the latter one and all test-statistics may follow the same distribution. This is expected to lead to an increase of the false-positive rate. We show in section 4.2.2 that the practical implementation of the STRAT2 algorithm may also suffer from an increase of the false-positive rate for similar reasons. A practical way to improve the situation is proposed in Section 4.2.2, in the real-life data analysis used to validate the STRAT2 algorithm.

The work on STRAT1 and STRAT2 has been accepted for poster presentation at the *ASHG 2013* conference in Boston [121]. This work was done just before the work on the gammaMAXT algorithm started. This is the practical reason why no ideas from the latter has been injected into STRAT1 and STRAT2, in order to try to speed up the algorithm while simultaneously controlling the FWER. However, there is also a theoretical reason. Indeed, the core idea of gammaMAXT is to compute only a small percentage of the test-statistics and predict what would happen if all were actually computed, based on the fact that they follow a mixture distribution that can be fitted. In the context of unmeasured confounding factors, the test-statistics follow different distributions and the whole idea falls apart.

In STRAT1, the same inflation factor is used for all pairs of SNPs, despite the fact that they may all follow different distributions. In this sense, STRAT1 is a bit too basic, consisting simply in increasing all p-values in order to reduce the false-positive rate, at the cost of some power loss. This algorithm is therefore not interesting to test on the real-life data, since it would lead to the exact same ranking as the maxT algorithm (only the p-values would be different). The idea of STRAT2 is to use an inflation factor that is specific to each pair of SNPs tested for association, a similar idea as what was done in the context of GWAS [132]. Due to the individual inflation factors, this algorithm may produce a very different ranking of the pairs of SNPs than maxT and is thus very interesting to test on real-life data.

4.2.2 STRAT2 algorithm

STRAT2 is an algorithm for correcting automatically for unmeasured confounding factors, taking into account the fact that the pairs of SNPs may follow different distributions and require specific inflation factors. The basic idea is illustrated in Figure 4.5 and can be decomposed as:

1) Generation of the sorted *Real Data* vector R. This vector is created empty with a preallocated size of m and filled by performing a loop over the pairs of SNPs. For $j = 1, \dots, m$: compute the test-statistic t_j corresponding to the j^{th} pair of SNPs and store an object containing the computed test-statistic and the names of the corresponding SNPs in R_j. After this loop, sort the vector in decreasing test-statistic order using the *quicksort* algorithm. Without loss of generality, let's note $t_1 \geq t_2 \geq \dots \geq t_m$ these test-statistics and $(SNP_{l1}, SNP_{r1}), \dots, (SNP_{lm}, SNP_{rm})$ the corresponding pairs of SNPs. Compute the median M_1 of the test-statistics stored in R.

2) Generation of the test-statistics *Permutation* vectors $T_1, \dots, T_B$. These vectors are created empty with a preallocated size of m and filled one by one. For $i = 1, \dots, B$: permute randomly the trait values of the subjects, without modifying their genotypes and perform a nested loop over the pairs of SNPs. For $j = 1, \dots, m$: compute the test-statistic $t_{i,j}$ corresponding to the j^{th} pair of SNPs and store it at the j^{th} index of the T_i vector. For $j = 1, \dots, m$: compute the median $M_{2,j}$ of the test-statistics stored in $(T_1)_j, \dots, (T_B)_j$. Force the monotonicity of the rows: for $i = 1, \dots, B$ perform a nested loop over the pairs of SNPs. For $j = m - 1, \dots, 1$ replace $T_{i,j}$ by $T_{i,j+1}$ if $T_{i,j} < T_{i,j+1}$.

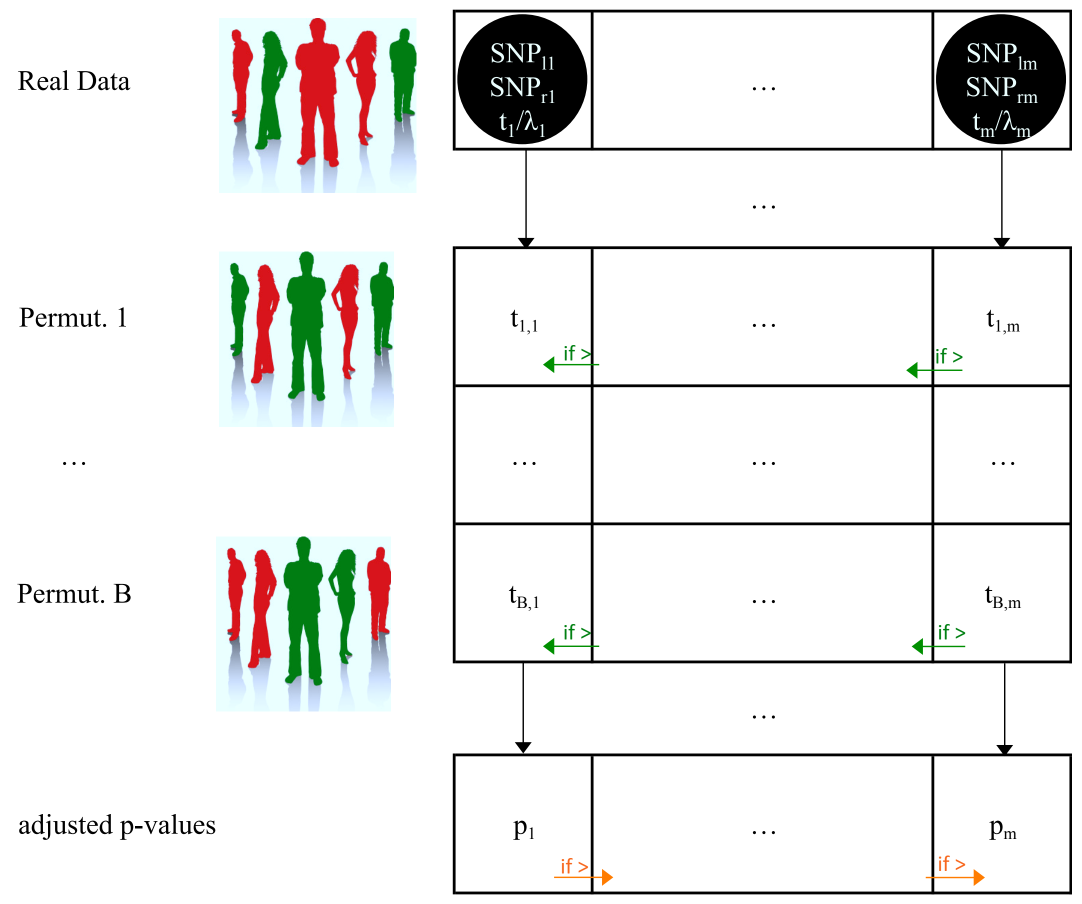


Figure 4.5: The basic idea of STRAT2 is the same as the one of STRAT1, except that the observed test-statistic values from the *Real data* vector are divided by individual inflation factors $\lambda_1, \ldots, \lambda_m$. The latter are obtained by $\lambda_j = \frac{M_1}{M_{2,j}}, \forall j = 1 \ldots m$, where M_1 is still the median of all the observed test-statistic values $t_1, \ldots, t_m$ and $M_{2,j}$ the median of the test-statistic computed on the j^{th} pair $t_{1,j}, \ldots, t_{B,j}$ (before the monotonicity enforcing step illustrated by green arrows was applied).

3) Inflation of the values of the *Real Data* vector R. For $j = 1, \ldots, m$: divide the test-statistic value t_j stored in R_j by $\lambda_j = \frac{M_1}{M_{2,j}}$.

4) Generation of the *adjusted p-values* vector P. This vector is again created empty with a preallocated size of m and filled by performing a loop over the pairs of SNPs. For $j = 1, \ldots, m$: compute the amount a_j of $T_{i,j} \geq t_j$ values $(i = 1, \ldots, B)$ and store the value $\frac{a_j+1}{B+1}$ at the j^{th} index of vector P. Force the monotonicity of vector P: for $j = 1, \ldots, m-1$ replace P_{j+1} by P_j if $P_{j+1} < P_j$.

Several accommodations are again needed to make this algorithm work in practice:

1) Since zero test-statistics represents the majority in practice, the medians $M_{2,j}$ are all equal to zero and the inflation factors cannot be defined by $\frac{M_1}{M_{2,j}}$, since this would imply a division by zero. The solution proposed in STRAT2 is to consider only the strictly positive numbers when computing the M_1 and $M_{2,j}$ values.

2) For the same reason as for STRAT1, the ideas of Van Lishout's implementation of maxT to save memory cannot be recycled here since all test-statistic values need to be stored in order to compute the medians $M_{2,j}$. The solution is again to use the algorithm for fast estimation of the median in linear time and constant space proposed in [30] and given in box 4.1.

3) Computing time remains a concern and the solution proposed in STRAT2 is still to restrain our attention to the top one million pairs of SNPs leading to a strictly positive test-statistic value on the real data. The drawback is again that such a filtering may increase the false positive rate. However, this issue is not as strong as for STRAT1, since the inflation factors are now test specific. This implies that pairs of SNPs leading to higher test-statistics on average because the median of their distribution under the null is significantly higher than the average one, are significantly more inflated. As a consequence, the p-value is higher than before and the FWER goes down. An important remark is that the final ordering of the pairs of SNPs is different in STRAT2 than STRAT1. This method has thus the potential to find hits that not been found by gammaMAXT. STRAT2 is a better choice to handle unmeasured confounding factors than STRAT1.

The final version of STRAT2 is reported in Box 4.3. and follows the same scheme as in Figure 4.4, except that individual inflation factors are now used.

Box 4.3. STRAT2 algorithm

(1) Create a vector R. For $j = 1, \dots, m$: compute t_j and if $t_j > 0$ push an object (t_j, SNP_{lj}, SNP_{rj}) at the back of R. We note $(R_k)_1$ the k^{th} test-statistic.

(2) Sort R in decreasing test-statistics order using the *quicksort* algorithm. If size f of R is higher than 10^6, resize R in order to keep only the top 10^6 elements.

(3) Find the median M_1. If f is an odd number, M_1 is the test-statistic stored in $R_{\lceil \frac{f}{2} \rceil}$, otherwise the average of the test-statistics stored in $R_{\frac{f}{2}}$ and $R_{\frac{f}{2}+1}$.

(4) Generate B permutations of the trait values and store them in memory.

(5) For $j = 1, \dots, f$: find the median $M_{2,j}$ by doing a nested loop over the permutations. For $i = 1, \dots, B$: set the trait values to the ones of the i^{th} permutation, compute $t_{i,j}$ and if $t_{i,j} > 0$ update $M_{2,j}$: if $M_{2,j}$ has not been initialized yet execute steps (1) and (2) of Box 4.1, otherwise step (3). When the loop over the permutations is finished, divide $(R_j)_1$ by $\lambda_j = \frac{M_1}{M_{2,j}}$.

(6) Create empty vectors T and P of size f. Fill P with 1's. For $i = 1, \dots, B$:

    (a) Set the trait values to the ones of the i^{th} permutation.
    (b) For $j = 1, \dots, f$: compute $t_{i,j}$ and store it in T_j.
    (c) For $j = f - 1, \dots, 1$: replace T_j by T_{j+1} if $T_j < T_{j+1}$.
    (d) For $j = 1, \dots, f$: increment P_j by one if $T_j \geq (R_j)_1$.

(7) For $j = 1, \dots, f$: divide P_j by $B + 1$. For $j = 1, \dots, f - 1$: replace P_{j+1} by P_j if $P_{j+1} < P_j$.

Real-life data analysis

We use extensive data of the international Inflammatory Bowel Disease (IBD) Genetic Consortium from 15 European countries typed on Immunochip. In this chapter we focus on case-control data on Crohn's disease (CD), which is one of the major types of IBD. For a full description of the data we refer to [44]. In summary, the initial cohort consists of 52,406 subjects for which 156,499 SNPs are available. The following quality control (QC) steps have been performed:

1) Removing subjects with missing phenotypes.
2) Keeping only the SNPs satisfying HWE (controls) > 0.0001, MAF>0.05 and call rate$>98\%$.
3) LD-pruning (SVS 8.3.1): window size of 50 SNPs, window increment 5 SNP, LD r^2 threshold 0.75 except for chromosome 6, where 0.35 is used.

The justification of all the aforementioned data handling is given in [44]. At the end of this procedure, we end up with 18,277 cases and 34,050 controls, for which 34,486 SNPs are available.

Analyzing this dataset with STRAT2 would take years. In fact, the software automatically prevents users to start runs with STRAT1 or STRAT2 involving more than one hundred million of SNP pairs[1]. For this reason, we implemented a two-stage analysis approach, using a random partition of the initial cohort, hereafter referred to as *discovery* and *replication* datasets. The partition of the data has been generated within the epistasis working group of the IIBDGC [Zhi Wei - New Jersey Institute of Technology, USA - unpublished]. The *discovery* data consists of 25,787 subjects (9,125 cases and 16,662 controls) and the *replication* one of 26,490 persons (9,102 cases and 17,388 controls). The first stage consists of analyzing the former with the parallel workflow of the gammaMAXT algorithm. This analysis can be performed within a few hours on a cluster. In this way, we have filtered the pairs of SNPs, i.e. obtained a list of the thousand most promising ones. At stage two, we use the STRAT2 algorithm on the *replication* data, but focus on these thousand SNP pairs only (using the *-s* option). The second stage takes about ten minutes.

Note that the fact that the *replication* dataset contains data that is independent from the *discovery* data, guarantees that the multiple-testing problem is handled properly. Undeniably, using the same datasets at both stages stage would lead to an important number of false-positives, since the true null distribution could not been assessed anymore. Indeed, since the data from stage two would be the same as the one from stage one, the pairs of SNPs would still be those producing the highest test-statistic values and would thus only allow to assess the head of the null distribution, not the whole distribution.

The analysis of the *discovery* dataset with the parallel workflow of the gamma-MAXT algorithm at stage one, already leads to a substantial reduction of the SNP pairs to investigate. Indeed, only 50 SNP pairs lead to a multiple-testing corrected p-value below 5%, out of the 594 624 855 candidates that were considered. These results are presented in Table 4.1.

[1] The total number of SNP pairs is here equal to $\frac{34486\times34485}{2} = 594\,624\,855$

Table 4.1: SNP pairs having a p-value < 0.05 on the *discovery* data with gammaMAXT

Rank	first SNP	second SNP	stat	p-value
1	imm_1_67478162	imm_1_67502643	153.33	0.001
2	imm_1_67476695	imm_1_67502643	141.301	0.001
3	imm_16_49402777	rs6500336	107.753	0.001
4	imm_16_49402274	rs6500336	105.335	0.001
5	imm_16_49317481	imm_16_49380465	96.862	0.001
6	imm_16_49317481	imm_16_49399698	84.014	0.001
7	rs6500336	rs17227589	78.187	0.001
8	imm_1_67453887	imm_1_67476695	72.289	0.001
9	imm_1_67439879	imm_1_67453887	71.721	0.001
10	imm_12_38780396	imm_12_38919755	70.505	0.001
11	rs3093664	rs17207986	69.699	0.001
12	rs3749946	rs17207986	67.751	0.001
13	imm_12_38780396	imm_12_38836591	67.155	0.001
14	imm_5_40528991	imm_5_40654985	65.849	0.001
15	rs3749946	rs3093664	65.563	0.001
16	rs12445755	rs16948451	64.791	0.001
17	rs17207986	rs206018	63.094	0.001
18	rs17207986	rs495089	61.646	0.001
19	imm_1_67441558	imm_1_67453887	60.955	0.001
20	imm_16_49251512	imm_16_49262042	58.909	0.001
21	rs2442719	rs3131621	58.154	0.001
22	imm_16_49399698	rs6500336	57.989	0.001
23	imm_5_150363823	imm_5_150367194	57.626	0.001
24	imm_5_40465299	imm_5_40477475	57.489	0.001
25	imm_16_49317481	imm_16_49400462	57.188	0.001
26	rs2243621	rs17207986	57.101	0.001
27	rs2073048	rs2858332	56.997	0.001
28	imm_12_38780396	imm_12_38887209	56.317	0.001
29	imm_16_49399698	imm_16_49404333	56.214	0.001
30	imm_16_49380465	rs6500336	56.123	0.001
31	imm_1_67399848	imm_1_67439879	54.670	0.001
32	imm_16_49403663	imm_16_49404333	53.623	0.001
33	rs17207986	rs2256594	53.292	0.002
34	imm_5_40438290	imm_5_40477475	52.050	0.004
35	rs2273017	rs2187823	51.657	0.004
36	rs2524082	rs2156875	51.197	0.006
37	rs17207986	rs13207945	51.059	0.006
38	imm_12_38780396	imm_12_38859741	51.047	0.006
39	imm_1_67442201	imm_1_67502643	50.626	0.007
40	rs3130286	rs206018	50.624	0.007
41	imm_20_44041068	imm_20_44044144	50.212	0.007
42	rs9673419	imm_16_49262042	50.189	0.007
43	rs9784876	rs241407	49.947	0.007
44	rs3749946	rs2273017	48.996	0.011
45	rs130075	rs3868078	48.883	0.013
46	rs4386816	rs3130286	48.511	0.021
47	imm_3_46393693	imm_3_46438428	48.439	0.021
48	rs130075	rs6905036	47.593	0.028
49	rs4386816	rs17207986	46.872	0.041
50	imm_12_38780396	imm_12_38850209	46.756	0.048

The analysis of the *replication* dataset with STRAT2, focusing on the top thousand pairs of SNPs identified at stage one, gives rise to a list of 65 significant SNP pairs, out of which 21 belong to the aforementioned 50 ones. Table 4.2 tabulates these 21 replicated results. A comparison and discussion of the number of significant pairs of SNPs observed in the different experiments of this chapter is reported to the discussion section. The results are indeed easier to interpret, once all figures are available.

Table 4.2: SNP pairs having a p-value < 0.05 on the *discovery* and *replication* data, using gammaMAXT on the *discovery* data and STRAT2 on the *replication* one.

Rank	first SNP	second SNP	repl. stat	repl. p-value
1	rs2524082	rs2156875	77.613	0.001
2	rs130075	rs3868078	68.289	0.001
3	imm_16_49402777	rs6500336	60.676	0.001
4	imm_1_67476695	imm_1_67502643	58.509	0.001
5	imm_16_49317481	imm_16_49380465	54.236	0.001
6	rs2273017	rs2187823	51.176	0.001
7	imm_16_49251512	imm_16_49262042	49.111	0.001
8	rs9673419	imm_16_49262042	48.630	0.001
9	imm_5_40438290	imm_5_40477475	46.189	0.001
10	imm_1_67478162	imm_1_67502643	43.694	0.001
11	rs3749946	rs3093664	43.305	0.001
12	imm_5_40465299	imm_5_40477475	37.001	0.001
13	imm_12_38780396	imm_12_38887209	36.641	0.001
14	imm_1_67439879	imm_1_67453887	36.213	0.001
15	imm_16_49399698	imm_16_49404333	35.806	0.001
16	imm_16_49317481	imm_16_49400462	34.720	0.001
17	rs9784876	rs241407	28.055	0.001
18	imm_12_38780396	imm_12_38836591	27.973	0.001
19	imm_16_49317481	imm_16_49399698	22.671	0.011
20	imm_16_49399698	rs6500336	21.486	0.017
21	imm_5_40528991	imm_5_40654985	21.014	0.021

Since the IBD data is from a consortium (a compilation of data from different hospitals and institutions in different countries), population structure is bound to occur. Hence, the fact that an automatic correction for unmeasured confounding factors reduces the list of relevant pairs of SNPs (from 50 to 21) is not surprising. Situations like the one depicted in Figure 4.2 are likely to be present in the dataset.

To investigate the consistency of the method, we perform another experiment. It consists in performing the exact same two-stage analysis as before, but in the other direction, i.e. use the gammaMAXT algorithm on the *replication* dataset and then the STRAT2 one on the *discovery* dataset (by focusing on the top thousand SNP pairs identified by the gammaMAXT). In this case, 216 SNP pairs lead to a multiple-testing corrected p-value below 5% at stage one (long results not shown, but recall that an overall interpretation of the number of results in the different analysis of this chapter is postponed to the discussion section). Note that 31 out of these 216 SNP pairs overlap with the 50 pairs identified on the *discovery* data with the gammaMAXT algorithm.

The analysis of the discovery dataset with STRAT2, focusing on the top thousand SNP pairs obtained at stage one, gives rise to a short list of 9 significant SNP pairs. All these pairs belong to the aforementioned 216 ones and are reported in Table 4.3.

Table 4.3: SNP pairs having a p-value < 0.05 on the *discovery* and *replication* data, using gammaMAXT on the *replication* data and STRAT2 on the *discovery* one.

Rank	first SNP	second SNP	repl. stat	repl. p-value
1	imm_1_67478162	imm_1_67502643	62.701	0.001
2	imm_1_67476695	imm_1_67502643	58.803	0.001
3	imm_16_49317481	imm_16_49380465	41.963	0.001
4	imm_16_49402777	rs6500336	34.633	0.001
5	imm_16_49317481	imm_16_49399698	28.067	0.002
6	imm_5_40528991	imm_5_40654985	26.973	0.004
7	imm_16_49399698	imm_16_49404333	24.274	0.007
8	imm_12_38780396	imm_12_38887209	23.336	0.010
9	rs2524082	rs2156875	22.879	0.015

These results are particularly interesting. Indeed, all SNP pairs from Table 4.3 are also in Table 4.2. Furthermore, the SNP pairs appearing in the former table, tend to reach a high ranking in the latter. These results are therefore very consistent and interesting to investigate from a biological point of view. In the next section, we propose methods to correct for covariates, in order to get rid of potential SNP pairs that would just be driven by the available covariates. Therefore, we postpone the follow up of the real-life analysis to after the presentation of these methods.

4.3 Correcting for available covariates

In GWAS, principal component (PC) on SNPs are used as corrective variables to adjust genetic association tests for confounding due to genetic ancestry [95]. When aiming to adjust tests used within MB-MDR for population stratification in this sense, a novel implementation that allows correcting for known covariates is needed. Obviously, such an implementation has a wider applicability and not necessarily restricts to confounding covariates. In the epigenetics context, when it goes beyond ethnicity, then one considers PCs as continuous axes of genetic information that capture "latent structure" [77]. In general, the data contains more information about the subjects than just the SNPs and the outcome, as was initially depicted on Figure 2.2. Standard available covariates are for instance gender, age of onset and some disease-related attributes (for instance, lung volume for Infant Respiratory Distress Syndrome). These covariates can be expressed on a quantitative or qualitative scale. Usually, it is not known if these covariates are associated to the disease or not. If not, taking them into account or not should not change the final result significantly. In this thesis, we propose two methods for correcting for available covariates in MB-MDR: residuals-based correction and on-the-fly correction. These methods are presented in the next two subsections.

4.3.1 Residuals-based correction

The idea of this method is to compute a new outcome variable r, based on the observed one y (or the two observed ones in the survival case, i.e. the time and censoring) and the k covariates $x_1, \ldots, x_k$ and run the analysis as usual on the dataset obtained by replacing y with r.

Continuous trait

Linear regression [140] is used to fit a model to the data. In our context, when there are S subjects, the aim is to fit the following model:

$$y_i = \beta_1 x_{i1} + \ldots + \beta_k x_{ik} + \beta_{k+1} + \epsilon, \qquad \forall i = 1, \ldots, S \tag{4.1}$$

Intuitively, the idea is to draw a plane in a $(k+1)$-dimensional space and update the coordinate system according to this it. The residuals are the vertical distances between the observations and the plane and can be used as a new outcome variable r. Finding the $\beta_1, \ldots, \beta_k, \beta_{k+1}$ values that minimize the sum of squared residuals of equation 4.3 is a classical problem. In this thesis, we use the library *alglib*, containing a speedy method called *lrbuild* solving this problem in $O(Sk)$ time. Writing a code from scratch to solve this task would not make any sense, since this function is called only once and increases only the final computing time by about one second. The residuals can be computed as

$$r_i = y_i - \beta_1 x_{i1} - \ldots - \beta_k x_{ik} - \beta_{k+1}, \qquad \forall i = 1, \ldots, S \tag{4.2}$$

Binary trait

In case of a trait expressed on a binary scale, logistic regression leads to more accurate results than a simple linear regression [35]. Intuitively, the idea is to force the predicted y_i values to be probabilities (i.e. the probability that the i^{th} subjects is a case, given its covariate values), whereas in classical regression they can take any real value.

