RESEARCH

Getting rid of permutation-based multiple-testing correction

François Van Lishout[??,??*], Francesco Gadaleta[??,??], Jason H Moore[??], Louis Wehenkel[??,??] and Kristel Van Steen[??,??]

[*]Correspondence:
F.VanLishout@ulg.ac.be
[??]Systems and Modeling Unit, Montefiore Institute, University of Liège, 4000 Liège, Belgium
Full list of author information is available at the end of the article

Abstract

Background: The purpose of the maxT algorithm, introduced by Westfall & Young in 1993, is to control the family-wise error rate (FWER), without being as conservative as a Bonferroni correction. However, this procedure basically multiplies the computing time and memory usage by the amount of permutations. In 2013, the memory issue was solved by Van Lishout's implementation of maxT, which makes the memory usage independent from the size of the problem. This algorithm is implemented in *MBMDR-3.0.3*, a software able to identify interaction effects for a variety of epistasis models in a powerful way. It is based on the MB-MDR methodology, a non-parametric data mining method able to distinguish between multiple pure interaction effects and interaction effects induced by important main effects. However, *MBMDR-3.0.3* is not able to perform a genome-wide interaction analysis study, because of the computing time issue. We present gammaMAXT, a new algorithm implemented in *MBMDR-4.2.0*, a software tool solving this issue.

Results: We show that the test-statistics produced by the MB-MDR methodology follow a mixture distribution with a point mass at zero and for positive values a shifted gamma distribution. The parameters of this distribution depend on the particular dataset at hand. With this in mind, we adapt Van Lishout's implementation of maxT, to avoid the need to compute test-statistics for all interactions of each permutation explicitely. We show that the gammaMAXT algorithm reaches a power similar to maxT, but requires a much lower computing time, while still controlling the FWER. In the case of a binary (affected/unaffected) trait, the parallel workflow of *MBMDR-4.2.0* analyzes all gene-gene interactions with a dataset of 1 million SNPs typed on 1000 individuals within 4 (?) days, using 999 permutations of the trait to assess statistical significance, on a cluster composed of 32 blades, containing each two quadcore Intel L5420 2.5 GHz. In the case of a continuous trait, a similar run takes 7 (?) days.

Conclusion: In this work, we could successfully replace a permutation-based multiple-testing correction strategy by a semi-analytical approach, in the context of the MB-MDR methodology. In this way, we could win orders of magnitudes of computing time.

Keywords: maxT; MB-MDR; multiple-testing; GWAIs

Introduction

Nowadays, the time to diagnosis of rare diseases is very long. According to patients surveyed, it takes on average 7.6 years in the US and 5.6 years in the UK to receive a proper diagnosis [**?**]. In the mean time, a patient typically visits four primary

care doctors, four specialits and receives 2 to 3 misdiagnoses [**?**]. A key factor to reduce the time to diagnosis duration is to better understand the genetic origins of complex diseases. Many scientists around the world are focusing on this task and big steps are achieved on a daily basis. This leads to an increasing amount of new healthcares tailored to the individual patients. The era of personalized medicine is comming [**?**, **?**, **?**, **?**, **?**]. This paper focuses on genome-wide association interaction studies (GWAIs), whose purpose is to identify pairs of genes and/or environmental factors that increase or dicrease susceptibility to disease.

Background

We present *MBMDR-4.2.0*, a new software tool implemented in C++, using the Model-Based Multifactor Dimensionality Reduction (MB-MDR) methodology [**?**, **?**, **?**, **?**]. We compare to *MBMDR-3.0.3*, a former version of this software [**?**], which greatly enhances MB-MDR's first implementation as an R-package [**?**], both in terms of flexibility and efficiency. In the case of a binary (affected/unaffected) trait, the parallel workflow of *MBMDR-3.0.3* analyzes all gene-gene interactions with a dataset of 100,000 SNPs typed on 1000 individuals within 4 (?) days, on a cluster composed of 32 blades, containing each two quadcore Intel L5420 2.5 GHz. Increasing the amount of SNPs to 1 million approximately multiplies the amount of interactions by 100 and so the computing time. Such a run would thus require about 400(?) days. *MBMDR-4.2.0* performs the same run in 4(?) days.

The heart of the MB-MDR methodology is to identify sets of gene-gene or gene-environment interactions via a series of association tests. Significance of the explored interactions is assessed using the *maxT* method [**?**] which provides adjusted p-values by controlling for the multiple correlated tests. This guarantees weak control of the family-wise error rate (FWER) under all conditions and strong control under the subset pivotality assumption [**?**]. In practice, only a few p-values will point towards interesting interactions to investigate. With this in mind, Van Lishout's implementation of *maxT* adapts the original method so that it still calculates the test-statistics for all SNP pairs, but only computes the p-values of the n best pairs, i.e. the ones with the n lowest p-values [**?**]. The default value of the software ($n = 1000$) is appropriate when epistasis is tested for in a hypotheses-free way, because it is highly unlikely that more than 1000 significant epistatic pairs will be identified. Note that this value can easily be increased if the context requires it. In this work, we show that the test-statistics produced by the MB-MDR methodology follow a mixture distribution with a point mass at zero and for positive values a shifted gamma distribution. With this in mind, for each permutation, only a sample from the possible interactions is taken. The test-statistics relative to this sample are computed and used to fitt the mixture distribution using the maximum likelihood estimation method. The maximum that would be obtained if all interactions would be explicitely computed is then estimated from this fitted distribution, winning orders of magnitude of computing time.

Method

Mixture of distribution behing the test-statistics of the MB-MDR methodology

MB-MDR is a flexible methodolgy, able to handle a trait expressed on a binary scale, as well as a continuous scale or as a censored trait. Furthermore, it can search

for interaction effects in a direct way or correct for lower order effects. This leads to very different test-statistics to assess significance of the epistatic pairs. However, we sustain that a mixture distribution with a point mass at zero and for positive values a shifted gamma distribution is a powerful family to model them. We show the goodness of fitt in eight practical scenarios.

One of the major difference between the MB-MDR methodology and its ancestror MDR [some citations comes here, ask Jason], is the introduction of the "O" category in the multi dimensionality reduction process [**?**, **?**]. In MDR, whenever two groups of subjects are compaired (for instance, those having the minor allele for two SNPs versus all other subjects) the first group must be either associated to a higher risk to develop to disease than the second one ("H" category) or a lower risk ("L" category). In MB-MDR, there is a third possibility: if the difference is not statistically significant, it can be associated to no evidence for risk change ("O" category). As a consequence, it can happen that no genotype combination shows any evidence for association with the trait. In this case MB-MDR returns an exact zero. In practice, this happens for the majority of the pairs! To account for this important amount of zeros, we use the same approcach as in [**?**]. We assign a discrete probability mass to the exact zero value. Hence, if X is a variable returning a random MB-MDR test-statistic, we can define the probabilites $\pi = P(X > 0)$ and $1 - \pi = P(X = 0)$. From this, the distribution of X is semicontinuous with a discontinuity at zero, implying the density $f_X(x) = (1 - \pi)\delta(x) + \pi g_X(x)\mathbb{1}_{(x>0)