Mathematically, when there are S subjects, the aim is to fit the following model:

$$P(y_i = 1) = \frac{e^{\beta_1 x_{i1} + \ldots + \beta_k x_{ik} + \beta_{k+1}}}{e^{\beta_1 x_{i1} + \ldots + \beta_k x_{ik} + \beta_{k+1}} + 1}, \qquad \forall i = 1, \ldots, S \tag{4.3}$$

In this thesis, we use the function *mnltrainh* from the aforementioned *alglib* library to solve this task quickly. The residuals can be computed as

$$r_i = P(y_i = 1) - \frac{e^{\beta_1 x_{i1} + \ldots + \beta_k x_{ik} + \beta_{k+1}}}{e^{\beta_1 x_{i1} + \ldots + \beta_k x_{ik} + \beta_{k+1}} + 1}, \qquad \forall i = 1, \ldots, S \tag{4.4}$$

Survival data

In the case of a trait expressed on a survival scale, let t_j represent the unique times and H_j the set of indexes i such that $Y_i = t_j$ and $C_i = 1$. Let m_j be the amount of events at time j. The idea is to use Efron's method (described in Section 2.4.2) and consisting in maximizing the following equation

$$L(\beta) = \prod_j \frac{\prod_{i \in H_j} \theta_i}{\prod_{l=0}^{m-1} (\sum_{i:Y_i \geq t_j} \theta_i - \frac{l}{m} \sum_{i \in H_j} \theta_i)} \tag{4.5}$$

where $\theta_j = e^{X_j \beta}$. To achieve this maximization, we can use the *coxfit6* function presented in Section 2.4.2. Recall that this function has been obtained by purifying the C source code of the R function *coxfit6*, in order to make a customized library out of it. The final function takes as arguments the time and censoring status of all subjects (sorted in increasing time order) and the covariate matrix (containing one line per individual and one column per covariate). It returns in particular the β vector that maximizes equation 4.5.

In this thesis, we use the martingale residuals of the Cox model [116] as a new outcome variable. These residuals can be interpreted, at each time t, as the difference over $[0, t]$ in the observed amount of events minus the expected amount given the Cox model. The R function *coxmart* computes the martingale residuals and the source code is fortunately again written in C. This function was added to our customized Cox library in a similar way as the *coxfit6* one, as described in Section 2.4.2. It takes as arguments the time and censoring status of all subjects (sorted in increasing time order), the score vector and some technical arguments. The score vector contains the score of each individual and can be computed as

$$e^{\beta_1 x_{i1} + \ldots + \beta_k x_{ik}}, \qquad \forall i = 1, \ldots, S \tag{4.6}$$

Real-life data analysis

To test the residuals-based correction method, we perform two different experiments. First, we analyze the *discovery* and *replication* datasets from Section 4.2.2 with the gammaMAXT algorithm, while performing a residuals-based correction of a covariate: gender. Second, we propose an experiment to correct for confounding by shared ancestry. The first analysis of the first experiment, consisting in analyzing the *discovery* dataset using gammaMAXT with residuals-based correction of gender, leads to 50 significant SNP pairs, reported in Table 4.4.

Table 4.4: SNP pairs having a p-value < 0.05 on the *discovery* data, using gamma-MAXT with residuals-based correction of gender.

Rank	first SNP	second SNP	stat	p-value	in Table 4.1
1	imm_1_67478162	imm_1_67502643	146.171	0.001	yes
2	imm_1_67476695	imm_1_67502643	133.786	0.001	yes
3	imm_16_49402777	rs6500336	113.038	0.001	yes
4	imm_16_49402274	rs6500336	107.681	0.001	yes
5	imm_16_49317481	imm_16_49380465	104.499	0.001	yes
6	imm_16_49317481	imm_16_49399698	91.579	0.001	yes
7	rs6500336	rs17227589	79.506	0.001	yes
8	imm_1_67439879	imm_1_67453887	70.893	0.001	yes
9	rs3093664	rs17207986	70.879	0.001	yes
10	imm_1_67453887	imm_1_67476695	70.571	0.001	yes
11	imm_12_38780396	imm_12_38919755	70.001	0.001	yes
12	rs3749946	rs17207986	69.628	0.001	yes
13	imm_16_49317481	imm_16_49400462	68.328	0.001	yes
14	rs3749946	rs3093664	67.357	0.001	yes
15	imm_12_38780396	imm_12_38836591	67.167	0.001	yes
16	imm_5_40528991	imm_5_40654985	66.753	0.001	yes
17	rs12445755	rs16948451	66.143	0.001	yes
18	rs17207986	rs206018	63.522	0.001	yes
19	imm_1_67441558	imm_1_67453887	62.812	0.001	yes
20	rs17207986	rs495089	61.758	0.001	yes
21	imm_16_49251512	imm_16_49262042	59.754	0.001	yes
22	rs2442719	rs3131621	58.288	0.001	yes
23	imm_16_49399698	imm_16_49404333	58.088	0.001	yes
24	imm_16_49399698	rs6500336	57.976	0.001	yes
25	imm_16_49380465	rs6500336	57.348	0.001	yes
26	imm_5_150363823	imm_5_150367194	57.228	0.001	yes
27	imm_5_40465299	imm_5_40477475	56.840	0.001	yes
28	imm_1_67399848	imm_1_67439879	56.590	0.001	yes
29	rs2243621	rs17207986	56.515	0.001	yes
30	imm_12_38780396	imm_12_38887209	56.380	0.001	yes
31	imm_16_49403663	imm_16_49404333	54.632	0.001	yes
32	rs17207986	rs2256594	53.595	0.002	yes
33	rs2073048	rs2858332	53.483	0.002	yes
34	imm_20_44041068	imm_20_44044144	52.476	0.006	yes
35	imm_12_38780396	imm_12_38859741	51.89	0.008	yes
36	rs2524082	rs2156875	51.701	0.010	yes
37	rs2273017	rs2187823	51.596	0.010	yes
38	rs17207986	rs13207945	51.247	0.011	yes
39	imm_5_40438290	imm_5_40477475	51.247	0.011	yes
40	rs3749946	rs2273017	50.493	0.014	yes
41	rs9784876	rs241407	50.447	0.014	yes
42	imm_1_67442201	imm_1_67502643	50.441	0.014	yes
43	rs130075	rs3868078	50.262	0.014	yes
44	rs9673419	imm_16_49262042	50.202	0.014	yes
45	rs130075	rs6905036	49.930	0.014	yes
46	rs3130286	rs206018	49.564	0.020	yes
47	rs2524082	rs3131621	48.713	0.023	no
48	rs4386816	rs3130286	48.053	0.030	yes
49	imm_12_38851428	imm_12_38887975	47.598	0.040	no
50	rs4386816	rs17207986	47.447	0.046	yes

Table 4.4 is extremely similar to Table 4.1. Only two SNP pairs appearing in the former do not appear in the latter and are marginal ones anyway, i.e. appearing in the very bottom of the list. The 48 SNP pairs out of 50 that are replicated tend to have a very similar ranking in both tables. These results suggests that gender may not be a strong contributor to the disease. The analysis of the *replication* dataset, using gammaMAXT with residuals-based correction of gender, leads to 221 significant SNP pairs, a number comparable to the 216 obtained without correction (long results again not shown, but examined in the discussion section). A total of 79 SNP pairs appear in both cases and most of these pairs are again top ranked ones, consolidating our hypothesis that gender may not be a strong contributor to the disease.

To conclude this first experiment, we compare the 50 SNP pairs identified by using gammaMAXT with a residuals-based correction of gender on the *discovery* data, with the 221 ones identified with the same method on the *replication* data. Table 4.5 reports the SNP pairs from the former analysis, replicated in the latter.

Table 4.5: SNP pairs having a p-value < 0.05 on the *discovery* and *replication* data, using each time gammaMAXT with residuals-based correction of gender.

Rank	first SNP	second SNP	repl. stat	repl. p-value
1	imm_1_67478162	imm_1_67502643	146.171	0.001
2	imm_1_67476695	imm_1_67502643	133.786	0.001
3	imm_16_49402777	rs6500336	113.038	0.001
4	imm_16_49402274	rs6500336	107.681	0.001
5	imm_16_49317481	imm_16_49380465	104.499	0.001
6	imm_16_49317481	imm_16_49399698	91.579	0.001
7	rs6500336	rs17227589	79.506	0.001
8	imm_1_67439879	imm_1_67453887	70.893	0.001
9	imm_1_67453887	imm_1_67476695	70.571	0.001
10	imm_16_49317481	imm_16_49400462	68.328	0.001
11	rs3749946	rs3093664	67.357	0.001
12	imm_12_38780396	imm_12_38836591	67.167	0.001
13	imm_5_40528991	imm_5_40654985	66.753	0.001
14	imm_16_49251512	imm_16_49262042	59.754	0.001
15	rs2442719	rs3131621	58.288	0.001
16	imm_16_49399698	imm_16_49404333	58.088	0.001
17	imm_16_49399698	rs6500336	57.976	0.001
18	imm_16_49380465	rs6500336	57.348	0.001
19	imm_5_150363823	imm_5_150367194	57.228	0.001
20	imm_5_40465299	imm_5_40477475	56.840	0.001
21	imm_12_38780396	imm_12_38887209	56.380	0.001
22	imm_16_49403663	imm_16_49404333	54.632	0.001
23	rs2073048	rs2858332	53.483	0.002
24	imm_12_38780396	imm_12_38859741	51.89	0.008
25	rs2524082	rs2156875	51.701	0.010
26	rs2273017	rs2187823	51.596	0.010
27	imm_5_40438290	imm_5_40477475	51.247	0.011
28	rs9784876	rs241407	50.447	0.014
29	rs130075	rs3868078	50.262	0.014
30	rs9673419	imm_16_49262042	50.202	0.014
31	rs130075	rs6905036	49.930	0.014
32	rs2524082	rs3131621	48.713	0.023
33	imm_12_38851428	imm_12_38887975	47.598	0.040

Table 4.5 shows that the SNP pairs that are replicated, tend as usual to be those reaching a high ranking at the first place. We have hence shown that the results are consistent when we correct for gender, a covariate that is probably not a contributor to the disease. We now perform a second experiment, as an attempt to correct for confounding by shared ancestry. Principal component analysis (PCA) was used in [44] to obtain the top 10 PCs from the SNP data of the *discovery* dataset and the top 10 PCs from the SNP data of the *replication* data. These top PCs can be seen as continuous axes of genetic variation. In this second experiment, we analyze the *discovery* and *replication* datasets with the gammaMAXT algorithm, while performing a residuals-based correction of their respective top 10 PCs taken from [44]. The analysis of the *discovery* data leads to a ranked list of 120 significant SNP pairs, while the analysis on the *replication* data leads to a ranked list of 169 ones. Table 4.6 has been obtained by removing the SNP pairs from the former list, that are not replicated in the latter one.

Table 4.6: SNP pairs having a p-value < 0.05 on the *discovery* and *replication* data, using each time gammaMAXT with residuals-based correction of the top 10 PCs.

Rank	first SNP	second SNP	repl. stat	p-value	in Table 4.5
1	rs130075	rs3868078	157.451	0.001	yes
2	rs3749946	rs3093664	112.947	0.001	yes
3	imm_16_49317481	imm_16_49380465	95.800	0.001	yes
4	rs2524082	rs2156875	94.032	0.001	yes
5	imm_16_49402777	rs6500336	91.476	0.001	yes
6	imm_16_49402274	rs6500336	91.168	0.001	yes
7	rs2442719	rs3131621	83.650	0.001	yes
8	rs4248153	rs1265048	83.646	0.001	no
9	imm_5_40528991	imm_5_40654985	76.976	0.001	yes
10	rs3749946	rs6922431	72.910	0.001	no
11	imm_16_84563031	imm_16_84576134	67.876	0.001	no
12	imm_10_6139051	ccc-10-6150332-C-G	66.274	0.001	no
13	imm_1_67453887	imm_1_67476695	64.412	0.001	yes
14	rs4713462	rs2848716	58.429	0.001	no
15	rs3749946	rs707934	58.424	0.001	no
16	imm_16_49251512	imm_16_49262042	57.354	0.001	yes
17	rs3093664	rs707934	55.590	0.001	no
18	imm_16_49399698	imm_16_49404333	54.936	0.001	yes
19	rs6500336	rs17227589	53.378	0.001	yes
20	imm_2_233869411	imm_2_233875787	53.362	0.001	no
21	rs4713438	rs2284178	52.338	0.001	no
22	rs6922431	rs495089	51.800	0.001	no
23	rs6922431	rs707934	50.610	0.002	no
24	rs707934	rs2256594	48.188	0.002	no
25	rs3749946	rs17207986	47.872	0.010	no

Correcting for the top 10 PCs reduces the list of replicated results stronger than correcting for gender. Indeed, only 12 SNP pairs from Table 4.6 also appear in Table 4.5, even if again, these pairs tend to rank better than the remaining ones. Nonetheless, the ranking obtained after correction of gender diverges much more from the ranking retrieved after correcting for the top 10 PCs, than from the ranking observed without any correction. This suggests that the population is stratified, a conclusion that is consistent with the one of the STRAT2 experiment.

A noticeable asset of the residuals-based correction method is that it is fast. Indeed, only one regression is performed to compute a new outcome variable and after that, the software runs as before. The computing time is hence approximately the same when using the residuals-based correction method as when the covariate is not taken into account. Note that the test-statistics are always computed using the methods of Section 2.3, regardless of the original scale on which the trait was expressed, since the residuals are invariably expressed on a continuous scale.

The most obvious drawback, is that this method removes the information provided by the covariates at the first place and that this information can therefore not be used in the analysis itself. As a consequence, covariates associated with the SNPs are not handled properly. For this reason, we propose a new correction method in the next section, called on-the-fly correction.

4.3.2 On-the-fly correction

In Chapter 2, we showed how to compute a number, capturing the strength of the association between a pair of SNPs and the trait, corrected for the main effects of the two SNPs. The idea of the on-the-fly correction is to generalize this procedure, in order to additionally correct for the available covariates. In other words, the idea is to improve the model by including additional covariates, to obtain a more accurate assessment of the association between an interactive pair and the trait, based on such a model.

Obviously, this method has an important impact on the computing time, since it complicates the model fitting behind every single MB-MDR test-statistic computation. The algorithms for just correcting for the main effects of the SNPs have been presented in Sections 2.2.2, 2.3.2 and 2.4.2 (respectively in the case of a trait expressed on a binary, continuous and survival scale). Generalizing them to take account of covariates requires important modifications of the source code, except for the latter. Indeed, the survival case is based on the *coxfit6* function, which readily takes account of covariates.

Survival data

The correction for the main effect of the SNPs from Section 2.4.2 is based on the Cox model and in particular on the function *coxfit6* from our customized Cox library, whose signature is given by

$$coxfit6(time, censoring, covar) \Rightarrow [\beta, l(\beta)] \tag{4.7}$$

This function readily takes account of covariates. Therefore, the algorithm from Box 2.12 only needs minor changes to take account of the k covariates $x_1, \ldots, x_k$. The amount of columns of the model matrix X should simply be increased by k. The content of these columns is trivial, since the rows correspond to the subjects of the study with no missing genotype value for neither of the two SNPs under investigation. The news columns thus contain the k covariate values of that subjects. No other changes are needed and the final algorithm is given in Box 4.4.

Box 4.4. Survival MB-MDR statistic with on-the-fly correction

(1) Create a vector T of size $dim = N_1 \times N_2$ and fill it with 0's. Create empty vectors V_t and V_c. Create empty matrix X consisting of $2+k$ columns in case of additive coding and N_1+N_2-2+k columns in case of codominant coding.

(2) For $s = 1,\ldots,S$: if g_{slj} and g_{srj} are not missing do $T_{g_{slj}\times N_2+g_{srj}}{++}$, add ti_s at the back of V_t, c_s at the back of V_c and insert a new row at the end of X:

- (a) If additive coding: put g_{slj} in the 1^{st} column and g_{slj} in the 2^{nd}.
- (b) If codominant coding: perform two loops. For $a = 1,\ldots,N_1-1$: if (a $= g_{slj}$) put 1 in the a^{th} column, otherwise 0. For $b = 1,\ldots,N_2-1$: if (b $= g_{srj}$) put 1 to the $(N_1+b)^{th}$ column, otherwise 0.
- (c) Fill the k last columns with the k covariate values of subject s.

(3) Call $coxfit6(V_t, V_c, X)$. Let $loglik_{main}$ be the returned log likelihood.

(4) Insert a new column at the end of X filled with 0's. Define $D_{\text{crit}} = 2,705543$.

(5) Create HLO vector R of size dim. For $h = 0,\ldots,dim-1$:

- (a) If $(T_h < 10)$ set $R_h =$ "O" and skips step (b), (c), and (d).
- (b) Initialize $a = 0$. Overwrite the last column of X. For $s = 1,\ldots,S$: if g_{slj} and g_{srj} are not missing: do (if $(g_{slj}\times N_2 + g_{srj} = h)$ overwrite the a^{th} element with 1, else with 0) and $a{++}$.
- (c) Call $coxfit6(V_t, V_c, X)$. Let β and $loglik_{main+}$ be the returned objects.
- (d) Compute $D_{\text{obs}} = 2(loglik_{main+} - loglik_{main})$.
- (e) If $(D_{\text{obs}} < D_{\text{crit}})$ set $R_{mn} =$ "O", else if the last element of the β vector is greater than zero set $R_{mn} =$ "H", else set $R_{mn} =$ "L".

(6) If there is no "H" and no "L" in the R matrix, return $t_j = 0$.

(7) If there is at least one "H":

- (a) Initialize $a = 0$. Overwrite the last column of X. For $s = 1,\ldots,S$: if $R_{g_{slj}\times N_2+g_{srj}} =$ "H") overwrite the a^{th} element with 1, else with 0) and $a{++}$.
- (b) Call $coxfit6(V_t, V_c, X)$. Let $loglik_{HvsLO}$ be the returned log likelihood.
- (c) Compute $D_{HvsLO} = 2(loglik_{HvsLO} - loglik_{main})$.

(8) If there is at least one "L":

- (a) Initialize $a = 0$. Overwrite the last column of X. For $s = 1,\ldots,S$: if $R_{g_{slj}\times N_2+g_{srj}} =$ "L") overwrite the a^{th} element with 1, else with 0) and $a{++}$.
- (b) Call $coxfit6(V_t, V_c, X)$. Let $loglik_{LvsHO}$ be the returned log likelihood.
- (c) Compute $D_{LvsHO} = 2(loglik_{LvsHO} - loglik_{main})$.

(9) Return $t_j = \max(D_{\text{HvsLO}}, D_{\text{LvsHO}})$.

Binary trait

The algorithm of Section 2.2.2, computing a number capturing the degree of association between a pair of SNPs and a trait expressed on a binary scale, corrected for the main effects of the two SNPs, has been extremely customized for this task. It relies on the fact that the subjects can be split into groups depending on their genotype (nine groups in the bi-allelic case). To optimize the computing time, it takes advantage of the fact that the trait is dichotomous: the ratio of cases versus controls is computed for each group and this value is the one that is fitted. As a consequence, the model matrix is small (nine rows for bi-allelic data) and the fitting does not affect the computing time too much.

This strategy is only made possible by the fact that the main effects of the SNPs (or environmental variables) that we are correcting for, are categorical variables. This is at the first place the property that allows a tabular organization of the information. Therefore, generalizing this idea to correct for continuous covariates is not possible. Such variables would force us to fit the actual trait value of every single subjects, in order to be able to correct for their individual continuous covariate values[2]. As a consequence, the model matrix would not be small anymore, since it would have as many rows as subjects. We showed in Section 2.4.2 that such model matrices rises the computing time by orders of magnitude. This would make a GWAIs impossible in practice. In the presence of continuous variables we propose to categorize them or to create clusters of individuals with similar patterns based on the available covariates, although we realize that this may be seen as a potential shortcoming of the current implementation.

Hence, since a set of categorical variables can always be reduced to a single categorical variable, we suppose that there is only one covariate to correct for in this section. Note that in practice, the software allows users to have several covariates, but internally recode them automatically into a single one after heaving read the data. This operation is done once for all and has thus a negligible impact on the computing time (less than one second). Note that the internal recoding takes into account that some factor level combination are not observed in the data (hereby reducing the actual number of factor levels). As a consequence, the model matrix is automatically smaller than if we had worked with several variables, improving the computing time.

The correction for the main effects of the SNPs from Section 2.2.2 is based on three matrices: the affected-subjects, the unaffected-subjects and the HLO one, as was illustrated in Figure 2.6. To take account of the different values of the categorical covariate, the first two matrices need to be split. Let's take gender as leading example. The first matrix is split into an affected-males and an affected-females matrix and the second one into an unaffected-males and unaffected-females matrix, as illustrated in Figure 4.6. This idea is easy to generalize to a covariate that can take $N_c > 2$ different values, i.e. by splitting the affected-subjects and unaffected-subjects into N_c matrices each.

[2] Taking the average covariates values of the group members would loose too much information and defeat the purpose of the on-the-fly correction method

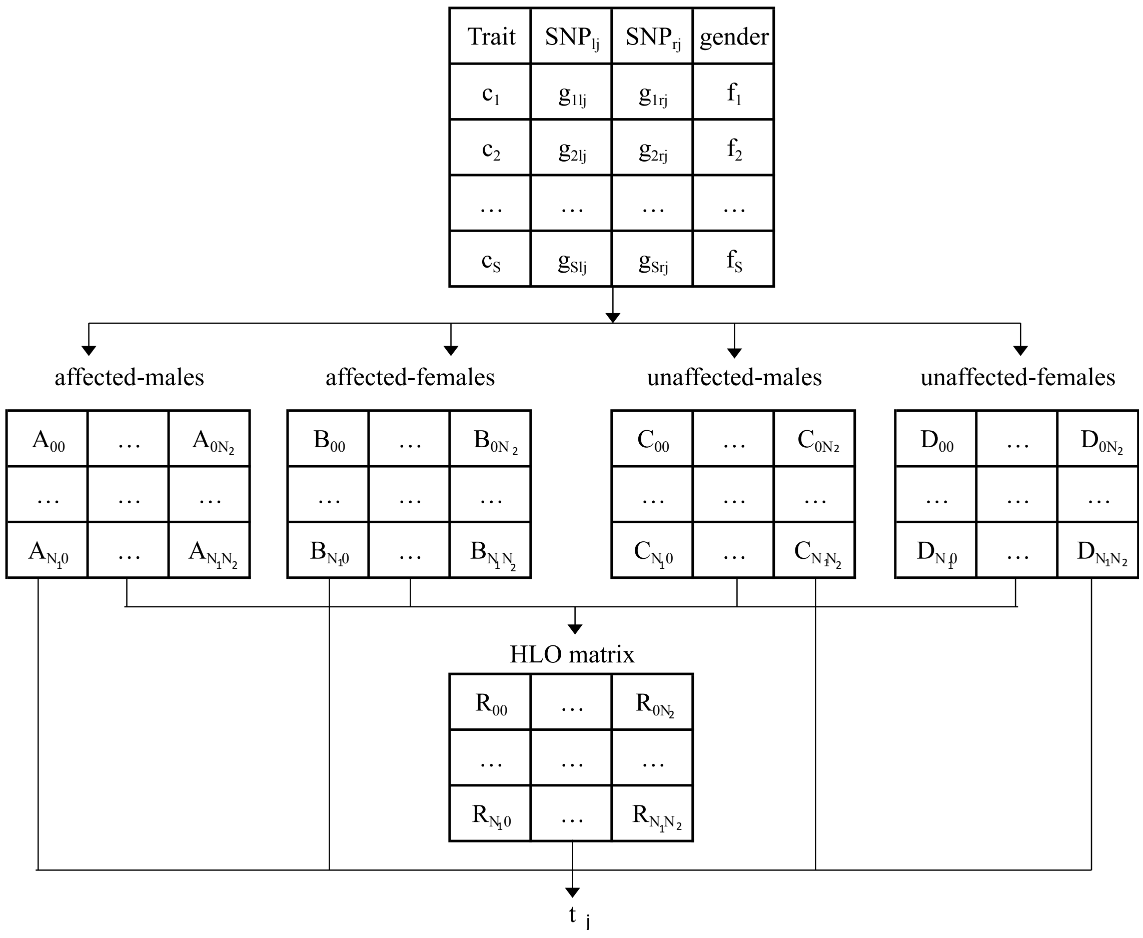


Figure 4.6: Computation of an MB-MDR statistic for a binary trait, with on-the-fly correction of gender and of the main effect of the SNPs. Input: c_s is 1 (0) if the s^{th} subject is a case (control), g_{slj} and g_{srj} are the genotypes of the s^{th} subject for SNP_{lj} and SNP_{rj} respectively and f_s is 0 if the subject is a male and 1 if it is a female . The computation can be decomposed in three steps. First, the affected-males, unaffected-males, affected-females and unaffected-females matrices are constructed. A_{mn} and B_{mn} are respectively the amounts of affected males/females, whose genotype $g_{slj} = m$ and $g_{srj} = n$. C_{mn} and D_{mn} are respectively the amounts of unaffected males/females, whose genotype $g_{slj} = m$ and $g_{srj} = n$. Second, the HLO matrix is constructed. R_{mn} is either “H” if the subjects whose genotype is m for SNP_{lj} and n for SNP_{rj} have a high statistical risk of disease, “L” if they have a low statistical risk and “O” if there is no statistical evidence. Third, the final t_j value is computed from the five matrices constructed at steps 1 and 2.

For efficiency reasons and to take account of the empty groups more easily, the affected-males, affected-females, unaffected-males and unaffected-females matrices are in practice in fact stored as two vectors A and U. The first/second element of A contains the amounts of affected males/females in the first genotype group, the third/fourth elements of A the amounts of affected males/females for the second genotype group and so on. The same method is used to construct U, which contains unaffected subjects amounts.

The correction method described in Section 2.2.2, was based on the creation of a vector Y, containing the ratio of cases versus number of subjects for every non-empty genotype group. The idea of the on-the-fly correction method (in the leading example of gender) is to multiply Y's size by two. The first (second) element of Y becomes the ratio of case males (females) versus total amount of males (females) belonging to the first genotype group, the third (fourth) element of Y becomes the ratio of case males (females) versus total amount of males (females) belonging to the second genotype group and so on. These ratios can easily be obtained from vectors A and U.

The model matrix obviously needs to be extended too. The former rows of the model matrix that was presented in Section 2.2.2 (called X_{old} here), needs to be duplicated (mimicking the fact that the members of each genotype groups have been split into males and females) and a new column with an alternation of 0's and 1's added at the end of the matrix (to indicate which rows correspond to males/females). In the case of bi-allelic data, when no group is empty, these matrices are explicitly given by

$$X_{old} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 2 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 0 \\ 1 & 2 & 1 \\ 1 & 2 & 2 \end{bmatrix} \qquad X_{new} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 2 & 0 \\ 1 & 0 & 2 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 0 \\ 1 & 1 & 2 & 1 \\ 1 & 2 & 0 & 0 \\ 1 & 2 & 0 & 1 \\ 1 & 2 & 1 & 0 \\ 1 & 2 & 1 & 1 \\ 1 & 2 & 2 & 0 \\ 1 & 2 & 2 & 1 \end{bmatrix}$$

This idea can be generalized for a covariate that can take $N_c > 2$, by multiplying Y's size by N_c instead of just two, making N_c copies of each rows of X_{old} instead of just two and adding $N_c - 1$ columns at the end of X instead of just one. The first new column should contain 1's for each row that is a second copy of a line from X_{old} and 0's elsewhere, the second newly added column should contain 1's for each row that is a third copy of a line from X_{old} and 0's elsewhere and so on. This procedure comes from the fact that the on-the-fly correction method is based on a codominant coding scheme (presented in Section 2.2.2).

As an example, in the case of $N_c = 3$, the model matrix becomes

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 2 & 0 & 0 \\ 1 & 0 & 2 & 1 & 0 \\ 1 & 0 & 2 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 2 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 \\ 1 & 1 & 2 & 0 & 1 \\ 1 & 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 \\ 1 & 2 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 1 & 0 \\ 1 & 2 & 1 & 0 & 1 \\ 1 & 2 & 2 & 0 & 0 \\ 1 & 2 & 2 & 1 & 0 \\ 1 & 2 & 2 & 0 & 1 \end{bmatrix}$$

The tricky part of the on-the-fly-correction method, is that even if Y and X have been expanded, the size of the HLO matrix should not be modified. Indeed, expanding it too would just lead to a 3D interaction MB-MDR test-statistic computation, where gender can be seen as an environmental variable. This method has been described in Section 2.2.4 and should not be confused with the subject of this section: the computation of a 2D interaction MB-MDR statistic, corrected for three factors (the main effects of the two SNPs and the categorical covariate).

The fitting of the model $Y = X\beta$ can be obtained by the algorithm from Box 2.2, except that when no group is empty, the A, U, T and Y vectors now have a dimension of $dim = N_1 \times N_2 \times N_c$ (instead of just $N_1 \times N_2$ as before) and that as consequence, some trivial changes are required at steps (2) and (3). However, the algorithm from Box 2.3, computing the MB-MDR statistic itself, needs some elaborate changes. Indeed, the fact that some groups may be empty makes the implementation tricky. A new vector C of size $dim_{HLO} = N_1 \times N_2$ is needed, to store the amount of non empty categories represented for each genotype group. This allows to remove elements of vectors A, U and Y (and rows of matrix X) without loosing track of which elements (rows) correspond to which genotype group, i.e. HLO matrix element. The computation, illustrated in Figure 4.6, can be decomposed into three steps:

1) Construction of the affected-subjects A and unaffected-subjects U vectors. They are created with a size of $N_1 \times N_2 \times N_c$, initialized with 0's and build up through a loop over the subjects of the dataset. For $s = 1, \ldots S$: if no value is missing for subject s: increment a cell of the affected-subjects vector if $c_s = 1$, otherwise a cell of the unaffected-subjects one. The index of the cell to be incremented is given by $g_{slj} \times N_2 \times N_c + g_{srj} \times N_c + f_s$, where g_{slj} is the subject's value for SNP1, g_{srj} for SNP2 and f_s it's covariate value. After this process, a loop is performed to construct the model matrix X, the categories vector C (created with a size of $dim_{HLO} = N_1 \times N_2$ and initialized with 0's), the vector T containing the total number of subjects in each non-empty group and the vector Y containing the ratio of cases versus number of subjects in these groups. An integer a, initialized to zero, is needed to keep track of the index of the non-empty genotype group under investigation. The model matrix X is created empty and at each iteration $h = 0, \ldots, dim - 1$, a loop is performed over the different categories of the covariate. For $c = 0, \ldots, N_c - 1$: if there is at least one subject whose genotype satisfies $(SNP_{lj} \times N_2 \times N_c + SNP_{rj} \times N_c + c = h)$, the number of such subjects is stored in T_a, a new row is added to X, whose first elements are the ones of the h^{th} row of the hard coded-matrix[3] and the remaining ones are all set to 0 except the c^{th} one set to 1, Y_a is computed as the ratio of A_a over T_a and a is incremented. When no such subject satisfying the given genotype is found, the a^{th} elements of A and U are removed. Let dim be the final size of vectors A, U, T and Y.

2) The second step consists in generating the HLO-matrix, but again coded in vector format. First, the model is fitted as described in Box 2.2 and the QR decomposition $Q_1R_1 = \sqrt{W}X_{main}$ is obtained by calling a library (alglib) dedicated for this task. Then, a column is added at the back of X. Each R_h element of the HLO-vector depends on a test for association between the trait and the belonging to the genotype group satisfying $(SNP_{lj} \times N_2 + SNP_{rj} = h)$, but this time adjusted for the main effects of the SNPs and of the covariate. An integer a again keeps track of the index of the non-empty genotype group under investigation. For $h = 0, \ldots, dim_{HLO} - 1$, compute the amount tot of subjects in this category. If $tot < 10$ set the h^{th} element of the HLO-vector to "O". Otherwise, overwrite the last column of X with 0's, except the elements from index a till index $a + C_h - 1$ which are set to 1. Then compute the score statistic from equation 2.5. When this value, following a 1df (one degrees of freedom) χ^2, is below the critical value $\chi^2_{\text{crit}} = 2,705543$, set the h^{th} element of the HLO-vector to "O". Otherwise, set it to either "H" if $u > 0$ or "L" otherwise. At the end of each loop over h, increment a by C_h to indicate that C_h groups have been considered.

3) If there is neither "H" nor "L" values in the HLO vector, return 0. Otherwise, compute two score statistics and return the maximum. The first/second is obtained by overwriting the last column of X with 0's, except for the elements for which the corresponding genotype group is in high/low risk which are set to 1 and computing the score statistic using the same strategy as in step 2.

The final algorithm is reported in Box 4.5.

[3] Recall that in the rare cases where the hard-coded matrix does not exist, i.e. when either $N_1 > 3$ or $N_2 > 3$, then the row is created explicitly

Box 4.5. Binary MB-MDR statistic with on-the-fly correction

(1) Create vectors A, U, T and Y of size $dim = N_1 \times N_2 \times N_c$.

(2) Fill A and U with 0's. For $s = 1, \ldots, S$: if g_{slj} and g_{srj} are not missing do $A_{g_{slj} \times N_2 \times N_c + g_{srj} \times N_c + f_s}{++}$ if $c_s = 1$ and $U_{g_{slj} \times N_2 \times N_c + g_{srj} \times N_c + f_s}{++}$ otherwise.

(3) Create matrix X with $dim_c = N_1 + N_2 + N_c - 2$ columns. Create a vector C of size $dim_{HLO} = N_1 \times N_2$ and fill it with 0's. Set $a = 0$. For $h = 0, \ldots, dim_{HLO} - 1$: for $c = 0, \ldots, N_c - 1$: compute $T_a = A_a + U_a$; if $(T_a = 0)$ erase the a^{th} element of vectors A and U, else perform the following operations

 (a) Add a new row to X. The first elements are those of the h^{th} row of the hard-coded model matrix. The other ones are filled by a loop. For $j = 1, \ldots N_c - 1$: set the next element to 1 if $j = c$, otherwise to 0.

 (b) Compute $Y_a = \frac{A_a}{T_a}$.

 (c) Perform $a{++}$.

 Compute $dim = a$.

(4) Compute $N_A = A_0 + \ldots + A_{dim-1}$, $N_U = U_0 + \ldots + U_{dim-1}$ and $N = N_A + N_U$.

(5) Create fitted values vector μ of size dim and initialize all elements to $\frac{N_A}{N}$.

(6) Create linear predictor vector $\eta = ln(\frac{\mu}{1-\mu})$.

(7) Compute $\mathcal{D} = 2\{N \ln N - N_A \ln N_A - N_U \ln N_U + \sum_{i=0}^{dim-1} [A_i \ln(Y_i) + U_i \ln(1 - Y_i)]\}$.
(where only the first term of the latter sum is taken if $Y_i = 1$ and only the second if $Y_i = 0$)

(8) Perform the following iterations:

 (a) Create a vector $z = \eta + \frac{Y-\mu}{\mu(1-\mu)}$ (taking $z_i = \eta_i - 1$ if $\mu_i = 0$ and $z_i = \eta_i + 1$ if $\mu_i = 1$)

 (b) Calculate $dim \times dim$ diagonal matrix W where $W_{ii} = T_i \mu_i (1 - \mu_i)$.

 (c) Calculate $dim_c \times dim_c$ symmetric positive definite matrix $\mathcal{I} = X'WX$.

 (d) Calculate right-hand side vector v of size dim_c defined by $v = X'Wz$.

 (e) Calculate vector β of size dim_c by trying to solve $\mathcal{I}\beta = v$. If it fails, remove all linearly dependent columns from X and go back to step (c).

 (f) Update $\eta = X\beta$ and $\mu = \frac{1}{1+e^{-\eta}}$.

 (g) Compute $\mathcal{D} = 2\{\sum_{i=0}^{dim-1} A_i \ln\frac{Y_i}{\mu_i} + U_i \ln\frac{1-Y_i}{1-\mu_i})$.
 (where only the first term is taken if $Y_i = 1$ and only the second if $Y_i = 0$)

 (h) Stop iterating if $\frac{|\mathcal{D} - \mathcal{D}_{\text{old}}|}{0.1 + \mathcal{D}} < 10^{-8}$ or after 25 iterations.

(9) Find $Q_1 R_1 = \sqrt{W} X$.

(10) Create empty vectors $\mathcal{V}$ of size dim_c and $\mathcal{O}$ of size dim. Create residual vector ρ of size dim. For $i = 0, \ldots, dim - 1$: if $(\mu_i = 0)$ set $\rho_i = -1$, else if $(\mu_i = 1)$ set $\rho_i = 1$, else set $\rho_i = \frac{Y_i - \mu_i}{\mu_i(1-\mu_i)}$.

(11) Create HLO vector R of size dim_{HLO}. Set $a = 0$. For $h = 0, \ldots, dim_{HLO} - 1$: if $C_h > 0$:

(a) Set $tot = 0$. For $k = 0, \ldots, C_h$: do $tot = tot + T_{a+k}$

(b) If ($tot < 10$) set $R_h =$ "O" and skip steps (c), (d) and (e).

(c) For $i = 0, \ldots, dim_c - 1$: do ($\mathcal{V}_i = 0$) and (for $j = a, \ldots, a + C_h - 1$: do ($\mathcal{V}_i = \mathcal{V}_i + Q_{1ji}\sqrt{W_j}$). For $i = dim_c - 1, \ldots, 0$: do (for $j = i + 1, \ldots, dim_c - 1$: $\mathcal{V}_i = \mathcal{V}_i - R_{1ij}\mathcal{V}_j$) and ($\mathcal{V}_i = \frac{\mathcal{V}_i}{R_{1ii}}$).

(d) For $s = 0, \ldots, dim - 1$: do (if ($a \leq s < a + C_h$) do $\mathcal{O}_s = 1$ else do $\mathcal{O}_s = 0$) and (for $t = 0, \ldots, dim_c - 1$: do $\mathcal{O}_s = \mathcal{O}_s - X_{st}\mathcal{V}_t$).

(e) Set $u = i = 0$. For $s = 0, \ldots, dim - 1$: do (u+=$W_s\mathcal{O}_s\rho_s$) and (i+=$W_s\mathcal{O}_s^2$). If i=0 set $R_h =$ "O", else calculate $\mathcal{S} = \frac{u^2}{i}$. If $\mathcal{S} < \chi^2_{\text{crit}} = 2,705543$ set $R_h =$ "O", else if ($u > 0$) set $R_h =$ "H", else set $R_h =$ "L".

(f) Execute $a = a + C_h$.

(12) If there is no "H" and no "L" in the HLO vector R, return $t_j = 0$.

(13) If there is at least one "H":

(a) For $i = 0, \ldots, dim_c - 1$: do ($\mathcal{V}_i = 0$) and (set $k = a = 0$; for $j = 0, \ldots, dim_{HLO} - 1$: if ($R_j =$ "H") do (for $j = a, \ldots, a + C_h - 1$: $\mathcal{V}_i = \mathcal{V}_i + Q_{1ji}\sqrt{W_j}$) ; $a = a + C_h$. For $i = dim_c - 1, \ldots, 0$: do (for $j = i + 1, \ldots, dim_c - 1$: $\mathcal{V}_i = \mathcal{V}_i - R_{1ij}\mathcal{V}_j$) and ($\mathcal{V}_i = \frac{\mathcal{V}_i}{R_{1ii}}$).

(b) Set $a = k = 0$. For $s = 0, \ldots, dim_{HLO} - 1$: if $C_h > 0$: execute (if ($R_j =$ "H") do (for $s = a, \ldots, a + C_h - 1$: $\mathcal{O}_s = 1$) else do (for $s = a, \ldots, a + C_h - 1$: $\mathcal{O}_s = 0$)) and (for $t = 0, \ldots, dim_c - 1$: for $s = a, \ldots, C_h - 1$: do $\mathcal{O}_s = \mathcal{O}_s - X_{st}\mathcal{V}_t$) and ($a = a + C_h$).

(c) Set $u = i = 0$. For $s = 0, \ldots, dim - 1$: do (u+=$W_s\mathcal{O}_s\rho_s$) and (i+=$W_s\mathcal{O}_s^2$). If (i>0) compute $\mathcal{S}_{\text{HvsLO}} = \frac{u^2}{i}$ else set it to zero.

(14) If there is at least one "L":

(a) For $i = 0, \ldots, dim_c - 1$: do ($\mathcal{V}_i = 0$) and (set $k = a = 0$; for $j = 0, \ldots, dim_{HLO} - 1$: if ($R_j =$ "L") do (for $j = a, \ldots, a + C_h - 1$: $\mathcal{V}_i = \mathcal{V}_i + Q_{1ji}\sqrt{W_j}$) ; $a = a + C_h$. For $i = dim_c - 1, \ldots, 0$: do (for $j = i + 1, \ldots, dim_c - 1$: $\mathcal{V}_i = \mathcal{V}_i - R_{1ij}\mathcal{V}_j$) and ($\mathcal{V}_i = \frac{\mathcal{V}_i}{R_{1ii}}$).

(b) Set $a = k = 0$. For $s = 0, \ldots, dim_{HLO} - 1$: if $C_h > 0$: execute (if ($R_j =$ "L") do (for $s = a, \ldots, a+C_h-1$: $\mathcal{O}_s = 1$) else do (for $s = a, \ldots, a+C_h-1$: $\mathcal{O}_s = 0$)) and (for $t = 0, \ldots, dim_c - 1$: for $s = a, \ldots, C_h - 1$: do $\mathcal{O}_s = \mathcal{O}_s - X_{st}\mathcal{V}_t$) and ($a = a + C_h$).

(c) Set $u = i = 0$. For $s = 0, \ldots, dim - 1$: do (u+=$W_s\mathcal{O}_s\rho_s$) and (i+=$W_s\mathcal{O}_s^2$). If (i>0) compute $\mathcal{S}_{\text{LvsHO}} = \frac{u^2}{i}$ else set it to zero.

(15) Return $t_j = \max(\mathcal{S}_{\text{HvsLO}}, \mathcal{S}_{\text{LvsHO}})$.

Continuous trait

Adapting the algorithm of Section 2.3.2 to additionally correct for an available covariate is similar to what was done for a binary trait. For the same reasons as before, we focus on a single categorical covariate which can take N_c different values. The algorithm from Section 2.3.2 is based on four matrices: the amount-matrix, mean-matrix, varnum-matrix and HLO-matrix, as was illustrated in Figure 2.8. To take account of the covariate, the first three need to be split, in a similar way as was done in the case of a trait expressed on a binary scale. Let's take again gender as leading example. The first matrix would be split into an amount-male and an amount-female matrix, the second into a male-mean-trait-value and female-mean-trait-value matrix and so on. This idea generalizes readily to a covariate that can take any amount N_c of different values, i.e. by splitting each of the matrices into N_c ones. The model matrix needs again to be extended and this can be done in the exact same way as in the previous section. The fitting of the model can be obtained from Box 2.7, except that the dimension is now given by $N_1 \times N_2 \times N_c$ when no group is empty (instead of just $N_1 \times N_2$ as before). However, the algorithm from Box 2.8, computing the MB-MDR statistic itself, needs again some more elaborate changes, similar to what was done in Box 4.5. A vector C of size $dim_{HLO} = N_1 \times N_2$ is once more used to store the number of non-empty categories represented for each genotype group. The computation can be decomposed into three steps:

1) Construction of the total amount T, mean Y and varnum V vectors. They are created with a size $N_1 \times N_2 \times N_c$ and initialized with zeros. Then, several loops are performed. First, a loop over the subjects of the dataset: for $s = 1, \dots S$, if no value is missing for subject s, increment an of T by one and an element of Y by c_s. The index of the cell to be incremented is given by $g_{slj} \times N_2 \times N_c + g_{srj} \times N_c + f_s$, where g_{slj} is the subject's value for SNP1, g_{srj} for SNP2 and f_s it's covariate value. Second, divide all elements of the mean-vector by the elements of T (except when the group is empty). Third, loop over the subjects of the dataset again: for $s = 1, \dots S$, if no value is missing for subject s increment an element of the varnum-vector by the square of the difference between an element of the mean-vector and the c_s value. The index of the element is computed as above. Finally, remove empty groups from the amount-, mean- and varnum- vectors and on the fly construct the model matrix from the hard-coded one (or explicitly in the rare scenarios explained earlier) and compute the SSE_{full} value. Let dim be the final size of the vectors.

2) The second step still consists in generating the HLO-matrix, but again coded in vector format. First, the model is fitted as described in Box 2.7. Then, a column is added to X. For $h = 1, \dots, dim - 1$, compute the amount tot of subjects in this cateogry. If $tot < 10$ set the h^{th} element of the HLO-vector to "O". Otherwise, overwrite the last column of X with 0's, except the elements from index a till index $a + C_h - 1$ which are set to 1. Then compute the Wald statistic from equation 2.18. When this value is below the critical value F_{crit} defined by an F distribution with degrees of freedom 1 and $N - (c+1)$, set the h^{th} element of the HLO-vector to "O". Otherwise, set it to either "H" if $\beta_{c+1} > 0$, to indicate a high risk for disease, or to "L" to indicate a low risk for the disease.

3) If there are neither "H" nor "L" values in the HLO vector, return 0. Otherwise, compute the following two Wald statistics and return the maximum. The first/second is obtained by overwriting the last column of X with 0's, except for the elements for which the corresponding genotype group is in high/low risk which are set to 1 and computing the score statistic using the same strategy as in step 2.

The final algorithm is reported in Box 4.6.

Box 4.6. Continuous MB-MDR statistic with on-the-fly correction

(1) Create vectors T, B, Y, V of size $dim = N_1 \times N_2 \times N_c$.

(2) Fill T and B with 0's. For $s = 1, \dots, S$: if g_{slj} and g_{srj} are not missing do $T_{g_{slj} \times N_2 \times N_c + g_{srj} \times N_c + f_s}$++ and $B_{g_{slj} \times N_2 \times N_c + g_{srj} \times N_c + f_s}$+=$c_s$.

(3) Set $N = 0$. For $h = 0, \dots, dim - 1$: if $(T_h > 0)$ do $Y_h = \frac{B_h}{T_h}$ and N+=T_h. Define $F_{crit} = F_{0,1}(1, N - 2)$.

(4) Fill V with 0's. For $s = 1, \dots, S$: if g_{slj} and g_{srj} are not missing do $V_{g_{slj} \times N_2 \times N_c + g_{srj} \times N_c + f_s}$+=$(Y_{g_{slj} \times N_2 \times N_c + g_{srj} \times N_c + f_s} - c_s)^2$.

(5) Create matrix X with $dim_c = N_1 + N_2 + N_c - 2$ columns. Create a vector C of size $dim_{HLO} = N_1 \times N_2$ and fill it with 0's. Set $SSE_{full} = 0$. Set $a = 0$. For $h = 0, \dots, dim_{HLO} - 1$: for $c = 0, \dots, N_c - 1$: if $(T_a = 0)$ erase the a^{th} element of vectors T, B, Y and V, else perform the following operations

   (a) Add a new row to X. The first elements are those of the h^{th} row of the hard-coded model matrix. The other ones are filled by a loop. For $j = 1, \dots N_c - 1$: set the next element to 1 if $j = c$, otherwise to 0.

   (b) Execute SSE_{full}+=V_a.

   (c) Perform a++.

   Compute $dim = a$.

(6) Create $dim \times dim$ diagonal weights matrix W. The diagonal elements correspond are those of T.

(7) Calculate $dim_c \times dim_c$ symmetric positive definite info matrix $\mathcal{I} = X'WX$.

(8) Calculate the right-hand side vector $v = X'WY$ of size dim_c.

(9) Estimate parameter vector β of size dim_c by solving the linear system $\mathcal{I}\beta = v$. If it fails, remove all linearly dependent columns from X and go back to (7).

(10) Create fitted values vector μ of size dim as $\mu = X\beta$.

(11) Set $SSE_{main} = SSE_{full}$. For $h = 0, \dots, dim - 1$: SSE_{main}+=$(Y_h - \mu_h)^2 \times T_h$.

(12) Update the sizes of X to $dim \times (dim_c + 1)$, $\mathcal{I}$ to $(dim_c + 1) \times (dim_c + 1)$ and v to $dim_c + 1$.

(13) Create HLO vector R of size dim_{HLO}. Set $a = 0$. For $h = 0, \ldots, dim_{HLO} - 1$: if $C_h > 0$:

  (a) Set $t = 0$. For $k = 0, \ldots, C_h$: do $t = t + T_{a+k}$
  (b) If ($t < 10$) set R_h = "O" and skip steps (c), (d), (e), (f), (g), (h) and (i).
  (c) For $h = 0, \ldots, dim_{HLO} - 1$: if ($a \leq h < a + C_h$) overwrite the h^{th} element of the last column of X with 1, otherwise with 0.
  (d) Calculate $\mathcal{I} = X'WX$. Calculate the right-hand side vector $v = X'WY$.
  (e) Estimate parameter vector β by solving the linear system $\mathcal{I}\beta = v$.
  (f) Create fitted values vector $\mu = X\beta$.
  (g) Initialize $SSE_{main+} = SSE_{main}$. For $h = 0, \ldots, dim - 1$: execute $SSE_{main+} += (Y_h - \mu_h)^2 \times T_h$.
  (h) Compute $W = (SSE_{main} - SSE_{main+})\frac{N-(dim_c+1)}{SSE_{main+}}$. If $W < F_{crit}$ set R_h = "O", else if $\beta_{dim_c+1} > 0$ set R_h = "H", else set R_h = "L".
  (i) Execute $a = a + C_h$.

(14) If there is no "H" and no "L" in the HLO vector R, return $t_j = 0$.

(15) If there is at least one "H":

  (a) Set $k = a = 0$. For $h = 0, \ldots, dim_{HLO} - 1$: if $C_h > 0$ do (if (R_k = "H") do (for $j = a, \ldots, a + C_h - 1$: overwrite the j^{th} element of the last column of X with 1) else (for $j = a, \ldots, a + C_h - 1$: overwrite the j^{th} element of the last column of X with 0) and k++).
  (b) Calculate $\mathcal{I} = X'WX$. Calculate the right-hand side vector $v = X'WY$.
  (c) Estimate parameter vector β by solving the linear system $\mathcal{I}\beta = v$.
  (d) Create fitted values vector $\mu = X\beta$.
  (e) Initialize $SSE_{main+} = SSE_{main}$. For $h = 0, \ldots, dim - 1$: execute $SSE_{main+} += (Y_h - \mu_h)^2 \times T_h$.
  (f) Compute $\mathcal{W}_{\text{HvsLO}} = (SSE_{main} - SSE_{main+})\frac{N-(dim_c+1)}{SSE_{main+}}$.

(16) If there is at least one "L":

  (a) Set $k = a = 0$. For $h = 0, \ldots, dim_{HLO} - 1$: if $C_h > 0$ do (if (R_k = "L") do (for $j = a, \ldots, a + C_h - 1$: overwrite the j^{th} element of the last column of X with 1) else (for $j = a, \ldots, a + C_h - 1$: overwrite the j^{th} element of the last column of X with 0) and k++).
  (b) Calculate $\mathcal{I} = X'WX$. Calculate the right-hand side vector $v = X'WY$.
  (c) Estimate parameter vector β by solving the linear system $\mathcal{I}\beta = v$.
  (d) Create fitted values vector $\mu = X\beta$.
  (e) Initialize $SSE_{main+} = SSE_{main}$. For $h = 0, \ldots, dim - 1$: execute $SSE_{main+} += (Y_h - \mu_h)^2 \times T_h$.
  (f) Compute $\mathcal{W}_{\text{LvsHO}} = (SSE_{main} - SSE_{main+})\frac{N-(dim_c+1)}{SSE_{main+}}$.

(17) Return $t_j = \max(\mathcal{W}_{\text{HvsLO}}, \mathcal{W}_{\text{LvsHO}})$.

Real-life data analysis

To test the on-the-fly correction method, we perform again two experiments. First, we analyze the *discovery* and *replication* datasets from Section 4.2.2 with the gammaMAXT algorithm, while performing an on-the-fly correction of gender. On the *discovery* data we obtain 48 significant SNP pairs, reported in Table 4.7.

Table 4.7: SNP pairs having a p-value < 0.05 on the *discovery* data, using gammaMAXT with on-the-fly correction of gender.

Rank	first SNP	second SNP	stat	p-value	in Table 4.1
1	imm_1_67478162	imm_1_67502643	151.761	0.001	yes
2	imm_1_67476695	imm_1_67502643	140.157	0.001	yes
3	imm_16_49402777	rs6500336	107.071	0.001	yes
4	imm_16_49402274	rs6500336	104.209	0.001	yes
5	imm_16_49317481	imm_16_49380465	96.597	0.001	yes
6	imm_16_49317481	imm_16_49399698	84.543	0.001	yes
7	rs6500336	rs17227589	77.862	0.001	yes
8	imm_1_67453887	imm_1_67476695	72.817	0.001	yes
9	imm_1_67439879	imm_1_67453887	72.290	0.001	yes
10	imm_12_38780396	imm_12_38919755	69.737	0.001	yes
11	rs3093664	rs17207986	68.787	0.001	yes
12	rs3749946	rs17207986	67.361	0.001	yes
13	imm_12_38780396	imm_12_38836591	67.030	0.001	yes
14	imm_5_40528991	imm_5_40654985	66.027	0.001	yes
15	rs3749946	rs3093664	65.278	0.001	yes
16	rs12445755	rs16948451	63.933	0.001	yes
17	rs17207986	rs206018	62.164	0.001	yes
18	rs17207986	rs495089	60.662	0.001	yes
19	imm_1_67441558	imm_1_67453887	60.518	0.001	yes
20	imm_16_49251512	imm_16_49262042	59.533	0.001	yes
21	imm_16_49399698	rs6500336	57.876	0.002	yes
22	rs2442719	rs3131621	57.394	0.002	yes
23	rs2073048	rs2858332	57.263	0.002	yes
24	imm_5_150363823	imm_5_150367194	57.257	0.002	yes
25	imm_16_49317481	imm_16_49400462	56.793	0.002	yes
26	imm_5_40465299	imm_5_40477475	56.691	0.002	yes
27	imm_16_49380465	rs6500336	56.135	0.002	yes
28	imm_12_38780396	imm_12_38887209	55.980	0.002	yes
29	rs2243621	rs17207986	55.941	0.002	yes
30	imm_16_49399698	imm_16_49404333	55.825	0.002	yes
31	imm_1_67399848	imm_1_67439879	55.718	0.002	yes
32	imm_16_49403663	imm_16_49404333	53.057	0.003	yes
33	rs17207986	rs2256594	52.614	0.003	yes
34	rs2524082	rs2156875	51.594	0.004	yes
35	imm_5_40438290	imm_5_40477475	51.355	0.004	yes
36	rs2273017	rs2187823	51.288	0.004	yes
37	imm_12_38780396	imm_12_38859741	51.274	0.004	yes
38	imm_1_67442201	imm_1_67502643	50.765	0.005	yes
39	rs17207986	rs13207945	50.551	0.006	yes
40	rs9784876	rs241407	50.366	0.006	yes
41	imm_20_44041068	imm_20_44044144	50.222	0.006	yes
42	rs9673419	imm_16_49262042	50.000	0.006	yes
43	rs3130286	rs206018	49.607	0.007	yes
44	rs130075	rs3868078	49.057	0.013	yes
45	rs3749946	rs2273017	48.770	0.016	yes
46	imm_3_46393693	imm_3_46438428	48.285	0.02	yes
47	rs4386816	rs3130286	48.118	0.02	yes
48	rs130075	rs6905036	47.585	0.030	yes

This result is extremely similar to the one obtained without correction for gender (reported in Table 4.1). All 48 SNP pairs appearing in Table 4.7 also appear in Table 4.1 and are ranked similarly. We came to the same conclusion with the residuals-based method. This result is hence very consistent with the rest and once more consolidates our hypothesis that gender is probably not a strong contributor to the disease.

The analysis of the *replication* dataset using gammaMAXT with on-the-fly correction of gender, leads to 169 significant SNP pairs, a number comparable to the 216 obtained without correction (long results again not shown, but examined in the discussion section), but again a bit lower. The on-the-fly correction method seem to bo more conservative than the residuals-base done. A total of 64 SNP pairs appear in both cases and most of these pairs are again top ranked ones.

To conclude this first experiment, we compare the 48 SNP pairs identified by using gammaMAXT with on-the-fly correction of gender on the *discovery* data, with the 169 ones identified with the same method on the *replication* data. Table 4.8 reports the significant pairs of SNPs from the former analysis, replicated in the latter.

Table 4.8: SNP pairs having a p-value < 0.05 on the *discovery* and *replication* data, using each time gammaMAXT with on-the-fly correction of gender.

Rank	first SNP	second SNP	repl. stat	p-value	in Table 4.5
1	imm_16_49402274	rs6500336	132.835	0.001	yes
2	rs2524082	rs2156875	130.353	0.001	yes
3	rs130075	rs3868078	128.857	0.001	yes
4	imm_16_49402777	rs6500336	116.925	0.001	yes
5	rs2442719	rs3131621	111.62	0.001	yes
6	imm_1_67476695	imm_1_67502643	103.885	0.001	yes
7	imm_16_49317481	imm_16_49380465	95.360	0.001	yes
8	imm_1_67478162	imm_1_67502643	93.458	0.001	yes
9	rs2273017	rs2187823	86.850	0.001	yes
10	imm_16_49251512	imm_16_49262042	84.929	0.001	yes
11	rs3749946	rs3093664	81.465	0.001	yes
12	imm_16_49317481	imm_16_49399698	80.976	0.001	yes
13	imm_5_40438290	imm_5_40477475	80.495	0.001	yes
14	imm_5_40465299	imm_5_40477475	80.081	0.001	yes
15	rs9673419	imm_16_49262042	79.273	0.001	yes
16	imm_16_49399698	rs6500336	78.921	0.001	yes
17	imm_12_38780396	imm_12_38887209	69.079	0.001	yes
18	imm_1_67439879	imm_1_67453887	67.880	0.001	yes
19	imm_16_49399698	imm_16_49404333	65.205	0.001	yes
20	imm_1_67453887	imm_1_67476695	64.806	0.001	yes
21	imm_12_38780396	imm_12_38859741	62.999	0.001	yes
22	imm_12_38780396	imm_12_38836591	62.875	0.001	yes
23	imm_16_49317481	imm_16_49400462	61.788	0.001	yes
24	imm_5_150363823	imm_5_150367194	61.050	0.001	yes
25	rs130075	rs6905036	58.931	0.001	yes
26	rs2073048	rs2858332	58.084	0.001	yes
27	imm_16_49380465	rs6500336	55.768	0.001	yes
28	imm_16_49403663	imm_16_49404333	52.766	0.002	yes
29	rs6500336	rs17227589	52.502	0.002	yes
30	rs9784876	rs241407	51.665	0.002	yes

The 30 SNP pairs reported in Table 4.8, had all already been identified via residuals-based correction of gender, i.e. are all in Table 4.5. However, the former contains 33 hits. The on-the-fly correction method was than again slightly more conservative than the residuals-based one. This may indicate that a few SNPs are associated with gender, a situation that cannot be taken into account by the residuals-based method.

In the second experiment, we correct for confounding by shared ancestry. In the case of the residuals-based correction method, we have used the top 10 PCs for this purpose. However, the on-the-fly correction method can only handle categorical variables. Therefore, we use the *R* package *mclust* (with the default options) to automatically split the subjects into clusters, depending on their top 10 PCs values [33]. The *mclust* package is based on a Gaussian finite mixture models fitted via EM algorithm for model-based clustering [32]. We run the function *Mclust* and get 9 clusters. We use the cluster to which the subjects belong as new categorical variable, that we correct for using the on-the-fly method. We analyze the *discovery* and *replication* datasets with the gammaMAXT algorithm, while performing an on-the-fly correction of cluster membership. On the *discovery* data, we obtain 210 significant SNP pairs and on the *replication* data, 93 ones. Table 4.9 reports the SNP pairs from the former analysis, that could be replicated in the latter one.

Table 4.9: SNP pairs having a p-value < 0.05 on the *discovery* and *replication* data, using each time gammaMAXT with on-the-fly correction of cluster membership.

Rank	first SNP	second SNP	repl. stat	p-value	in Table 4.6
1	rs130075	rs3868078	149.208	0.001	yes
2	rs3749946	rs3093664	112.459	0.001	yes
3	rs2524082	rs2156875	96.197	0.001	yes
4	rs4248153	rs1265048	85.689	0.001	yes
5	rs3749946	rs6922431	79.215	0.001	yes
6	imm_5_40528991	imm_5_40654985	78.653	0.001	yes
7	imm_16_49317481	imm_16_49380465	78.089	0.001	yes
8	rs2442719	rs3131621	75.858	0.001	yes
9	1kg_19_18149069	1kg_19_18170365	73.765	0.001	no
10	imm_16_49402777	rs6500336	71.521	0.001	yes
11	imm_10_6139051	ccc-10-6150332-C-G	70.074	0.001	yes
12	imm_16_49402274	rs6500336	66.874	0.001	yes
13	rs3749946	rs707934	65.902	0.001	yes
14	rs2273017	rs2187823	60.036	0.001	no
15	rs6922431	rs707934	58.673	0.001	yes
16	rs3093664	rs707934	56.757	0.001	yes
17	rs6922431	rs495089	54.014	0.001	yes
18	imm_16_49399698	imm_16_49404333	53.898	0.001	yes

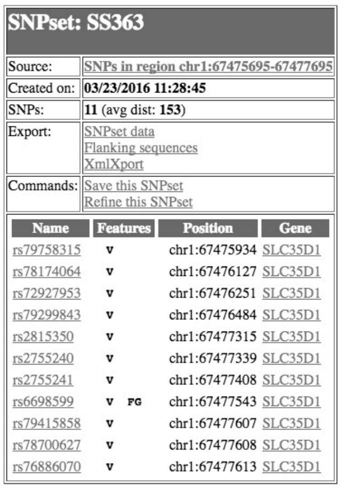
Correcting for cluster membership reduces the list of replicated results much stronger than correcting for gender. A similar observation was made via the residuals-based method, where correcting for the top 10 PCs reduces the list of significant results much stronger than correcting for gender. Interesting to note is also that correcting for cluster membership with the on-the-fly method reduces leads to a shorter list than correcting for the top 10 PCs via the residuals-based approach. Indeed, Table 4.9 is made up of 18 SNP pairs and Table 4.6 of 25 ones. Almost all SNP pairs appearing in the former (16), also appear in the latter, showing great consistency between the methods.

To validate the results of this chapter, we perform a short biological interpretation of a selection of SNP pairs that appeared in the very top of most rankings of Table 4.1-4.9. These pairs are recorded in Table 4.10.

Table 4.10: Promising SNP pairs to investigate from a biological point of view.

Rank	first SNP	second SNP	top obs. stat	HLO matrix
1	rs130075	rs3868078	157.451	$\begin{bmatrix} L & H & H \\ H & L & O \\ O & O & L \end{bmatrix}$
2	imm_1_67478162	imm_1_67502643	153.330	$\begin{bmatrix} H & L & L \\ L & H & L \\ L & L & H \end{bmatrix}$
3	imm_1_67476695	imm_1_67502643	141.301	$\begin{bmatrix} H & L & L \\ L & H & L \\ L & L & H \end{bmatrix}$
4	imm_16_49402777	rs6500336	132.835	$\begin{bmatrix} H & L & L \\ L & H & H \\ L & O & H \end{bmatrix}$
5	rs2524082	rs2156875	130.353	$\begin{bmatrix} L & L & H \\ O & H & L \\ H & O & L \end{bmatrix}$
6	rs3749946	rs3093664	112.947	$\begin{bmatrix} L & H & H \\ H & L & L \\ H & O & O \end{bmatrix}$
7	imm_16_49317481	imm_16_49380465	104.499	$\begin{bmatrix} L & O & H \\ H & L & L \\ H & H & L \end{bmatrix}$
8	rs2273017	rs2187823	86.850	$\begin{bmatrix} H & L & L \\ L & O & H \\ L & H & H \end{bmatrix}$

The difficulty to find the genes that are close to the SNPs that we want to investigate from a biological point of view, lies in the fact that Table 4.10 contains a mixture of SNP names starting with `rs` and `imm`. The former are standard names and can be found in the genome browser at `https://genome.ucsc.edu/`. The latest assembly (Dec. 2013 (GRCH38/hg38)) was used. The variant annotator tool allows to find out easily on which chromosome the SNP is located and which is the nearest gene. The hits starting with `imm` contains one short number after the first underscore and a larger one after the second one. The short number gives the chromosome and the larger one the location on the chromosome. This position does not always correspond to a SNP in particular, making it a bit tricky to find out the nearest genes. An online SNP finder tool at `http://snpper.chip.org/bio/snpper-enter-snp` is used to solve this issue (the build 131 of dbSNP was used, with the hg19 genome). For every SNP, a search is performed 1000 positions around the position given by the larger number in the name. In practice, the result list returned by the tool is either empty, or containing several SNP names, all located on the same gene, as illustrated in Figure 4.7 for one of the hits of Table 4.10. The identified genes are reported in Table 4.11 and when no gene could be identified, we consider that the location is intergenic.



SNPset: SS363

Source:	SNPs in region chr1:67475695-67477695
Created on:	**03/23/2016 11:28:45**
SNPs:	**11** (avg dist: **153**)
Export:	SNPset data Flanking sequences XmlXport
Commands:	Save this SNPset Refine this SNPset

Name	Features	Position	Gene
rs79758315	V	chr1:67475934	SLC35D1
rs78174064	V	chr1:67476127	SLC35D1
rs72927953	V	chr1:67476251	SLC35D1
rs79299843	V	chr1:67476484	SLC35D1
rs2815350	V	chr1:67477315	SLC35D1
rs2755240	V	chr1:67477339	SLC35D1
rs2755241	V	chr1:67477408	SLC35D1
rs6698599	V FG	chr1:67477543	SLC35D1
rs79415858	V	chr1:67477607	SLC35D1
rs78700627	V	chr1:67477608	SLC35D1
rs76886070	V	chr1:67477613	SLC35D1

Figure 4.7: Result obtained on SNP Finder to find out the nearest gene corresponding to the hit imm_1_67476695 from Table 4.10, which is SLC35D1.

Table 4.11: Chromosomes and nearest genes of the SNPs from Table 4.10

first SNP	chrom.	nearest gene	second SNP	chrom.	nearest gene
rs130075	6	CCHCR1	rs3868078	6	intergenic
imm_1_67478162	1	SLC35D1	imm_1_67502643	1	SLC35D1
imm_1_67476695	1	SLC35D1	imm_1_67502643	1	SLC35D1
imm_16_49402777	16	C16orf78	rs6500336	16	intergenic
rs2524082	6	HLA-C	rs2156875	6	HLA-C
rs3749946	6	HCP5	rs3093664	6	LTA
imm_16_49317481	16	CBLN1	imm_16_49380465	16	intergenic
rs2273017	6	C6orf10	rs2187823	6	intergenic

We observe that all SNP-pairs involve concomitant variants located on the same chromosomes. The top ranked hit is located on chromosome 6 and involve the gene CCHCR1, typically linked to psoriasis, itself associated with many systemic disorders including Crohn's disease [117, 115]. The second and third hits of Table 4.11 are very similar an may be driven by the same biological phenomenon. All these hits involve SNPs located close to the SLC35D1 gene, a well-known one associated to CD [96]. This demonstrates the capabilities of mbmdr-4.4.1.out to retrieve such significant associations. To our knowledge, the gene C16orf78 was not associated to Crohn's disease yet and may be a new discovery. Lab validation is required to confirm this hypothesis. However, note that the C16orf78 gene is a protein coding one that was already related to prostate cancer [113]. The SNP pairs corresponding to the fifth hit are located on chromosome 6, the main one reported in this table, close to the HLA-C gene. This region is again a well known susceptibility one for Crohn's disease [83, 58]. The sixth hit is an interaction implicating two different genes, both linked to CD: HCP5 [68, 20] and LTA [142]. CBLN1 enhances secretory activity of human adrenal gland [85] and since adrenal insufficiency is a known CD symptom, this hit is not really surprising from a biological point of view either. The gene C6orf10 has been linked with psoriasis [100]. These very interesting results demonstrates that mbmdr-4.4.1.out can find SNP pairs that are relevant from a biological point of view.

4.4 Discussion

As the core of MB-MDR involves association tests, it may be affected by confounding that is unaccounted for. Therefore, in this chapter, we explored a few options to develop a version of MB-MDR that can be applied to structured populations. We have introduced four approaches: two based on principles of genomic control as developed for GWAs and two based on using principal components as continuous axes of genetic variation.

The STRAT1 algorithm is the immediate extension of Devlin and Roeder [25] for GWAIs. This algorithm produces the exact same ranking as the original maxT algorithm, with different p-values and has hence not the potential to discover pairs of SNPs not identified by maxT. However, it can be regarded as a method to reduce the FWER of the maxT algorithm, observed in the presence of population stratification. The interpretation of the inflation factor as defined in STRAT1 is different than the one in Devlin and Roeder. There the aim is to estimate the inflation of the variance of the test statistic. In MB-MDR we have a particular situation in that there is no closed form for the theoretical distribution of the MB-MDR test statistics. So there is no closed form for the variance of these statistics either. Earlier work [80] revealed the impact of highly varying MAFs on MB-MDR test distributions. In the basic STRAT1 idea, the primary aim was not to estimate variance inflation, but to account for overly optimistic results in a naive, yet easy to compute, way. Therefore, an immediate generalization could be to compute variances V_1 and V_2 instead of medians M_1 and M_2. In the original implementation of Devlin and Roeder of an estimated inflation factor lamda, the assumption was made that the same factor could be used for all test statistics. One of these assumptions was based on inbreeding values being rather similar across SNPs. Roughly translated to the epistasis scene: is a tendency towards multi-locus homozygosity in subpopulations compared to the entire population, stable across SNP pairs? We hypothesize that this may be too strong an assumption (pilot work under progress - results not shown). This led to the creation of STRAT2.

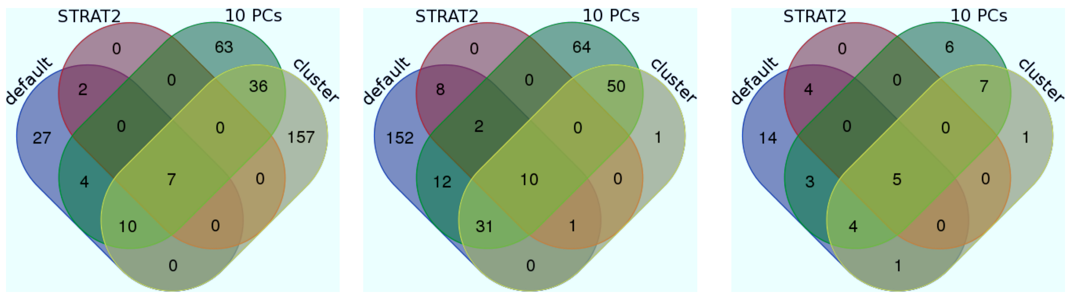
The STRAT2 algorithm is a method based on test-specific inflation factors, that has the potential to find significant pairs of SNPs, that could neither be identified by the maxT algorithm, nor by the gammaMAXT or STRAT1 one. To make the basic idea work in practice, several accommodations were needed, leading to a potential increase of the FWER. However, in practice, the order of magnitude of pairs of SNPs identified is similar to what is obtained via the maxT algorithm. Only recently, we came across a paper that also uses ideas of creating test-specific corrections [132]. Notably, these authors use effect size estimates to construct test-specific inflation factors. An effect size is a quantitative measure for the strength of the association. MB-MDR does not readily provide these, as using a function of the MB-MDR test values directly to estimate some sort of effect size will lead to winner's curse problems [144]. When estimated effect sizes or penetrances are based on the original data used to detect the signal, the results are affected by an ascertainment bias known as the "winner's curse". In addition, the reported MB-MDR test values are the result of an accumulation of association evidences built on the same data set each time. Their variability can therefore not be computed via theoretical distributions. This was also the reason to adhere to permutation tests for significance assessments. Theoretical distributions for the final testing step in MB-MDR no longer hold due to the aforementioned accumulation process.

The principal component method is inherently unable to completely account for the impact of population stratification [132]. Due to the popularity and wide-spread use of principal components for genome-wide association studies in structured populations, we explored how PCs can be inserted in the MB-MDR framework. Either they are regressed out up front and residuals are submitted to MB-MDR as new traits, or they are considered as categorical adjustments to the tests implemented in MB-MDR. These tests are the multilocus genotype prioritization tests and the final MB-MDR test statistic based on an aggregate of the multilocus. Rather than setting up simplistic simulation scenarios in which we can formally evaluate the proposed strategies for MB-MDR in structured populations, we focused on a real-life data application. Table 4.12 compares the number of significant SNP pairs found in the different experiments performed in this chapter on the IBD data (except the experiments correcting for gender, which have been shown to lead to almost the same results as without correction). Figure 4.8, indicates the amount of common results across the methods of Table 4.12.

Table 4.12: Amount of SNP pairs having a p-value < 0.05 on the *discovery* data, on the *replication* data and on both datasets, depending on the method used.

Method	*discovery*	*replication*	in both
gammaMAXT with default options	50	216	31
STRAT2 with a two-stage approach	9	21	9
gammaMAXT with 10 PCs correction (residuals-based)	120	169	25
gammaMAXT with cluster correction (on-the-fly)	210	93 (*)	18

(*) However, note that many SNP pairs are marginally not significant at 5%, i.e. 213 SNP pairs have a p-value $\leq 5,1\%$.



(a) *discovery* (b) *replication* (c) in both

Figure 4.8: Venn diagram of the results of Table 4.12

Since the IBD data is consortium one, population structure is bound to occur. The fact that STRAT2 leads to significantly less hits than gammaMAXT with the default options, suggests that STRAT2 corrects strongly for unmeasured confounding factors, yet maybe too much. On one side, more optimal results are expected with genomic kinship control in structured populations, i.e. leading to less false positive interactions. On the other side, the two-stage approach that had to be put in place to take account of the fact that STRAT2 requires a lot of additional computing time may have lowered the power. Indeed, focusing on a subset of the SNP pairs at our disposal, may alter our estimation of the null. Undoubtedly, the sample of investigated SNPs is not an independent and identically distributed one anymore, since it contains pair of SNPs that have higher chances to be regarded as associated to the outcome. This can lead to a right shift of the head of the null distribution. As a consequence, the observed values will be compared to higher values than they should, leading to a power decrease. However, the fact that the selection of the SNP pairs to investigate is performed on another dataset than the one on which the analysis itself is performed, should soften this potential issue. Indeed, the SNP pairs that are not linked to the disease, but lead to a high test-statistic by chance on a particular dataset, do probably not also lead to a large test-statistic on another dataset.

We notice that the number of significant SNP pairs is significantly higher on the *replication* data than on the *discovery* one[4]. Since this is in particular also true when gammaMAXT is used with the default options (a setting that has been tested intensively on synthetic data), we don't expect that this is due to an issue with the method. In fact, a closer look at the partition performed by the epistasis working group of the IIBDGC to split the original data into the *discovery* and *replication* one, reveals that the subjects have not been assigned totally randomly to the datasets. In reality, the IBD data is composed of several batches of subjects with different origins and in practice whole batches have been kept together. Hence, the characteristics of the subjects belonging to the *discovery* data may be slightly different from those belonging to the *replication* one. In the rest of the discussion, we focus on the results obtained on the *discovery* data, which may contain less false-positives.

The fact that the on-the-fly correction method lead to more significant results than the residuals-based one (210 vs 120) can seem strange at first. In fact, this comes from the fact that the residuals-based and on-the-fly method do not rely on the same null hypothesis. Indeed, in the on-the-fly correction method, we have computed clusters and use cluster membership as a new categorical covariate to correct for. However, the correction itself is performed within the low-level functions of `mbmdr-4.4.1.out` that compute the test-statistics. The architecture of the software separates these function from the high-level ones presented in Chapter 3 to correct for multiple-testing. However, the latter are all based on the idea that the trait values should be permuted. As a consequence, the cluster stays with the SNP values and not with the trait values, i.e. the null hypothesis behind the on-the-fly correction method is that the trait values are not associated with the cluster and SNP values. In the residuals-based method yet, the residuals are used as new outcome variable, so that the shared ancestry information is now implicitly permuted with the trait values. The null hypothesis in this case is hence that the trait values and subpopulation belongings are not associated with the SNP values. Hence, a perspective of this thesis is to write a new option to permute the categorical covariates together with the trait values. This would allow users to test both null hypothesis, the new option being more in line with the residuals-based one and enabling a better comparison of their respective results.

As we can see, the real-life application on IBD presented in this chapter, gives further food for thought to ameliorate our approaches, prior to extensive and computationally intensive analysis runs. The importance we give to real-life applications as part of a feed-back/feed-in loop to methods improvement is highlighted by the examples we give in the next chapter.

[4] Except for the last method reported in Table 4.12, where this amount is similar if the (*) is taken into account.

Chapter 5

Learning from data

5.1 Outline

In this chapter, we present a few practical data analyses methods that helped us to increase our understanding of the issues that GWAIs involves in practice and to develop novel methods. This chapter addresses four different analytic problems. First, taking structured populations correctly into account. This problematic was already intensively discussed in Chapter 4 and a data analysis was already performed on the IBD data. However, in the course of this PhD, other experiments were performed in our lab on the IBD data and the most relevant ones are described here. This leads to interesting results from a biological point of view, demonstrating that the methods proposed in Chapter 2 allows to identify pairs of SNPs having high chances to be associated to a disease. The pros and cons of using another approach than MB-MDR are also discussed shortly. Second, we analyze data on asthma disease to show that non-genetic factors cannot be overlooked. They can be important predictors for disease-related traits on their own, but they can also modify genetic effects or genetic interactions. Therefore, in this experiment, we investigate the flexibility of our software to discover GxE interactions. Third, we validate the survival part of our software, which was so far only described from a theoretical point of view in this manuscript. Censored traits can be seen as a form of missingness and are omnipresent in clinical practice (time to disease progression, time to death, time to first replace, ...). Not so many tools exists that can rapidly and efficiently carry out a GWAIs with categorical variables. In this experiment, we also test the effect of LD pruning on power. We give users an idea of what they can expect from an analysis, depending on the number of individuals available in their study and the heritability of the disease under investigation. Fourth, we discuss the issues linked with deriving biological interpretations or novel hypothesis from the pairs of SNPs identified by our software. We have shown throughout the thesis that it is important to interpret statistical epistasis findings in the context of potential marginal effects. For this reason, we propose a tool for visualizing main effect and pairwise-interactions results at a glance. We demonstrate the tool on data on Crohn's disease. Finally, we present the first results of an undergoing analysis of real-life data on Coronary artery disease (CAD). However, since these results are not published yet and since the data cannot be shared with other parties, the SNP names have been replaced by dummy ones (snp1, snp2, snp3, . . .). In this way, the results can be kept confidential until publication.

5.2 MB-MDR for structured population

Handling structured population in MB-MDR was already densely discussed in Chapter 4. An analysis of data on IBD disease was used to validate the methods. However, in the course of this PhD, other experiments were performed in our lab on this data and the most relevant ones are presented here. Since the IBD data is consortium one (a compilation of data from different institutions in different countries), population structure is bound to occur. Hence, this data is a good testbed for this section.

In [44], the gammaMAXT algorithm was used to analyze both the *discovery* and *replication* datasets described in Section 4.2.2. This analysis was performed at a time when the residuals-based and on-the-fly correction methods were not developed yet. Therefore, the approach adopted in the lab was similar to the path adopted by collaborators on this project, namely to use PCs as newly constructed variables that can then be taken into account in the analysis. For regression-based analysis this is obvious: one simply includes them as extra covariates in the model. However, even when adhering to a regression framework, it may not be possible to readily include covariates. It is highly dependent on properties of software packages. For instance, BOOST [131], one method of choice with our collaborators, cannot include covariates and thus PC adjustments need to be made via a two-stage analysis approach: screening with BOOST, and following up with classic logistic regressions (*glm* in R) that naturally allow for covariate adjustments. In [44], GenABEL was used to compute residuals based on the top 5 PCs. This analysis leads to 14 significant pairs of SNPs associated to Crohn's disease and 6 associated to ulcerative colitis. All SNP-pairs involved concomitant variants located on the same chromosome. For CD, they are located at 1p31.3, 5p13.1 and 16q12.1, susceptibility regions already discovered in Chapter 4. As explained in the latter, using PCs outside of the software package is suboptimal and thus more research followed, leading to the STRAT2 algorithm and the on-the-fly correction method.

Another experiment that was performed in the lab on the IBD data is interesting to mention, because it takes another approach than using PCs. This work was done in the context of the *BMC Bioinformatics 2013* paper [125], at a time when the STRAT1, STRAT2, residuals-based and on-the-fly correction methods were not developed yet. A real-life data analysis on the CD part of the IBD data was used [73, 5]. The idea of this work was mainly to discuss the speed performances of Van Lishout's implementation of maxT and provide more insight into the disease and the capabilities of the mbmdr-3.0.3.out software. A cleansing process on the CD was performed, giving rise to a set of 1687 unrelated Caucasians (676 CD patients and 1011 healthy controls) and 311,192 SNPs [125]. This experiment was performed before gammaMAXT was discovered and analyzing the whole dataset would have taken years. Therefore, Biofilter.0.5.1 [10] has been used as an additional data preparation step. It uses a knowledge-driven approach to prioritize genetic markers in gene-gene interaction screening while reducing the search space. In particular, Biofilter allows the explicit detection and modeling of interactions between a large set of SNPs based on biological information about gene-gene relationships and gene-disease relationships. The knowledge-based support for the models is attributed by implication index, which is simply a number of data sources that provide evidence of gene-gene interaction or gene-disease relationship, and is calculated by summing the number of data sources supporting each of the two genes and the connection between them (see [10] for more details).

In practice, to make the prioritization procedure in Biofilter more focused on CD, we applied a list of candidate genes for CD (120 genes collected from the publications [5, 99, 34, 62, 24]) and 160 groups (collected basing on selective search in Biofilter using keywords *crohn, enteritis, inflam, autoimmune, immune, bowel, gastrointest, ileum, ileitis, intestine, lleocolic, diarrhea, stenosis* and *cytokine*). Using this approach we ended up with 12,471 SNPs that we further analyzed in MB-MDR. Table 5.1 lists all the discovered statistically significant interactions.

Table 5.1: SNP-SNP interactions having a multiple testing corrected p-value < 0.05

Rank	first SNP	second SNP	p-value
1	rs11209026	rs7573680	0.004
2	rs11465804	rs7573680	0.017
3	rs11209026	rs2064689	0.018
4	rs11209026	rs6911639	0.021
5	rs11209026	rs4766584	0.023
6	rs11465804	rs2064689	0.025
7	rs11465804	rs4766584	0.028
8	rs11465804	rs6911639	0.029
9	rs11465804	rs10849401	0.033
10	rs11209026	rs296513	0.037
11	rs1343151	rs2076756	0.04
12	rs11209026	rs10849401	0.044
13	rs11209026	rs7786745	0.048
14	rs11209026	rs4655683	0.048

Note that the results of Table 5.1 are adjusted for testing 77 756 685 pairs of SNPs. Table 5.2 shows the genomic location of the SNPs involved in these significant SNP-SNP interactions.

Table 5.2: Location of the SNPs involved in a significant SNP-SNP interaction

SNP	Position	Gene
rs11209026	chr1:67705958	*IL23R*
rs11465804	chr1:67702526	*IL23R*
rs7573680	chr2:240169077	*HDAC4*
rs2064689	chr1:67653010	*IL23R*
rs6911639	chr6:32978178	*HLA-DOA*
rs4766584	chr12:109663581	*ACACB*
rs296513	chr1:200906473	*C1orf81*
rs10849401	chr12:6273238	intergenic
rs7786745	chr7:18422684	intergenic
rs4655683	chr1:67611613	intergenic
rs1343151	chr1:67719129	*IL23R*
rs2076756	chr16:50756881	*NOD2*

The following discussion of these results is mainly taken from my *BMC Bioinformatics* 2013 publication [125]. This biological interpretation was mainly written by Isabelle Cleynen, one of my co-authors on this paper. A total of 13 out of 14 significant interactions involve rs11209026 or rs11465804. Both SNPs are located in the interleukin-23 receptor (*IL23R*) gene, a known susceptibility gene for CD. The SNP rs11209026 is a non-synonymous coding SNP (Arg381Gln substitution), while rs11465804 is intronic and in strong linkage disequilibrium (LD) with rs11209026 (r^2=0.97). In the original GWA studies [73], rs11209026 and rs11465804 gave the most significant association signals with $p < 10^{-9}$. Given the strong association between the SNPs, it is to be expected that all interactions found for one SNP are also found for the other (Table 5.1). The most significant interaction is between rs11209026 and rs7573680 (p=0.004). The latter is an intronic SNP located in *HDAC4* (histone deacetylase 4). Figure 5.1 shows a Synergy Disequilibrium (SD) plot [133] for the SNPs listed in Table 5.1. Synergy is defined as the amount of information that can be obtained about the trait $\mathcal{T}$ by observing the pair of SNPs, minus the sum of the amounts of information that can be obtained about $\mathcal{T}$ by observing the two SNPs separately:

$$\mathcal{I}(SNP_{lj}, SNP_{rj}; \mathcal{T}) - [\mathcal{I}(SNP_{lj}; \mathcal{T}) + \mathcal{I}(SNP_{rj}; \mathcal{T})] \tag{5.1}$$

A positive synergy value is represented in red and suggests epistasis (a part of the information given by the two SNPs on the disease is attributable to a purely cooperative interaction between the two SNPs). A negative synergy value is represented in blue and indicates that the two SNPs are redundantly associated with the trait [133]. An SD plot is able to highlight disease-associated haplotypes, as well as epistatically interacting loci with respect to disease.

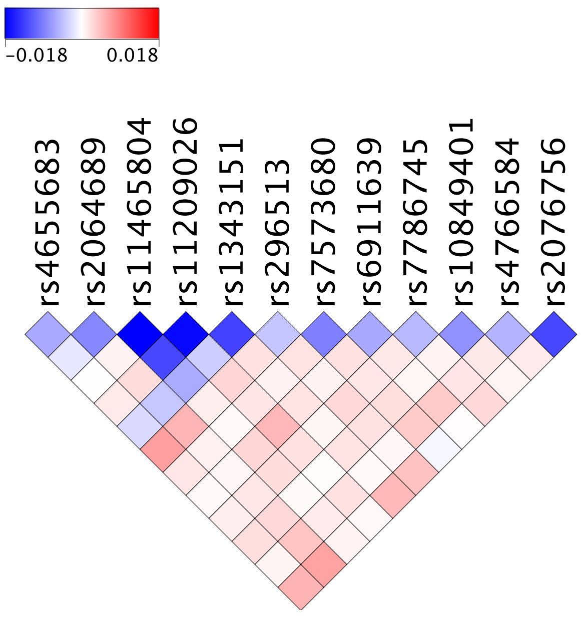


Figure 5.1: SD plot of potential epistasis interactions between the loci indicated in Table 5.1. Since there are no high-intensity red squares, strong epistasis is probably not present. This is inline with the fact that when we adjust the MB-MDR screen for main effects, no significant SNP pair is relevant [80].

Several studies have suggested that different signals exist in *IL23R*, conferring risk or protection to Crohn's disease. A study by Taylor et al [114], where they aimed to estimate the total contribution of the *IL23R* gene to CD risk using a haplotype approach, showed that the population attributable risk for these haplotypes was substantially larger than that estimated for the *IL23R* Arg381Gln variant alone. *MBMDR-3.0.3* identified several "epistatic" signals from pairs of SNPs located in the *IL23R* gene. It should be noted though that epistasis signals on SNPs in LD are considered to be non-synergetic.

The MB-MDR discoveries on Crohn's disease also seem to give us a new working hypothesis to expand on the current knowledge (histone deacetylation). Indeed, histone deacetylation results in a compact chromatin structure commonly associated with repressed gene transcription (epigenetic repression), and hereby plays a critical role in transcriptional regulation, cell cycle progression and developmental events. Although not known to physically interact directly, IL23R and HDAC4 could be linked trough MAPK1/STAT3 signaling: MAPK1 has been shown to associate with phosphorylate HDAC4 [143]. Protein phosphorylation regulates the corepressor activity of the deacetylase. MAPK1 also acts as an important activator of STAT3 (signal transducer and activator of transcription 3) which is an essential regulator of immune-mediated inflammation. In addition, the IL23/IL23R pathway modulates STAT3 transcriptional activity, and recently it has been shown that CD8+ T cells from Arg381Gln *IL23R* carriers showed decreased STAT3 activation compared with WT CD8+ T cells [104]. It can thus be hypothesized that a balanced action between the HDAC1/MAPK1 and IL23/IL23R pathways, converging on STAT3 signaling, are important for CD pathogenesis. To be complete, note that the new working hypothesis presented here (histone deacetylation) may not have clinical/biological significance since our conclusions are based on rs7573680, an intronic SNP located in HDAC4. Indeed, identification of gene-gene variants within coding regions is raither straightforward as they have impact on the protein sequences, whereas intergenic variants are less likely to have a direct effect. Only when they reside for instance in ultra conserved elements such as listed in the Vista enhancer database, they can interfere with transcription of genes and have clinical/biological significance [130].

This analysis shows that the methods of Chapter 2 allows to identify pairs of SNPs that may truly be associated to the disease. However, since the results are not corrected for unmeasured confounding factors, some hits may just be false positives. Furthermore, available covariates like gender and age were not taken into account. In this manuscript, we have intensively spoken about correcting for such factors. However, MB-MDR also allows to test if such factors directly modifies genetic effects or genetic interactions, as discussed in Chapter 2. Indeed, mbmdr-4.4.1.out cannot only perform GxG interaction analysis, but also GxE and GxGxE ones. These features have not been applied yet in this manuscript and an example of such an analysis is given in the next section.

5.3 Nature versus nuture

Non-genetic factors cannot be overlooked in GWAIs. They can be important predictors for disease-related traits on their own, but they can also modify genetic effects or genetic interactions. Therefore, in this section, we investigate the flexibility of our software to analyze GxE interactions. This work was done in the context of the *ASHG 2013* conference in Boston [121].

Using phenotypic and GWAS genotype data available through the Single nucleotide polymorphism Health association Asthma Resource Project (SHARP), we analyze the difference in pre-bronchodilator FEV1 in patients following or not ICS therapy for a period of 8 weeks, for 550 pediatric Caucasian CAMP (ages 5-12) subjects [112]. The trait of interest is *prech_short* expressed on a continuous scale and represents a relative difference in preFEV1. The environmental variable is dichotomous and refers to inhaled corticosteroids therapy (ICS) based on budesonide. If ICS is administrated it is coded 1 and 0 otherwise.

We analyze 550 samples containing no reported family structure. Missing genotypes were MaCH-imputed using 1000 Genomes Project Reference Panels resulting in 8,221,073 SNPs. Since the T-gene was found to be associated to asthma, a total of 50 SNPs found in the coding part of the T-gene (chr6:166,491,076-166,501,355) were added to the marker panel. Genotype data QC steps consisted of the following steps:

1) LD pruning (via PLINK) with maximum r^2 between-markers of 0.05, 0.2, 0.50, 0.75 and 0.8 respectively (LD pruning thresholds).
2) Removal of poorly annotated SNPs.
3) Removal of SNPs not present in the dbSNP database.
4) Removal of SNPs with MAF$<$ 0.05.
5) HWE at 10^{-8}.

Samples and their genotypes passing QC were extracted with PLINK. The final subset consisted, for instance, in the case of $r^2 = 0.2$ of 283,665 markers with population inflation factor λ=1.001 (minimal population stratification effects). Genetic Identity-by-State (IBS) kinship matrix was calculated using allelic frequency and applied as part of polygenic model. Trait residuals were computed from trait sex+age+BMI based on a polygenic regression model using observed kinships (GenABEL 1.7.6). The datasets were taken as input to mbmdr-4.0.1.out.

MB-MDR 2D analysis

We first perform a SNP-environmental interaction analysis with the command:

```
./mbmdr-4.0.1.out --continuous -f steroid_drug thedataset.txt
```

Tables 5.3-5.7 show the results when the LD pruning is respectively at 0.05, 0.2, 0.5, 0.75 and 0.8.

Table 5.3: Significant SNP-environment pairs found with LD pruning at 0.05

SNP	environment variable	MB-MDR statistic	p-value
rs1840125	steroid_drug	30.919	0.025
rs62385586	steroid_drug	30.574	0.03
rs7832479	steroid_drug	29.159	0.052

Table 5.4: Significant SNP-environment pairs found with LD pruning at 0.2

SNP	environment variable	MB-MDR statistic	p-value
rs10105588	steroid_drug	44.367	0.001
rs4361508	steroid_drug	37.700	0.003
rs6668694	steroid_drug	32.879	0.033

Table 5.5: Significant SNP-environment pairs found with LD pruning at 0.5

SNP	environment variable	MB-MDR statistic	p-value
rs10105588	steroid_drug	44.367	0.004
rs11123492	steroid_drug	43.814	0.004
rs12823753	steroid_drug	43.013	0.004
rs4361508	steroid_drug	37.700	0.019
rs6458016	steroid_drug	35.611	0.033
rs13126782	steroid_drug	35.007	0.041
rs3819722	steroid_drug	34.891	0.042

Table 5.6: Significant SNP-environment pairs found with LD pruning at 0.75

SNP	environment variable	MB-MDR statistic	p-value
rs11902232	steroid_drug	46.461	0.001
rs10105588	steroid_drug	44.367	0.003
rs11123492	steroid_drug	43.814	0.005
rs12823753	steroid_drug	43.013	0.006
rs4361508	steroid_drug	37.700	0.029
rs6458016	steroid_drug	35.611	0.05

Table 5.7: Significant SNP-environment pairs found with LD pruning at 0.8

SNP	environment variable	MB-MDR statistic	p-value
rs11902232	steroid_drug	46.461	0.001
rs10105588	steroid_drug	44.367	0.003
rs11123492	steroid_drug	43.814	0.003
rs12823753	steroid_drug	43.013	0.003
rs35480637	steroid_drug	39.605	0.011
rs4361508	steroid_drug	37.700	0.019
rs6458016	steroid_drug	35.611	0.049

We observe that none of the SNP-environment pairs found in Tables 5.4-5.7 appear in Table 5.3. This implies that either all results of Tables 5.4-5.7 are false-positives, which would be very surprising, or that pruning at 0.05 removes too much signal. The SNPs rs10105588 and rs4361508 are found in all analysis with an LD pruning value of 0.2 or more. These should be investigated from a biological point of view.

MB-MDR 3D analysis

The 2D analysis showed that pruning at 0.05 may remove too much (potentially important) SNPs. For this reason, we start this 3D analysis with a pruning at 0.2. We obtain the results from Table 5.8.

Table 5.8: Significant SNP-SNP-environment triplets found with LD pruning at 0.2

SNP1	SNP2	environment variable	MB-MDR statistic	p-value
rs61779746	rs12465925	steroid_drug	96.258	0.001
rs4894458	rs2076408	steroid_drug	92.422	0.001
rs13424105	rs661059	steroid_drug	80.651	0.005
rs7001245	rs11782903	steroid_drug	80.262	0.005
rs2490544	rs476646	steroid_drug	79.225	0.007
rs7001245	rs4434656	steroid_drug	79.157	0.007
rs28538891	rs13258014	steroid_drug	78.943	0.007
rs4776560	rs12951840	steroid_drug	78.742	0.008
rs61779746	rs10889296	steroid_drug	78.528	0.008
rs244404	rs11251120	steroid_drug	77.071	0.008
rs244404	rs7916097	steroid_drug	76.962	0.008
rs3098325	rs10415616	steroid_drug	76.770	0.008
rs1792325	rs8100069	steroid_drug	76.273	0.01
rs244404	rs11251133	steroid_drug	76.153	0.01
rs508370	rs759909	steroid_drug	75.674	0.011
rs2063995	rs6481397	steroid_drug	75.369	0.012
rs9850675	rs10982024	steroid_drug	74.423	0.016
rs56347641	rs7702195	steroid_drug	74.359	0.016
rs11585329	rs62164102	steroid_drug	74.258	0.016
rs10208913	rs739900	steroid_drug	73.303	0.023
rs2493929	rs1410936	steroid_drug	73.018	0.03
rs993908	rs12540872	steroid_drug	72.973	0.03
rs4879560	rs1410936	steroid_drug	72.906	0.031
rs2063995	rs12298457	steroid_drug	72.813	0.031
rs989312	rs55843044	steroid_drug	72.707	0.032
rs7625876	rs12892727	steroid_drug	72.636	0.032
rs61779746	rs4585362	steroid_drug	72.512	0.034
rs523395	rs35578731	steroid_drug	72.508	0.034
rs1580312	rs1432805	steroid_drug	72.502	0.034
rs4894458	rs2095343	steroid_drug	72.444	0.034
rs858549	rs7319633	steroid_drug	72.270	0.036
rs6544315	rs55870158	steroid_drug	72.185	0.038
rs9595453	rs12441091	steroid_drug	72.181	0.038
rs1990608	rs55870158	steroid_drug	72.043	0.042
rs6696341	rs9876895	steroid_drug	71.851	0.045
rs1432805	rs362509	steroid_drug	71.717	0.045
rs6820227	rs28434700	steroid_drug	71.648	0.045
rs2490544	rs13119846	steroid_drug	71.628	0.046
rs12711727	rs9705930	steroid_drug	71.393	0.047

We observe that the 3D analysis already leads to a lot of significant results with LD pruning at 0.2. We observe that the SNPs rs10105588 and rs4361508 found in the 2D analysis do not appear in this list. We therefore assume that these are directly interacting with steroid_drug. The fact that the number of 3D results is so important is difficult to interpret. An attempt is that the biology of the disease is more complex than direct interactions and that 3D analysis opens the door to see some parts of a complex graph network behind the disease. For instance, the interaction of several SNPs with steroid_drug could be required to start some biological processes, not starting when only a single SNP is present. Note that the fact that there are much more 3D interaction candidates than 2D ones can also play a role here. The results with higher values of LD pruning than 0.2 are not shown, since they also lead to a lot of significant results and are similarly difficult to interpret than the ones from Table 5.8.

In this section, we have shown that our software is easy to use to conduct a GxE or GxGxE interaction analysis. This is an asset compared to many other interaction software. Another edge of the software is that it adequately takes care of missingness. Chapter 2 describes an optimal strategy to take account of missing SNP values. Furthermore, censored data can be seen as a form of missingness as well and are handled flexibly in our software. However, this was not demonstrated on data yet in this thesis and an example of such an analysis is given in the next section.

5.4 Censoring

Censored traits are omnipresent in clinical practice (time to disease progression, time to death, time to first relapse, ...). Not so many tools exist that can rapidly and efficiently carry out a GWAIs with varying numbers of factor levels. One possibility is randomForestSRC [56]. Random Forests have indeed been discussed as an alternative method to detect interactions [138]. Alternatively, a kernel machine Cox regression framework is employed in [74]. In the versions of our software up to mbmdr-4.3.2.out (included), the survival part was solely based on the logrank test. Due to the interesting properties of Cox regression, we extended our software to allow users to adopt the Cox model to correct for main effects (see Section 2.4.2) and covariates (see Section 4.3.1). The work presented in these sections is in fact a very recent, unpublished one.

A simulation study was conducted to analyze the power and type I error rate of mbmdr-3.0.0.out in the case of a trait expressed on a survival scale, based on the logrank test. This contribution has been accepted for oral presentation at the *ERCIM 2012* conference in Oviedo [126]. The effect of different parameters is measured: the hazard ratio (2, 4 or 6), the minor allele frequency (0.2 or 0.4), the censoring rate (20% or 50%), the heritability (0.2 or 0.4) and the number of individuals (200, 400, 800 or 1600). This gives rise to a total of $3 \times 2 \times 2 \times 2 \times 4 = 96$ different settings. For each possible setting, 100 datasets are generated. We create the new datasets starting from the balanced case-control datasets (consisting of 1000 attributes) at

`http://discovery.dartmouth.edu/epistatic_data/`.

To transform this data for which the trait is expressed on a binary scale into a survival one, we generate a new trait value from a Weibull(a_0, b_0) for the former controls and from a Weibull(a_1, b_0) for the former cases [134]. The parameter a_1 is selected in such a way that the hazard ratio (HR) matches the prespecified values. We generate the censoring variable according to a uniform distribution $U[0, c_{max}]$. The parameter c_{max} is selected so that the censoring approximately matches the pre-specified values. Tables 5.9-5.12 shows the power results, i.e. the percentage of times that the causal pair is found, in the different scenarios.

Table 5.9: Percentage of times that the causal pair is found (200 individuals)

200 individuals		MAF=0.2 h^2=0.2	MAF = 0.2 h^2=0.4	MAF = 0.4 h^2=0.2	MAF = 0.4 h^2=0.4
Cens=20%	HR=2	1%	18%	1%	6%
Cens=20%	HR=4	11%	79%	21%	43%
Cens=20%	HR=6	30%	99%	40%	78%
Cens=50%	HR=2	1%	8%	0%	1%
Cens=50%	HR=4	7%	61%	10%	30%
Cens=50%	HR=6	28%	91%	24%	51%

Table 5.10: Percentage of times that the causal pair is found (400 individuals)

400 individuals		MAF=0.2 h^2=0.2	MAF = 0.2 h^2=0.4	MAF = 0.4 h^2=0.2	MAF = 0.4 h^2=0.4
Cens=20%	HR=2	8%	48%	10%	25%
Cens=20%	HR=4	51%	99%	72%	95%
Cens=20%	HR=6	74%	100%	90%	100%
Cens=50%	HR=2	2%	20%	2%	12%
Cens=50%	HR=4	38%	96%	51%	81%
Cens=50%	HR=6	68%	100%	83%	98%

Table 5.11: Percentage of times that the causal pair is found (800 individuals)

800 individuals		MAF=0.2 h^2=0.2	MAF = 0.2 h^2=0.4	MAF = 0.4 h^2=0.2	MAF = 0.4 h^2=0.4
Cens=20%	HR=2	23%	83%	38%	71%
Cens=20%	HR=4	93%	100%	99%	100%
Cens=20%	HR=6	99%	100%	100%	100%
Cens=50%	HR=2	10%	65%	23%	46%
Cens=50%	HR=4	84%	100%	95%	100%
Cens=50%	HR=6	98%	100%	100%	100%

Table 5.12: Percentage of times that the causal pair is found (1600 individuals)

1600 individuals		MAF=0.2 h^2=0.2	MAF = 0.2 h^2=0.4	MAF = 0.4 h^2=0.2	MAF = 0.4 h^2=0.4
Cens=20%	HR=2	79%	99%	91%	100%
Cens=20%	HR=4	100%	100%	100%	100%
Cens=20%	HR=6	100%	100%	100%	100%
Cens=50%	HR=2	45%	95%	73%	95%
Cens=50%	HR=4	100%	100%	100%	100%
Cens=50%	HR=6	100%	100%	100%	100%

The most obvious observation is that power increases when the number of subjects rises. Datasets containing only 200 subjects are too small to reach a good power, except when the heritability and hazard ratio are both very strong. Doubling the number of individuals already improves the results significantly, but the best results are of course observed with 1600 individuals. Recall that the runs have all been performed with the default option of the software, i.e. the logrank test. At the time of this publication, the Cox model was not implemented yet. Anyway, running all these analyses with the latter would not have been possible, because the overall computing times would have required years. Tables 5.13-5.16 shows the FWER results, i.e. the percentage of times that at least one non causal pair is incorrectly found, in the different scenarios.

Table 5.13: Percentage of datasets leading to at least one false-positive (200 subjects)

200 individuals		MAF=0.2 h^2=0.2	MAF = 0.2 h^2=0.4	MAF = 0.4 h^2=0.2	MAF = 0.4 h^2=0.4
Cens=20%	HR=2	1%	7%	7%	5%
Cens=20%	HR=4	5%	5%	8%	4%
Cens=20%	HR=6	5%	2%	8%	6%
Cens=50%	HR=2	3%	2%	6%	2%
Cens=50%	HR=4	2%	9%	7%	2%
Cens=50%	HR=6	5%	0%	4%	6%

Table 5.14: Percentage of datasets leading to at least one false-positive (400 subjects)

400 individuals		MAF=0.2 h^2=0.2	MAF = 0.2 h^2=0.4	MAF = 0.4 h^2=0.2	MAF = 0.4 h^2=0.4
Cens=20%	HR=2	3%	5%	8%	5%
Cens=20%	HR=4	7%	1%	7%	7%
Cens=20%	HR=6	2%	1%	5%	3%
Cens=50%	HR=2	1%	3%	1%	2%
Cens=50%	HR=4	6%	6%	4%	2%
Cens=50%	HR=6	8%	10%	6%	3%

Table 5.15: Percentage of datasets leading to at least one false-positive (800 subjects)

800 individuals		MAF=0.2 h^2=0.2	MAF = 0.2 h^2=0.4	MAF = 0.4 h^2=0.2	MAF = 0.4 h^2=0.4
Cens=20%	HR=2	2%	6%	9%	4%
Cens=20%	HR=4	5%	2%	3%	6%
Cens=20%	HR=6	7%	4%	8%	6%
Cens=50%	HR=2	4%	7%	3%	6%
Cens=50%	HR=4	6%	9%	3%	5%
Cens=50%	HR=6	1%	1%	4%	1%

Table 5.16: Percentage of datasets leading to at least one false-positive (1600 subjects)

1600 individuals		MAF=0.2 h^2=0.2	MAF = 0.2 h^2=0.4	MAF = 0.4 h^2=0.2	MAF = 0.4 h^2=0.4
Cens=20%	HR=2	7%	3%	5%	7%
Cens=20%	HR=4	7%	6%	6%	4%
Cens=20%	HR=6	5%	4%	3%	5%
Cens=50%	HR=2	6%	6%	4%	6%
Cens=50%	HR=4	8%	7%	10%	10%
Cens=50%	HR=6	3%	2%	4%	2%

In most of the case, we note that the estimated rates are below or within the interval $[2,5\% - 7,5\%]$ and satisfies thus Bradley's liberal criterion of robustness for the significance level $\alpha = 5\%$ [8]. However, since we work with only 100 datasets per settings (working with 1000 datasets to avoid this issue would have required months), it is not a surprise that some values are outside of this range. The averages of tables 5.13-5.16 (respectively 4.6, 4.4, 4.7 and 5.4) are nonetheless all inside the range. In the next section, we discuss the interpretability of the results found by our software.

5.5 Interpretation and visualization

In terms of deriving biological interpretations or formulating novel hypotheses, extra value can be obtained by constructing statistical epistasis networks. However, as we have shown throughout the thesis, it is important to interpret statistical epistasis findings in the context of potential marginal effects. When an ordered list of significant SNP pairs after an MB-MDR run shows an aggregation of 1 or a few SNPs, it may point towards the importance of the SNP or a problem with the main SNP effect not adequately being removed from the signal. That is why we created MoG-Plot (Manhattan over Grail plot), an R tool used internally to visualize main-effect and interaction results at a glance. The generated figure also localize the genes on their chromosomes, facilitating interpretability. It was based on a poster presentation of Ben Grady at the ASHG in which he presented a schematic "solar" plot of negative log p-values of results from a primary case–control analysis of mtDNA variants associated with CD4 count change ≥ 100 at 48 weeks [42]. Mitochondrial DNA is naturally represented in circular form but it inspired us to also use this representation to overlay results from a GWAs with results from an MB-MDR epistasis runs. Interestingly, independent from our efforts, another group also had the idea to use such circular representations to visualize connections between disease loci [98]. GRAIL plots (as they called their visualizations) are primarily based on evidences from the literature, whereas our plots are based on evidence from an epistasis screen on real-life data.

The R code of MoG-Plot takes as argument two files:

- a file containing a list of SNPs and the p-values representing their main effects for a particular disease
- a file containing a list of pairs of SNPs and the p-values corresponding to their interaction effects for the same disease

The R function produces a graphic that is a mixture of a Manhattan plot (except that it is presented on a circle) and a grail plot[98]. This allows to study visually the relation between the main effects and the interactions. By default, the files are in MB-MDR format, but the code is trivial to adapt to other formats. MoG-Plot requires the R libraries SNPlocs.Hsapiens.dbSNP.20120608, biomaRT, GenomicFeatures and GenomicRanges. These libraries can be download at bioconductor [40]. The program first constructs a database of the transcripts/genes based on the human genome version 19. This step needs to be executed only once, the database can then be used to generate as many figures as needed. MoG-Plot associated each SNPs of the input files to their chromosome number and position. The p-values of the first input file are changed on a -log(10) scale for the manhattan-like circular part of the figure.

In order to highlight the graphical capacities of MoG-Plot, we analyze two files constructed from the real-life data on Crohn's disease described in Section 5.2. The first file has been obtained by running a main effect analysis with mbmdr-4.0.1.out on the reduced dataset composed of 12,471 SNPs described in Section 5.2. It contains the 12,471 SNP names and the corresponding p-values. The second file has been obtained by running an interaction analysis with mbmdr-4.0.1.out on the same dataset. It contains the names of the top 1000 pairs of SNPs obtained at the end of this analysis and the corresponding p-values. Figure 5.2 gives the output of MoG-Plot.

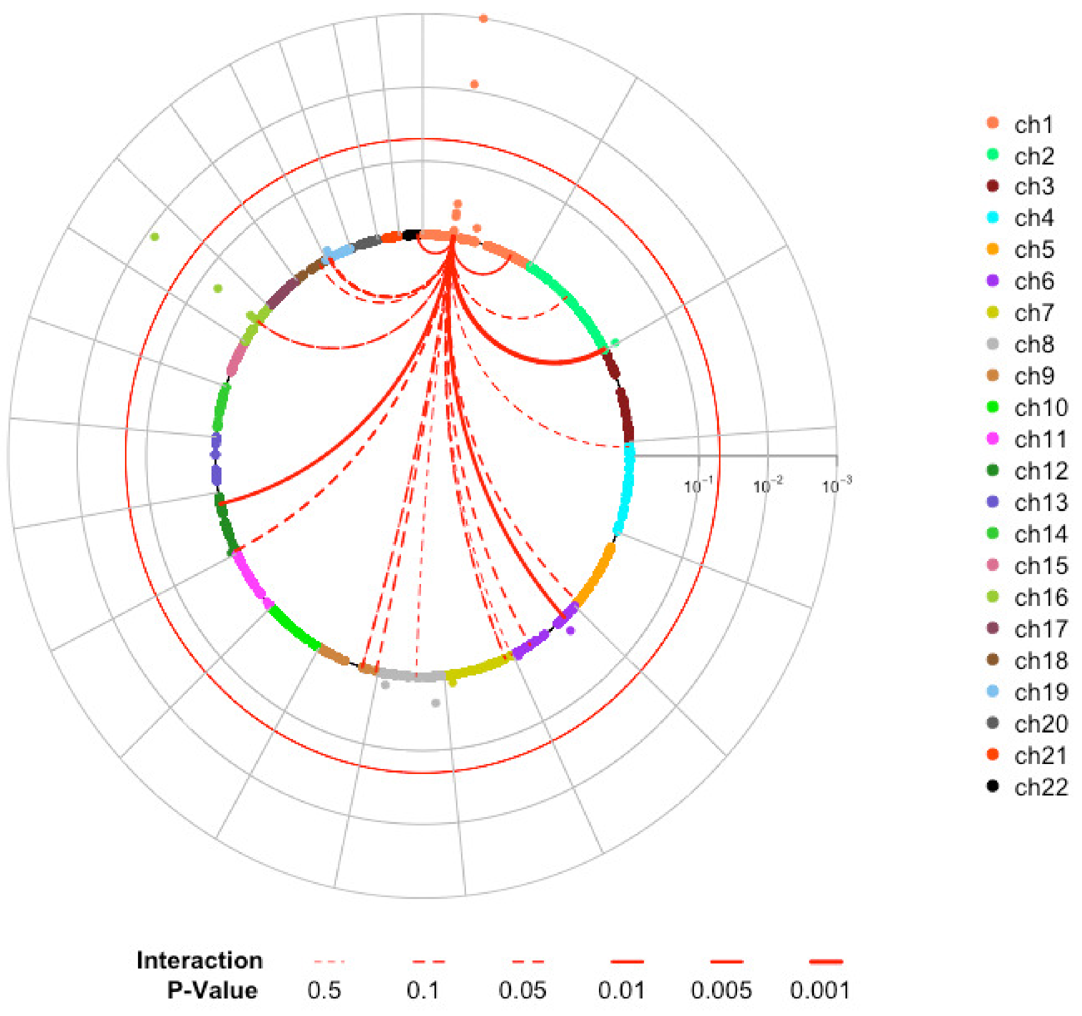


Figure 5.2: This figure is a mixture of a Manhattan-like circle plot and a grail plot. Only three dots are outside of the red line, meaning that only three SNPs have a significant main effect. Two of these SNPs are located on chromosome 1, while the third is on chromosome 16. The interactions seems to be driven by the main effects of the two first SNPs, since they all start from this location.

The total execution time of MoG-Plot in this case was about 15 minutes on a 2.4 GHz Intel Core 2 Duo processor. However, the majority of the time was spent in the construction of the transcripts/genes database, which must be executed only once.

Most of the SNPs have no main-effect at all, so that most dots are on the smallest circle of the figure. The 5% significant threshold is represented by a red circle. One can see that only 3 SNPs have a significant main-effect, two from chromosome 1 and one from chromosome 16. From the interaction-part of the figure, one can recognize that chromosome 1 plays a crucial role. All interactions starts from about the same position on this chromosome and seems to be driven by strong main effects, represented by the orange dots.

The over-representation of SNPs in top SNP pairs outputted by MB-MDR (or any epistasis analysis run) is crucial to rule out significant higher-order interactions that may be caused by strong lower order components. In addition, other elements in a check-list should be addressed, when interpreting the results of an epistasis analysis. To illustrate this, we present very recent results of an analysis on Coronary artery disease (CAD). In the field of statistical genetics, many research center own data that cannot be shared with third parties. In this context, researchers use our software and contact us to give them indications on the best options to use. Usually, they send us results where the SNP names have been replaced by dummy ones (snp1, snp2, ...). In this way, they can keep the results secret until publication.

The group of Inke König [66, 41] did such an analysis with mbmdr-4.3.3.out on CAD data, in collaboration with our lab. Five datasets are at our disposal, a discovery and four replications ones. The former consists of 2281 subjects (637 cases and 1644 controls) for which 193 711 SNPs are available (after QC). The parallel workflow of gammaMAXT was used to analyze it, with the default options of the software, except that the minimal number of individuals in a group to be significant (the *-m* option) was set to 50 and the number of permutation (the *-p* option) was set to 9999. This reduces the probability of finding sporadic results. This analysis allowed to identify 67 significant pairs of SNPs out of the 18 761 878 905 candidates that were considered!

To go further, the residuals-based correction method described in Section 4.3.1 was used on the same dataset to take account of gender and age. This analysis identified 49 significant pairs of SNPs. The replication datasets consists respectively of 1802 subjects (735 cases and 1067 controls), 2150 subjects (701 cases and 1449 controls), 2127 subjects (990 cases and 1137 controls) and 3919 subjects (2389 cases and 1570 controls). Instead of taking all the SNPs of the replication datasets into account, we focus on the 86 ones that belongs to a pair identified on the discovery dataset, either with or without taking the covariates into account (the *-k* option enables to focus solely on these SNPs). Without taking the covariates into account, 15 pairs out of the 67 previously identified ones could be replicated in 3 out of 4 replication datasets and are given in Table 5.17.

These results look promising from a biological point of view after discussion with cardiologists. Many of the interesting results are close to each other. When gender and age are taken into account, 13 pairs out of the 49 previously identified ones could be replicated in 3 out of 4 replication datasets and are given in Table 5.18. We note that all pairs appearing in Table 5.18 also appear in Table 5.17. This suggests that the pairs from Table 5.17 not appearing in Table 5.18 anymore may be false-positive results. One of the covariate or the combination of both may be a confounding factor between the genetics and the disease. The next step in such an analysis, is to translate the SNP-pair list into a gene-pair list and see whether there is evidence from expression data (co-expression networks), or from software such as biofilter (that give implication factors for gene-gene pairs based on several biological data bases, see Section 5.2). What is also often done in our lab is to uses gene ontology (GO) to see whether there is an over-representation of pathways in the gene list that is derived from the sign gene pair list. Finally, further functional analysis can be performed in the lab to actually investigate co-localization of the corresponding proteins.

Table 5.17: Significant SNP-SNP pairs found for CAD, without covariate correction

pair of SNPs		discovery		replication 1		replication 2		replication 3		replication 4	
first	second	stat	p-value	stat	p-value	stat	p-value	stat	p-value	stat	p-value
snp1	snp2	220.4	0.0001	103.2	0.0001	290	0.0001	123.3	0.0001	0	1
snp2	snp3	191.2	0.0001	81.4	0.0001	228.5	0.0001	157.7	0.0001	0	1
snp4	snp5	197.5	0.0001	70	0.0001	227.8	0.0001	0	1	376.7	0.0001
snp6	snp5	164.9	0.0001	65	0.0001	199.9	0.0001	0	1	303.1	0.0001
snp5	snp7	185.3	0.0001	63.7	0.0001	209.7	0.0001	3.2	1	360.8	0.0001
snp8	snp5	137.1	0.0001	59.1	0.0001	155.9	0.0001	0	1	215.8	0.0001
snp2	snp9	186.1	0.0001	52.1	0.0001	217.2	0.0001	126.1	0.0001	0	1
snp5	snp10	137.1	0.0001	49.7	0.0001	150	0.0001	0	1	217.1	0.0001
snp2	snp11	173	0.0001	40	0.0001	174.2	0.0001	124.8	0.0001	0	1
snp12	snp5	69.6	0.0001	37.2	0.0001	109	0.0001	0	1	104.1	0.0001
snp13	snp5	90.8	0.0001	31.2	0.0003	81	0.0001	0	1	112.1	0.0001
snp2	snp14	142.5	0.0001	28.3	0.0009	144.8	0.0001	108.5	0.0001	0	1
snp15	snp5	63.2	0.0002	19.8	0.0817	74.8	0.0001	0	1	57.8	0.0001
snp16	snp17	52.7	0.0314	20.1	0.0679	37.3	0.0001	11.4	0.9992	73.6	0.0001
snp2	snp18	140.7	0.0001	19.1	0.1161	158	0.0001	107.2	0.0001	0	1

Table 5.18: Significant SNP-SNP pairs found for CAD, corrected for gender and age

pair of SNPs		discovery		replication 1		replication 2		replication 3		replication 4	
first	second	stat	p-value	stat	p-value	stat	p-value	stat	p-value	stat	p-value
snp4	snp5	186,9	0,0001	55,6	0,0001	236,3	0,0001	0	1	259,2	0,0001
snp6	snp5	151,6	0,0001	55,4	0,0001	209,7	0,0001	0	1	212,8	0,0001
snp5	snp7	175,0	0,0001	49,4	0,0001	213,6	0,0001	4,5	1	250,3	0,0001
snp5	snp10	123,9	0,0001	37,3	0,0001	153,2	0,0001	3,0	1	134,0	0,0001
snp8	snp5	123,0	0,0001	45,4	0,0001	170,2	0,0001	0	1	127,9	0,0001
snp1	snp2	169,3	0,0001	71,9	0,0001	310,6	0,0001	112,9	0,0001	0	1
snp2	snp3	159,6	0,0001	54,6	0,0001	274,0	0,0001	146,4	0,0001	0	1
snp2	snp11	148,8	0,0001	49,2	0,0001	208,3	0,0001	111,8	0,0001	0	1
snp2	snp9	189,1	0,0001	57,3	0,0001	251,9	0,0001	115,4	0,0001	0	1
snp12	snp5	67,8	0,0002	31,0	0,0005	101,0	0,0001	0	1	45,3	0,0001
snp13	snp5	78,4	0,0001	27,4	0,0022	80,1	0,0001	0	1	54,6	0,0001
snp2	snp14	115,7	0,0001	18,4	0,1602	169,2	0,0001	94,5	0,0001	0	1
snp15	snp5	53,2	0,0282	15,4	0,5724	77,1	0,0001	0,0	1	32,7	0,0002

5.6 Discussion

In Chapter 4, we have demonstrated the different methods on real-life data on IBD. In this chapter, we have described several application of our software, both on real-life and simulated data. Furthermore, we have introduced a tool facilitating interpretation of the results. Obviously, more elaborate such tools are available on the internet. However, the tool presented in this thesis is customized for MB-MDR results and allows to produce nice graphics quickly, without having to learn a new software.

This chapter has shown the relevance of the methods introduced in this thesis. Flexibility is probably the main asset of `mbmdr-4.4.1.out`, as should be clear after this chapter where analyses of very different kinds were performed. Users are invited to try different combinations of options, to get new insights on the data that they have at hand. We advocate starting with the default options and depending on the results, make more advanced analysis or not. Of course, if covariates are available, trying to take them into account is a must. For this, two strategies are possible. First, the covariates are considered as environmental factors and 3D interactions are searched for, considering every possible combination of two SNPs and one environmental factor. Second, the covariates are just corrected for, in order to identify pure gene-gene interactions. To achieve this, we advocate on-the-fly correction if the covariate is a categorical one, which does not have too much different categories (otherwise, computing time becomes quickly an issue). Otherwise, we advocate residuals-based correction. Working with a two-stage analysis design is also an interesting strategy to reduce the number of SNP pairs to further investigate from a biological point of view. Using STRAT2 at stage two is a good strategy to correct automatically for unmeasured confounding factors.

Chapter 6

Conclusions and perspectives

6.1 Conclusions

The main contribution of this PhD thesis is gammaMAXT, a novel implementation of the maxT algorithm for multiple testing correction. The algorithm was implemented in software `mbmdr-4.4.1.out`, as part of the MB-MDR framework to screen for SNP-SNP, SNP-environment or SNP-SNP-environment interactions at a genome-wide level. In this context, we analyzed a dataset composed of one million SNPs and thousands individuals within one day on a 256-core computer cluster. The same analysis would have taken more than a year with Van Lishout's implementation of maxT, another contribution of this PhD, which is already an improvement of the classic Westfall and Young implementation.

These results are promising for future GWAIs. Furthermore, the proposed gammaMAXT algorithm offers a general significance assessment and multiple testing approach, applicable to any context that requires performing hundreds of thousands of tests. It offers new perspectives for fast and efficient permutation-based significance assessment in large-scale (integrated) omics studies. The manual of our software is given in the Appendix and contains a description of all the options and a history of the main changes in public versions. The `mbmdr-4.4.1.out` software can be downloaded freely at `http://www.statgen.ulg.ac.be` with the corresponding documentation.

In order to reduce the amount of false discoveries, population stratification and covariates are important to correct for. A novel application of existing methods to take account of this problematic was provided, leading to several contributions. STRAT1 and STRAT2 are two algorithms similar to maxT, but that in addition correct automatically for unmeasured confounding factors. Since STRAT1 consistently leads to the same ranking as maxT (yet with different p-values), whereas STRAT2 does not, we systematically advocate using the latter to correct for unmeasured confounding factors automatically. Residuals-based and on-the-fly correction are two methods to correct for covariates. The former is based on the classical idea, consisting in computing residuals based on a linear model including the trait and the covariates, to finally use these residuals as a new outcome variable. However, this method is not suitable when covariates are associated with the SNPs and also lead to supplementary limitations. Therefore, the on-the-fly correction method was introduced, whose idea is to modify the MB-MDR statistic computations that were presented in Chapter 2 to correct for the main effects of the SNPs, to additionally correct for the covariates.

A real-life dataset on IBD disease was used to validate all these novel methods. This analysis has lead to interesting biological results. Some of them were already known in the literature and others may be new discoveries. This demonstrates the ability of mbmdr-4.4.1.out to handle state-of-the-art datasets in an appropriate way. However, one cannot draw definitive conclusions on the power of novel methods solely based on a real-life data analysis. Clearly, an investment is needed in extensive simulation studies, where we can have full control over factors affecting disease trait to understand the assets and limits of the new methods. However, generating realistic synthetic data is far from trivial, since inducing the situation of confounding implies creating scenarios in which non-linear patterns are distributed differently between the populations and at the same time generating different disease prevalence between the populations. To our knowledge, no such tool exists. In replacement, our software was applied to both simulated and real-life datasets, to learn from data.

This work has lead to several conclusions. First, using principal components as newly constructed variables that can then be taken into account in the analysis is a promising idea in the context of MB-MDR, especially in combination with the on-the-fly correction method. Second, non-genetic factors cannot be overlooked in GWAIs and MB-MDR is an easy-to-use tool to search for GxE and GxGxE interactions. Third, censored traits are omnipresent in clinical practice and our software can rapidly and efficiently carry out a GWAIs with varying number of factor levels in this context. Finally, facilitating interpretability of the results is an important aspect and a tool for visualizing main effect and pairwise-interaction results at a glance was proposed.

6.2 Perspectives

A first perspective of this PhD is to write a modified version of the on-the-fly correction methods. The current architecture of the software is based on the idea that the multiple-testing correction algorithms require a permutation of the trait values, not the SNPs and covariates ones. Nonetheless, in the context of the on-the-fly correction, keeping the covariates values together with the trait ones would be an interesting idea. This would allow to test another null hypothesis than with the current implementation. Namely, one where the link between the trait and the covariates has not been broken.

We have already mentioned that an investment is needed in extensive simulation study to evaluate the new methods correcting for population stratification and covariates. A perspective of this PhD is therefore to write a tool for generating synthetic data in a flexible way, in which the situation of confounding has been induced in a non-linear way, with different disease prevalence in subpopulations. Such a tool could possibly identify the limitations of the methods and lead to improved versions. Note that the basic idea of STRAT2 guarantees control of the FWER. However, some accommodations had to be done in order to make the algorithm work in practice, leading to a pragmatic implementation that does not guarantee it anymore. Nonetheless, most of these accommodations were linked with the fact that STRAT2 should be able to be readily used on genome-wide data. Since STRAT2 was usually used at the second stage of a discovery-replication analysis in this PhD (on a reduced dataset), a perspective would be to write an alternative implementation of STRAT2 dedicated to small datasets and guaranteeing control of the FWER.

Finally, an obvious perspective of this PhD work is to apply the software on real-life data. Indeed, a set of novel methods to perform interaction analysis studies in a flexible way is described in this thesis, but these methods have not yet been intensively used in practical analysis. In particular, the survival part of the software was so far only put into use on simulated data. Since survival data are not taken into account by many concurrent software tools, this is really an asset of mbmdr-4.4.1.out that should be put forward. Hence, some marketing work needs to be done, in order to create new collaborations with labs having genome-wide data available for analysis.

Appendix - Manual

mbmdr-4.4.1.out is a software that is able to detect multiple sets of significant gene-gene and/or gene-environment interactions in relation to a trait of interest, while efficiently controlling type I error rates. The trait can be expressed either on a binary or a continuous scale, or as a censored trait. To see the command line help, type

```
./mbmdr-4.4.1.out help
```

The instructions to run mbmdr-4.4.1.out are (depending on the data type) as follows:

```
./mbmdr-4.4.1.out --binary [options] 'mbmdrFile'
./mbmdr-4.4.1.out --continuous [options] 'mbmdrFile'
./mbmdr-4.4.1.out --survival [options] 'mbmdrFile'
```

If the data is expressed on a binary or continuous scale, then the 'mbmdrFile' must be represented using the following structure (for censored trait see −−help −−survival)

```
Tr1  ...  Trx  Cv1  ...  Cvy  Ma1  ...  Maz
T11  ...  T1x  C11  ...  C1y  M11  ...  M1z
...  ...  ...  ...  ...  ...  ...  ...  ...
Tk1  ...  Tkx  Ck1  ...  Cky  Mk1  ...  Mkz
```

The first line is a title line: the Trj's are the names of the x traits ($x \geq 1$), the Cvj's are the names of the y covariates ($y \geq 0$) and the Maj's are the names of the z markers, i.e. SNPs and/or environment variables ($z \geq 2$).

The first x columns contain the trait values: in the binary case, Tij is 1 if the i^{th} subject is a case for the j^{th} trait and 0 if it is a control; in the continuous case Tij is a continuous value representing the state of the i^{th} subject for the j^{th} trait. Usually, there is only a single trait (x=1). The next y columns are covariate values (missing values are not allowed).

The last z columns are markers values (missing values must be coded '-9'):

- if Maj is a SNP: Mij is 0 if the i^{th} subject is homozygous for the first allele, 1 if heterozygous and 2 if homozygous for the second allele.
- if Maj is an environment variable: the X different possible values of the environment variables should be coded 0, 1, ..., X-1.

If the dataset is in PLINK format, one can first use the following command line to create the 'mbmdrFile':
(replace −−binary by −−continuous or −−survival depending on your trait)

```
./mbmdr-4.4.1.out --plink2mbmdr --binary -ped 'pedFile' -map 'mapFile'
-o 'mbmdrFile' -tr 'trFile'
```

The file 'trFile' is an output file giving the chosen labels for the genotypes of each SNP. The 'pedFile' must contain a title line (insert a dummy line at the beginning of your file if your file does not have a title line). See −−help −−plink2mbmdr for more options. Note that if you have a 'pheFile' the header has to be *ID sex trait cov*1 *cov*2 . . .

The different options of the program are[1]:

[$-n$ *INT*]	number of top pairs in the output (default: 1000)
[$-p$ *INT*]	permutation amount for multiple-testing (default: 999)
[$-r$ *INT*]	random seed parameter (default: random value)
[$-m$ *INT*]	minimum group size to be statistically relevant (default: 10)
[$-at$ *INT*]	amount traits (default: 1)
[$-ct$ *INT*]	current trait (default: 1)
[$-ac$ *INT*]	amount covariates (default: 0)
[$-x$ *DOUBLE*]	cutoff value for the statistical test (default: 0.1)
[$-mt$ *STRING*]	multiple testing correction algorithm: NONE, MAXT, MINP, RAWP, STRAT1, STRAT2 or gammaMAXT (default)
[$-rc$ *STRING*]	regress covariates: RESIDUALS (default), ONTHEFLY
[$-o$ *STRING*]	output file name (default: 'inputprefix'_output.txt)
[$-o2$ *STRING*]	models file name (default: 'inputprefix'_models.txt)
[$-a$ *STRING*]	adjust: CODOMINANT (main default), ADDITIVE, ONESTEP or NONE (survival default)
[$-d$ *STRING*]	dimension of interactions: 1D, 2D (default) or 3D
[$-pb$ *STRING*]	progress bar: NONE or NORMAL (default)
[$-v$ *STRING*]	verbose in models file: SHORT, MEDIUM (default) or LONG
[$-if$ *STRING*]	input format: MBMDR (default), MDR or ISALIVE
[$-e$ *LIST*]	erase markers (LIST: comma-separated list of marker names)
[$-E$ *FILE*]	erase markers (FILE: composed of one marker name per line)
[$-k$ *LIST*]	keep only the markers from the comma-separated list
[$-K$ *FILE*]	keep only the markers from the file
[$-s$ *FILE*]	second stage of a discovery-replication analysis: keep only the pairs from the given output FILE of the first stage
[$-f$ *LIST*]	filter: analyze only the pairs composed of exactly one marker from the given comma-separated list of marker names
[$-F$ *FILE*]	filter: analyze only the pairs composed of exactly one marker from the given file and one marker from the input file
[$-rt$ *STRING*]	rank transformation (continuous trait only): NONE (default), RANK_TRANSFORM or RANK_TRANSFORM_TO_NORMALITY

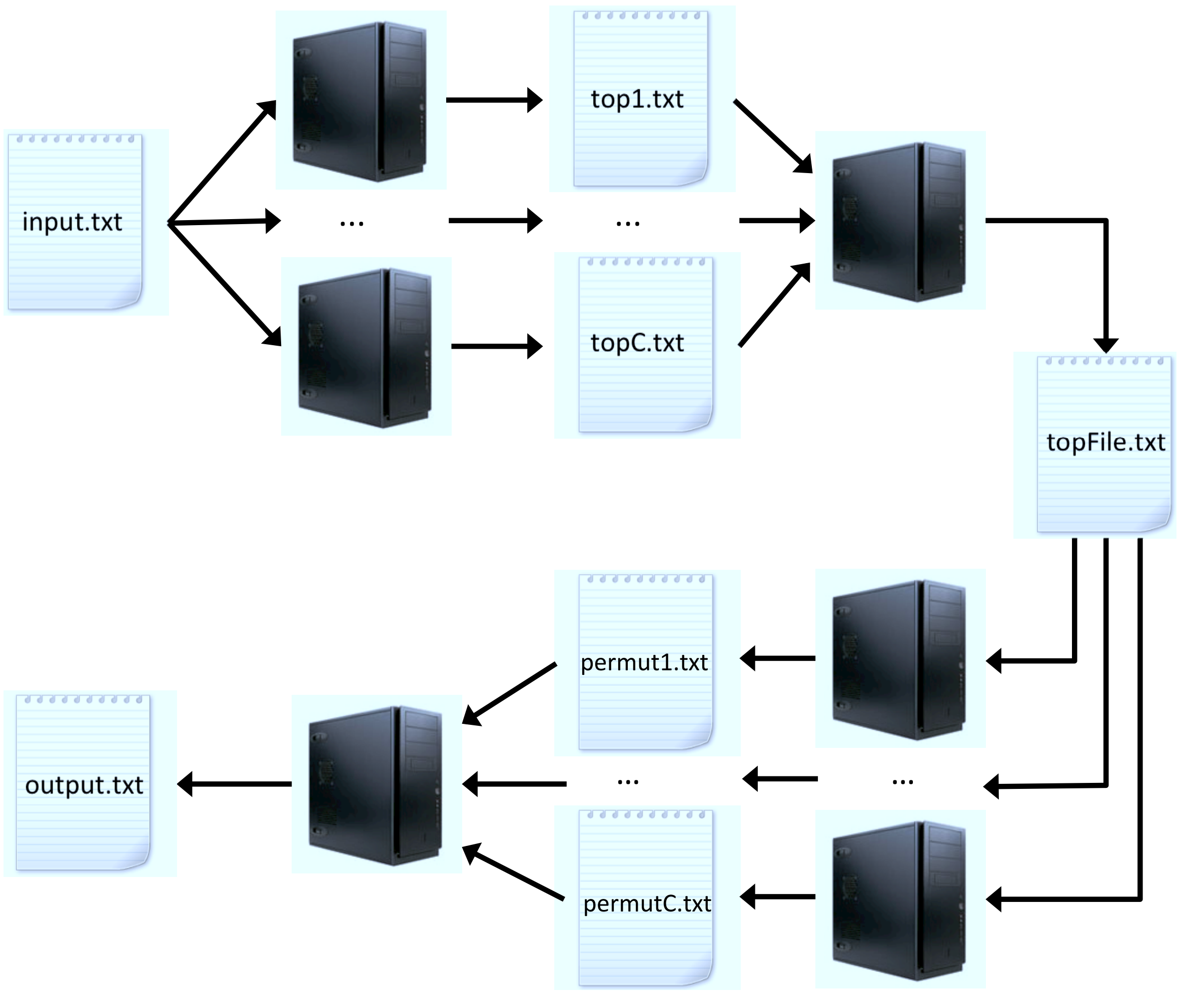
[1] In this manual, the options that are not mandatory are put between square brackets

Parallel Workflow

Users analyzing large-scale datasets should use the gammaMAXT parallel workflow. (to consult the online manual see −−help −−parallel)

WARNING: please use the same set of computing options at each step!!

The workflow consists in 4 steps, illustrated in the following figure and described below. (replace −−binary by −−continuous or −−survival depending on your trait)



STEP 1: compute partial top vectors on N CPUs (1, 2, ..., N)

```
./mbmdr-4.4.1.out --continuous --gammastep1 -i INT -N INT [options] 'mbmdrFile'
```

SPECIFIC OPTIONS

$-i$ *INT*	sets the current CPU id
$-N$ *INT*	sets the total number of CPUs
[$-ti$ *STRING*]	sets the prefix of the temporary top files (default: top)

STEP 2: create the final top vector on one CPU

```
./mbmdr-4.4.1.out --continuous --gammastep2 -N INT 'mbmdrFile'
```

SPECIFIC OPTIONS

$-N$ *INT*	sets the total number of CPUs
[$-t$ *STRING*]	sets the top file name (default: topFile.txt)
[$-ti$ *STRING*]	sets the prefix of the temporary top files (default: top)

STEP 3: compute the permutations on N CPUs (1, 2, ..., N)

```
./mbmdr-4.4.1.out --continuous --gammastep3 -p INT -o STRING [options] 'mbmdrFile'
```

SPECIFIC OPTIONS

$-p$ *INT*	sets the permutation amount to be run on the current CPU
$-o$ *STRING*	sets the output file name (all CPUs must use 'xxxi.txt' where xxx is a common prefix and i the CPU id)
[$-t$ *STRING*]	sets the top file name (default: topFile.txt)

STEP 4: create the final output file on one CPU

```
./mbmdr-4.4.1.out --continuous --gammastep4 -c STRING -q INT [options] 'mbmdrFile'
```

SPECIFIC OPTIONS

$-c$ *STRING*	sets the common prefix 'xxx' of the files generated at step 3
$-q$ *INT*	sets the quantity of files generated at step 3
[$-p$ *INT*]	sets the permutation amount (default: 999)
[$-o$ *STRING*]	sets the output file name (default: 'inputprefix'_output.txt the file will be created in the directory of the input file)
[$-t$ *STRING*]	sets the top file name (default: topFile.txt)

History of changes in public versions of MB-MDR

1) mbmdr-4.4.1.out: contains on-the-fly correction of categorical covariates and an option to perform a two-stage analysis with a discovery and a replication file.

2) mbmdr-4.3.3.out: solving a bug when correcting for covariates. Adding the possibility to correct for main effects when the dataset is a survival one (based on the Cox proportional hazards model).

3) mbmdr-4.3.2.out: adding the possibility to handle multiple traits (one by one). Adding the possibility to take account of covariates (residuals-based correction). Creating a new file containing the models.

4) mbmdr-4.3.1.out: same version as mbmdr-4.3.2.out, except that there is a bug when handling covariates expressed on a continuous scale, in the case of a trait expressed on a binary one. This version is not available anymore.

5) mbmdr-4.2.2.out: replacing the speedMAXT algorithm by the much faster gammaMAXT one. As a bonus, gammaMAXT does control the FWER (whereas speedMAXT did not guarantee it). Revising the default options.

6) mbmdr-4.1.0.out: solving a bug when combining speedMAXT with the 3D option. Solving a bug when using -v LONG. Allowing to combine -f and 3D.

7) mbmdr-4.0.3.out: allowing to combine the speedMAXT algorithm with the 3D option. Solving a bug with the -v option. Allowing MDR input format.

8) mbmdr-4.0.1.out: contains the new STRAT1, STRAT2 and speedMAXT algorithms. The purpose of the first two is to correct for population stratification. The purpose of the speedMAXT is to perform a MAXT-like multiple-testing, but faster (based on a maximum of 1 million non-zero test-statistics per permutation).

9) mbmdr-3.1.0.out: first version with a progress bar. Solves a bug with the 3D option.

10) mbmdr-3.0.3.out: adding code to handle a censored trait and solving a small bug related to the cholesky decomposition.

11) mbmdr-3.0.2.out: refactoring the code and adding an option for performing 3D interactions analysis. This version is about 1.5 times faster than mbmdr-2.7.5.

12) mbmdr-2.7.5.out: first public C++ version of MB-MDR containing Van Lishout's implementation of maxT. This version handles traits expressed on either a binary or a continuous scale. It can perform a main effect analysis or a two-order interaction analysis.

13) FAM-MDR: R-implementation of the MB-MDR methodology. Very slow and not flexible. Can only handle a trait expressed on a binary scale. Note that this software prepares the data by computing residuals that have removed familial relationships as much as possible, before running the MB-MDR methodology. Removing the familial relationships is done using the polygenic function in the GenABEL package. The C++ software does not remove the familial relationships, allowing you to adopt another modeling strategy.

Bibliography

[1] Alanis-Lobato, G., Cannistraci, C. V., Eriksson, A., Monica, A., and Ravasi, T. Highlighting nonlinear patterns in population genetics datasets. *Scientific Reports 5*, 8140 (2015).

[2] Allenby, G. M., Leone, R. P., and Jen, L. A dynamic model of purchase timing with application to direct marketing. *Journal of the American Statistical Association 94* (1999), 365–74.

[3] Aschard, H., Lutz, S., Maus, B., Duell, E., Fingerlin, T., Chatterjee, N., Kraft, P., and Van Steen, K. Challenges and opportunities in genome-wide environmental interaction (GWEI) studies. *Hum Genet 131*, 10 (2012), 1591–613.

[4] Balding, D. J. A tutorial on statistical methods for population association studies. *Nat Rev Genet 7* (2006), 781–791.

[5] Barett, J. C., Hansoul, S., Nicolae, D. L., Cho, J. H., Duerr, R. H., Rioux, J. D., Brant, S. R., Silverberg, M. S., Taylor, K. D., Barmada, M. M., et al. Genome-wide association defines more than 30 distinct susceptibility loci for crohn's disease. *Nat Genet 40*, 8 (2008), 955–962.

[6] Beevers, C. G., and E, M. J. Therapygenetics: moving towards personalized psychotherapy treatment. *Trends in Cognitive Sciences 16*, 1 (2012), 11–12.

[7] Bickel, P. J., and Doksum, K. A. *Mathematical Statistics, Basic Ideas and Selected Topics.* Prentice-Hall, Inc., 1977.

[8] Bradley, J. Robustness? *British Journal of Mathematical and Statistical Psychology 31* (1978), 144–152.

[9] Breslow, N. E. Analysis of survival data under the proportional hazards model. *International Statistical Review 43*, 1 (1975), 45–57.

[10] Bush, W. L., Dudek, S. M., and Ritchie, M. D. Biofilter: A knowledge-integration system for the multi-locus analysis of genome-wide association studies. In *Pacific Symposium on Biocomputing* (2009), pp. 368–379.

[11] Calle, M. L., Urrea, V., Malats, N., and Van Steen, K. MB-MDR: model-based multifactor dimensionality reduction for detecting interactions in high-dimensional genomic data. Tech. Rep. 24, Department of Systems Biology, Universitat de Vic, Vic, Spain, 2008.

[12] Calle, M. L., Urrea, V., Malats, N., and Van Steen, K. mbmdr: an R package for exploring gene-gene interactions associated with binary or quantitative traits. *Bioinformatics 26*, 17 (2010), 2198–2199.

[13] Calle, M. L., Urrea, V., Vellalta, G., Malats, N., and Van Steen, K. Improving strategies for detecting genetic patterns of disease susceptibility in association studies. *Statistics in Medicine 27* (2008), 6532–6546.

[14] Cattaert, T. Main effects adjustments in MB-MDR. Tech. rep., University of Liège, 2011.

[15] Cattaert, T., Calle, M. L., Dudek, S. M., Mahachie John, J. M., Van Lishout, F., Urrea, V., Ritchie, M. D., and Van Steen, K. Model-based multifactor dimensionality reduction for detecting epistasis in case-control data in the presence of noise. *Ann Hum Genet 75* (2011), 78–89.

[16] Cattaert, T., Rial Garcia, J. A., Gusareva, E., and Van Steen, K. Comparison of different methods for detecting gene-gene interactions in case-control data. *20th Annual IGES Conference* (2011).

[17] Cerrito, P., and Cerrito, J. *Clinical Data Mining for Physician Decision Making and Investigating Health Outcomes*. Hershey, New York : Medical Information Science Reference, 2010.

[18] Chanda, P., Sucheston, L., zhang, A., Brazeau, D., Freudenheim, J. L., Ambrosone, C., and Ramanathan, M. Ambience: A novel approach and efficient algorithm for identifying informative genetic and environmental associations with complex phenotypes. *Genetics 180*, 2 (2008), 1191–210.

[19] Choi, S. C., and Wette, R. Maximum likelihood estimation of the parameters of the gamma distribution and their bias. *Technometrics 11*, 4 (1969), 683–690.

[20] Ciccacci, C., Perricone, C., Ceccarelli, F., Rufini, S., Di Fusco, D., Allessandri, C., Spinelli, F. R., Cipriano, E., Novelli, G., Valesini, G., et al. A multilocus genetic study in a cohort of italian sle patients confirms the association with stat4 gene and describes a new association with hcp5 gene. *PLoS ONE 9*, 11 (2014).

[21] Collins, R. L., Hu, T., Wejse, C., Sirugo, G., Williams, S. M., and Moore, J. H. Multifactor dimensionality reduction reveals a three-locus epistatic interaction associated with susceptibility to pulmonary tuberculosis. *BioData Mining 6*, 4 (2013).

[22] Cover, T. M., and Thomas, J. A. *Elements of information theory.* Wiley, 1991.

[23] Cox, D. R. Regression models and life-tables. *Journal of the Royal Statistical Society, Series B 34*, 2 (1972), 187–220.

[24] Dalal, S. R., and Kwon, H. K. The role of microRNA in inflammatory bowel disease. *Gastroenterology and Hepatology 6* (2010), 714–722.

[25] Devlin, B., and Roeder, K. Genomic control for association studies. *Biometrics 55*, 4 (1999).

[26] Dunn, O. J. Multiple comparisons among means. *Journal of the American Statistical Association 56*, 293 (1961), 52–64.

[27] Durstenfeld, R. Algorithm 235: Random permutation. In *Communications of the ACM 7* (1964), vol. 7, p. 420.

[28] Efron, B. The efficiency of cox's likelihood function for censored data. *Journal of the American Statistical Association 72*, 359 (1974), 557–565.

[29] Eichler, E. E., Flint, J., Gibson, G., Kong, A., Lean, S., Moore, J. H., and Nadeau, J. H. Missing heritability and strategies for finding the underlying causes of complex disease. *Nature Reviews Genetics 11*, 6 (2010), 446–450.

[30] Feldman, D., and Shavitt, Y. An optimal median calculation algorithm for estimating internet link delays from active an optimal median calculation algorithm for estimating internet link delays from active measurements. In *E2EMON* (2007), IEEE.

[31] Fisher, R. A., and Yates, F. *Statistical tables for biological, agricultural and medical research.* Oliver & Boyd, 1938.

[32] Fraley, C., and Raftery, A. E. Model-based clustering, discriminant analysis and density estimation. *Journal of the American Statistical Association 97* (2002), 611–31.

[33] Fraley, C., Raftery, A. E., Murphy, B., and Scrucca, L. mclust version 4 for r: Normal mixture modeling for model-based clustering, classification, and density estimation. Tech. rep., University of Washington, 2012.

[34] Franke, A., McGovern, D. P., Barrett, J. C., Wang, K., Radford-Smith, G., Ahmad, T., Lees, C. W., Balschun, T., Lee, J., Roberts, R., et al. Genome-wide meta-analysis increases to 71 the number of confirmed crohn's disease susceptibility loci. *Nat Genet 42*, 12 (2010), 1118–1126.

[35] Freedman, D. A. *Statistical Models: Theory and Practice.* Cambridge University Press, 2009.

[36] Galambos, J. Bonferroni inequalities. *Annals of Probability 5*, 4 (1977), 577–581.

[37] Galas, D. J., and Hood, L. Systems biology and emerging technologies will catalyze the transition from reactive medicine to predictive, personalized, preventive and participatory (P4) medicine. *Interdisciplinary Bio Central 1* (2009), 1–4.

[38] Gauderman, W. J., Thomas, D. C., Murcray, C. E., Conti, D., Li, D., and Lewinger, J. P. Efficient genome-wide association testing of gene-environment interaction in case-parent trios. *American Journal of Epidemiology 172* (2010), 116–122.

[39] Ge, Y., Dudoit, S., and Speed, T. P. Resampling-based multiple testing for microarray data analysis. Tech. Rep. 633, Department of Statistics, University of California, Berkley, 2003.

[40] Gentleman, R., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., and Ge, Y. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology 5*, R80 (2004).

[41] Gola, D., Mahachie John, J. M., Van Steen, K., and König, I. A roadmap to multifactor dimensionality reduction methods. *Briefings in Bioinformatics* (2015).

[42] Grady, B. J., Samuels, D. C., Robbins, G. K., Selph, D., Canter, J. A., Pollard, R. B., Haas, D. W., Shafer, R., Kalams, S. A., Murdock, D. G., et al. Mitochondrial genomics and cd4 t-cell count recovery after antiretroviral therapy initiation in aids clinical trials group study 384. *J acquir Immune Defic Syndr 58*, 4 (2011), 363–70.

[43] Gusareva, E., and Van Steen, K. Practical aspects of genome-wide association interaction analysis. *Hum Genet* (2014).

[44] Gusareva, E., Wei, Z., Traherne, J. A., Hugot, J. P., Cleynen, I., Cho, J. H., Hakonarson, H., and Van Steen, K. Epistasis associated to inflammatory bowel disease (IBD) in humans. In *International Genetic Epidemiology Societz (IGES) meeting* (2015).

[45] Gyenesei, A., Moody, J., Semple, C., Haley, C., and wei, w. High-throughput analysis of epistasis in genome-wide association studies with biforce. *Bioinformatics 19* (2012), 376–382.

[46] Hahn, L. W., Ritchie, M. D., and Moore, J. H. Multifactor dimensionality reduction software for detecting gene–gene and gene–environment interactions. *Bioinformatics 19*, 3 (2002), 376–382.

[47] Hardy, J., and Singleton, A. Genome-wide association studies and human disease. *New England Journal of Medicine 360* (2009), 1759–1768.

[48] Harms, M. J., and Thornton, J. W. Evolutionary biochemistry: revealing the historical and physical causes of protein properties. *Nat Rev Genet 14*, 8 (2013), 559–71.

[49] Hautsch, N., Malec, P., and Schienle, M. Capturing the zero: A new class of zero- augmented distributions and multiplicative error processes. *Journal of financial econometrics* (2013).

[50] Hemani, G., Theocharidis, A., Wei, W., and Haley, C. epiGPU: exhaustive pairwise epistasis scans parallelized on consumer level graphics cards. *Bioinformatics 27* (2011), 1462–1465.

[51] Hoggart, C. J., Parra, E. J., Shriver, M. A., Bonilla, C., Kittes, R. A., Clayton, D. G., and McKeigue, P. M. Control of confounding of genetic associations in stratified populations. *Am J Hum Genet 72*, 6 (2003), 1492–1504.

[52] Horvarth, S., and Laird, N. M. A discordant-sibship test for disequilibrium and linkage: No need for parental data. *Am J Hum Genet 63*, 6 (1998), 1886–1897.

[53] Horvath, S., Xu, X., and Laird, N. M. The family based association test method: strategies for studying general genotype-phenotype associations. *European Journal of Human Genetics 9*, 4 (2001), 301–6.

[54] Institute for digital research and education. How are the likelihood ratio, wald, and lagrange multiplier (score) tests different and/or similar? *http://www.ats.ucla.edu/stat/mult_pkg/faq/general/nested_tests.htm* (Last checked: 30 may 2016).

[55] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature 431*, 7011 (2014), 931–945.

[56] Ishwaran, H., and Kogalur, U. B. Random survival forests for r. *R News 7*, 2 (2007), 25–31.

[57] Johnston, J., and DiNardo, J. *Econometric methods*, 4th ed. Cambridge University Press, 1997.

[58] Jung, E. S., Cheon, J. H., Lee, J. H., Park, S. J., Jang, H. W., Chung, S. H., Park, M. H., Kim, T. G., Oh, H. B., Yang, S. K., et al. HLA-C*01 is a risk factor for crohn's disease. *Inflammatory Bowel Disease 22*, 4 (2016), 796–806.

[59] Kalbfleisch, J. D., and Prentice, R. L. *The statistical analysis of failure time data.* Wiley, 1980.

[60] Kam-Thong, T., Azencott, C., Cayton, L., Putz, B., Altmann, A., Karbalai, N., Samann, P., Scholkopf, B., Muller-Myhsok, B., and Borgwardt, K. GLIDe: GPU-based linear regression for detection of epistasis. *Hum Hered 73* (2012), 220–236.

[61] Kam-Thong, T., Czamara, D., Tsuda, K., Borgwardt, K., Lewis, C., Erhardt-Lehmann, A., Hemmer, B., Rieckmann, P., Daake, M., Weber, F., et al. ePIBLaSTeR-fast exhaustive two-locus epistasis detection strategy using graphical pro- cessing units. *European Journal of Human Genetics 19* (2011), 465–471.

[62] Kaser, A., Zeissig, S., and Blumberg, R. S. Inflammatory bowel disease. *Annu. Rev. Immunol. 28* (2010), 573–621.

[63] Kleinbaum, D. *Survival Analysis, a self learning text.* Springer-Verlag, 1996.

[64] Knights, J., Yang, J., Zhang, A., and Ramanathan, M. Symphony, an information-theoretic method for gene–gene and gene–environment interaction analysis of disease syndromes. *Heredity 110*, 6 (2013), 548–59.

[65] Knuth, D. *The Art of Computer Programming, Volume 3: Sorting and Searching, Second Edition.* Addison-Wesley, 1998.

[66] König, I., Loley, C., and Ziegler, A. How to include chromosome x in your genome-wide association study. *Genetic Epidemiology 38*, 2 (2014), 97–103.

[67] Kotz, S., Balakrishnan, N., and Johnson, N. L. *Continuous Multivariate Distributions, Models and Applications.* Wiley, 2000.

[68] Laing, B., Han, D. Y., and Ferguson, L. R. Candidate genes involved in beneficial or adverse responses to commonly eaten brassica vegetables in a new zealand crohn's disease cohort. *Nutrients 5*, 12 (2013), 5046–64.

[69] Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., Devon, K., Dewar, K., Doyle, M., FitzHugh, W., et al. Initial sequencing and analysis of the human genome. *Nature 409*, 6822 (2001), 860–921.

[70] Lee, S. H., Wray, N. R., Goddard, M. E., and Visscher, P. M. Estimating missing heritability for disease from genome-wide association studies. *The American Journal of Human Genetics 88*, 3 (2011), 294.

[71] Lester, K. J., and Eley, T. C. Therapygenetics: Using genetic markers to predict response to psychological treatment for mood and anxiety disorders. *Biology of mood and anxiety disorders 3*, 1 (2013), 1–16.

[72] Levy, S., Sutton, G., Ng, P., Feuk, L., Halpern, A., Walenz, B., Axelrod, N., Huang, J., Kirkness, E., Denisov, G., and et al. The diploid genome sequence of an individual human. *PLoS Biol 5*, 10 (2007).

[73] Libioulle, C., Louis, E., Hansoul, S., Sandor, C., Farnir, F., Franchimont, D., Vermeire, S., Dewit, O., de Vos, M., Dixon, A., et al. Novel crohn disease locus identified by genome-wide association maps to a gene desert on 5p13.1 and modulates expression of PTGER4. *Plos Genetics 3*, 4 (2007), e58.

[74] Lin, X., Cai, T., Wu, M. C., Zhou, Q., Liu, G., Christiani, D., and Lin, X. Kernel machine snp-set analysis for censored survival outcomes in genome-wide association studies. *Genet Epidemiol. 35*, 7 (2011), 620–631.

[75] Lipsitz, S., Fitzmaurice, G. M., Sinha, D., Hevelone, N., Giovannucci, E., and Hu, J. C. Testing for independence in $j \times k$ contingency tables with complex sample survey data. *Biometrics 71*, 3 (2015), 832–40.

[76] Liu, L., Zhang, D., Liu, H., and Arendt, C. Robust methods for population stratification in genome wide association studies. *BMC Bioinformatics 14*, 132 (2013).

[77] Londin, E. R., Keller, M. A., Maista, C., Smith, G., Mamounas, L. A., Zhang, R., Madore, S. J., Gwinn, K., and Corriveau, R. A. CoAIMs: A cost-effective panel of ancestry informative markers for determining continental origins. *PLoS ONE 5*, 10 (2010).

[78] Mackay, T. F. C. Epistasis and quantitative traits: Using model organisms to study gene-gene interactions. *Nat Rev Genet 15*, 1 (2014), 22–23.

[79] Mahachie John, J. M. *Genomic Association Screening Methodology for High-Dimensional and Complex Data Structures: Detecting n-Order Interactions*. PhD thesis, University of Liège, 2012.

[80] Mahachie John, J. M., Cattaert, T., Van Lishout, F., Gusareva, E., and Van Steen, K. Lower-order effects adjustment in quantitative traits model-based multifactor dimensionality reduction. *PLoS ONE 7(1)* (2012), e29594. doi:10.1371/journal.pone.0029594.

[81] Mahachie John, J. M., Gusareva, E., Van Lishout, F., and Van Steen, K. A robustness study to investigate the performance of parametric and non-parametric tests used in model-based multifactor dimensionality reduction epistasis detection. *BioData Mining* (2013).

[82] Mahachie John, J. M., Van Lishout, F., and Van Steen, K. Model-based multifactor dimensionality reduction to detect epistasis for quantitative traits in the presence of error-free and noisy data. *European Journal of Human Genetics 19*, 6 (2011), 696–703.

[83] Mahdi, B. M. Role of HLA typing on crohn's disease pathogenesis. *Ann Med Surg 4*, 3 (2015).

[84] Manolio, T. A., Collins, F. S., Goldstein, D. B., Hindorff, L. A., Hunter, D. J., McCarthy, M. I., Ramos, E. M., Cardon, L. R., Chakravarti, A., Cho, J. H., et al. Finding the missing heritability of complex diseases. *Nature 461*, 7265 (2009), 747–753.

[85] Mazzocchi, G., Andreis, P. G., De Caro, R., Aragona, F anf Gottardo, L., and Nussdorfer, G. Cerebellin enhances in vitro secretory activity of human adrenal gland. *J Clin Endocrinol Metab 84*, 2 (1999), 632–5.

[86] Minka, T. P. Estimating a Gamma distribution. . *http://research.microsoft.com/en-us/um/people/minka/papers/minka-gamma.pdf* (Last checked: 30 may 2016).

[87] Moore, J. H., and Williams, S. M. Traversing the conceptual divide between biological and statistical epistasis: systems biology and a more modern synthesis. *BioEssays 27*, 6 (2005), 637–646.

[88] Moreau, Y., and Tranchevent, L. Computational tools for prioritizing candidate genes: boosting disease gene discovery. *Nature Review Genetics 13*, 8 (July 2012), 523–36.

[89] National Center for Biotechnology Information. NCBI dbSNP build 138 for human. *United States National Library of Medicine* (2013).

[90] Niu, A., Zhang, S., and Sha, Q. A novel method to detect gene–gene interactionsin structured populations: MDR-SP. *Ann Hum Genet 75* (2011), 742–54.

[91] Park, M. Y., and Hastie, T. Penalized logistic regression for detecting gene interactions. *Biostatistics 9*, 1 (2007), 30–50.

[92] Pattin, K. A., White, B. C., Barney, N., Gui, J., Nelson, H. H., Kelsey, K. T., Andrew, A. S., Karagas, M. R., and Morre, J. H. A computationally efficient hypothesis testing method for epistasis analysis using multifactor dimensionality reduction. *Genet Epidemiol. 33*, 1 (2009), 87–94.

[93] Pavlopoulos, G., Oulas, A., Iacucci, E., Sifrim, A., Moreau, Y., Schneider, R., Aerts, J., and Iliopoulos, I. Unraveling genomic variation from next generation sequencing data. *BioData Mining 6*, 13 (July 2013), 1–25.

[94] Peto, R., and Peto, J. Asymptotically efficient rank invariant test procedures. *Journal of the Royal Statistical Society, Series A (Blackwell Publishing) 135*, 2 (1972), 185–207.

[95] Price, A. L., Patterson, N. J., Plenge, R. M., Weinblatt, M. E., Shadick, N. A., and Reich, D. Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics 38*, 8 (2006), 904–9.

[96] Raelson, J. V., Little, R. D., Ruether, A., Fournier, H., Paquin, B., Van Eerdewegh, P., Bradley, W. E. C., Croteau, P., Nguyen-Huu, Q., Segal, J., et al. Genome-wide association study for crohn's disease in the quebec founder population identifies multiple validated disease loci. *Proceedings of the National Academy of Sciences 104*, 37 (2007), 14747–52.

[97] RARE Project. Rare disease impact report: Insights from patients and the medical community. *Survey from the global Genes Report financed by Shire* (2013).

[98] Raychaudhuri, S. VIZ-GRAIL: visualzing functional connections across disease loci. *Bioinformatics 27* (2011), 1589–1590.

[99] Raychaudhuri, S., Plenge, R. M., Rossin, E., Ng, A. C., Consortium, I. S., Purcell, S. M., Sklar, P., Scolnick, E. M., Xavier, R. J., Altshuler, D., et al. Identifying relationships among genomic disease regions: Predicting genes at pathogenic snp associations and rare deletions. *PLoS Genetics 5*, 9 (2009), 1–15.

[100] Reveille, J. D. Genetics of spondyloarthritis—beyond the mhc. *Nature Reviews Rheumatology 8* (2012), 296–304.

[101] Ritchie, M. D., Hahn, L. W., and Moore, J. H. Power of multifactor dimensionality reduction for detecting gene-gene interactions in the presence of genotyping error, missing data, phenocopy, and genetic heterogeneity. *Genet Epidemil. 24*, 2 (2003), 150–7.

[102] Ritchie, M. D., Hahn, L. W., Roodi, N., Bailey, L. R., Dupond, W. D., Parl, F. F., and Moore, J. H. Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. *Am J Hum Genet 69* (2001), 138–147.

[103] Romano, J. P., Shaikh, A. M., and Wolf, M. Formalized data snooping based on generalized error rates. *Econometric Theory 24* (2008), 404–447.

[104] Sarin, R., Wu, X., and Abraham, C. Inflammatory disease protective R381Q IL23 receptor polymorphism results in decreased primary CD4+ and CD8+ human T-cell functional responses. *Proc Natl Acad Sci USA* (2011).

[105] Scott, L. J. *Regression Models for Categorical and Limited Dependent Variables.* Thousand Oaks CA: Sage Publications, 1997.

[106] Shastry, B. S. Pharmacogenetics and the concept of individualized medicine. *Pharmacogenomics J. 6*, 1 (2006), 16–21.

[107] Simpson, E. H. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society, Series B 13*, 238-241 (1951).

[108] Slager, S. L., and Schaid, D. J. Case-control studies of genetic markers: power and sample size approximations for armitage's test for trend. *Human Heredity 52* (2001), 149–153.

[109] Slatkin, M. Epigenetic inheritance and the missing heritability problem. *Genetics 182*, 3 (2009), 845–850.

[110] Smyth, G. An efficient algorithm for REML in heteroscedastic regression. *Journal of computational and Graphical Statistics 11*, 4 (2002), 836–847.

[111] Spielman, R. S., McGinnis, R. E., and Ewens, W. J. Transmission test for linkage disequilibrium: the insulin gene region and insulin-dependent diabetes mellitus (IDDM). *Am J Hum Genet 52*, 3 (1993), 506–16.

[112] Tantisira, K. G., Damask, A., Szefler, S. J., Schuemann, B., Markezich, A., Su, J., Klanderman, B., Sylvia, J., Wu, J., Martinez, F., et al. Genome-wide association identifies the t gene as a novel asthma pharmacogenetic locus. *Am J Respir Crit Care Med 185*, 12 (2012), 1286–91.

[113] Tao, S., Wand, Z., Feng, J., Hsu, F. C., Jin, G., Kim, S. T., Zhang, Z., Gronberg, H., zheng, L. S., Isaacs, W. B., et al. A genome-wide search for loci interacting with known prostate cancer risk-associated genetic variants. *Carcinogenesis 33*, 3 (2012), 598–603.

[114] Taylor, K. D., Targn, S. R., Mei, L., Ippoliti, A. F., McGovern, D., Mengesha, E., King, L., and Rotter, J. I. IL23R haplotypes provide a large population attributable risk for crohn's disease. *Inflammatory Bowel Disease 14*, 9 (2008), 1185–1191.

[115] Tervaniemi, M. H., Siitonen, H. A., Söderhäll, C., Minhas, G., Vuola, J., Tiala, I., Sormunen, R., Samuelsson, L., Suomela, S., Kere, J., et al. Centrosomal localization of the psoriasis candidate gene product, cchcr1, supports a role in cytoskeletal organization. *PLoS ONE 7*, 11 (2012).

[116] Therneau, T. M., Grambsch, P. M., and Fleming, T. R. Martingale-based residuals for survival models. In *Biometrika* (1990), vol. 77, pp. 147–60.

[117] Tiala, I., Wakkinen, J., Suomela, S., Puolakkainen, P., Tammi, R., forsberg, S., Rollman, O., Kainu, K., Rozell, B., Kere, J., et al. The psors1 locus gene cchcr1 affects keratinocyte proliferation in transgenic mice. *Hum. Mol. Genet. 17*, 7 (2008), 1043–51.

[118] Ting, H., Sinnott-Armstrong, N. A., Kiralis, J. W., Andrew, A. S., Karagas, M. R., and H, M. J. Characterizing genetic interactions in human disease association studies using statistical epistasis networks. *BMC Bioinformatics 12*, 364 (2011).

[119] Tucker, G., Price, A. L., and Berger, B. Improving the power of gwas and avoiding confounding from population stratification with pc-select. *Genetics 187*, 3 (2014), 1045–9.

[120] Urbanowicz, R. J., Kiralis, J., Sinnott-Armstrong, N. A., T, H., M, F. J., and H, M. J. GAMETES: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures. *BioData Mining 5*, 16 (2012).

[121] Van Lishout, F., Bessonov, K., Duan, K., Gusareva, E., Mahachie John, J. M., Tantishira, K., and Van Steen, K. Genome-wide environmental interaction analysis using multidimensional data reduction principles to identify asthma pharmacogenetic loci in relation to corticosteroid therapy. In *Annual Society of Human Genetics annual Meeting, Boston, USA* (2013).

[122] Van Lishout, F., Cattaert, T., Mahachie John, J. M., Gusareva, E., Urrea, V., Cleynen, I., Theatre, E., Charloteaux, B., Kvasz, A., Calle, M. L., Wehenkel, L., and Van Steen, K. An efficient algorithm to perform multiple testing in epistasis screening. In *Benelux Bioinformatics Conference, Luxembourg* (2011).

[123] Van Lishout, F., Gadaleta, F., Moore, J. H., Wehenkel, L., and Van Steen, K. gammaMAXT: a fast multiple-testing correction algorithm. In *ERCIM 2014 Abstract Book* (2014), Pisa, Ed.

[124] Van Lishout, F., Gadaleta, F., Moore, J. H., Wehenkel, L., and Van Steen, K. gammaMAXT: a fast multiple-testing correction algorithm. *BioData Mining* (2015).

[125] Van Lishout, F., Mahachie John, J. M., Gusareva, E. S., Urrea, V., Cleynen, I., Théâtre, E., Charloteaux, B., Calle, M. L., Wehenkel, L., and Van Steen, K. An efficient algorithm to perform multiple testing in epistasis screening. *BMC Bioinformatics 14*, 138 (2013).

[126] Van Lishout, F., Vens, C., Urrea, V., Calle, M. L., Wehenkel, L., and Van Steen, K. Survival analysis: finding relevant epistatic SNP pairs using model-based multifactor dimensionality reduction. In *5th International Conference of the ERCIM WG on computing & statistics, Oviedo, Spain* (2012).

[127] Van Steen, K. Traveling the world of gene-gene interactions. *Briefings in Bioinformatics 13*, 1 (2011), 1–19.

[128] van't Veer, L. J., and Bernards, R. Enabling personalized cancer medicine through analysis of gene-expression patterns. *Nature 452*, 7187 (2008), 564–70.

[129] Varadan, V., Miller, D. M., and Anastassiou, D. Computational inference of the molecular logic for synaptic connectivity in c. elegans. *Bioinformatics 22*, 14 (2006).

[130] Visel, A., Minovitsky, S., Dubchak, I., and Pennacchio, L. A. VISTA enhancer browser—a database of tissue-specific human enhancers. *Nucleic Acids Research 35* (2007).

[131] Wan, X., Yang, C., Yang, Q., Xue, H., Fan, X., et al. BOOST: A fast approach to detecting gene-gene interactions in genome-wide case-control studies. *Am J Hum Genet 87* (2010), 325–340.

[132] Wang, K. Testing for genetic association in the presence of population stratification in genome-wide association studies. *Genet Epidemil. 33*, 7 (2009), 637–45.

[133] Watkinson, J., and Anastassiou, D. Synergy disequilibrium plots: graphical visualization of pairwise synergies and redundancies of snps with respect to a phenotype. *Bioinformatics 25*, 11 (2009), 1445–1446.

[134] Weibull, W. A statistical distribution funciton of wide applicability. *J. Appl. Mech.-Trans. ASME 18*, 3 (1951), 293–97.

[135] Westfall, P. H., and Troendle, J. F. Multiple testing with minimal assumptions. *Biom J. 50*, 5 (2008), 745–755.

[136] Westfall, P. H., and Young, S. S. *Resampling-base multiple testing.* Wiley, New York, 1993.

[137] Wienbrandt, L., Kassens, J., Gonzalez-Dominguez, J., Schmidt, B., D, E., and Schimmler, M. FPGA-based acceleration of detecting statistical epistasis in GWAS. In *14th International Conference on Computational Science* (2014), P. C. Science, Ed., vol. 29, pp. 220–230.

[138] Winham, S. J., Colby, L. C., Freimuth, R. R., Wang, X., de Andrade, M., Huebner, M., and Biernacka, J. M. Snp interaction detection with random forests in high-dimensional genetic data. *BMC Bioinformatics 13*, 164 (2012).

[139] Won, S., Lu, Q., Bertram, L., Tanzi, R. E., and Lange, C. On the meta-analysis of genome-wide association studies: A robust and efficient approach to combine population and family-based studies. *Hum Hered 73*, 1 (2012), 35–46.

[140] Xin, Y., and Xia, G. S. Linear regression analysis: Theory and computing. In *World Scientific* (2009).

[141] Yang, J., Weedon, M. N., Purcell, S., Lettre, G., and Estrada, K. e. a. Genomic inflation factors under polygenic inheritance. *European Journal of Human Genetics 19*, 7 (2011), 807–12.

[142] Yang, S. K., Lee, S. G., Cho, Y. K., Lim, J., Lee, I., and Song, K. Association of tnf-alpha/lta polymorphisms with crohn's disease in koreans. *Cytokine 35*, 1-2 (2006), 13–20.

[143] Zhou, X., Richon, V. M., Wang, A. H., Yang, X. J., Rifkind, R. A., and Marks, P. A. Histone deacetylase 4 associates with extracellular signal-regulated kinases 1 and 2, and its cellular localization is regulated by oncogenic ras. *Proc Natl Acad Sci USA 97* (2000), 14329–14333.

[144] Zöllner, S., and Pritchard, J. K. Overcoming the winner's curse: Estimating penetrance parameters from case-control data. *Am J Hum Genet 80*, 4 (2007), 605–15.

[145] Zuk, O., Hechter, E., Sunyaev, S. R., and Lander, E. S. The mystery of missing heritability: Genetic interactions create phantom heritability. *Proceedings of the National Academy of Sciences 109*, 4 (2012), 1193–1198.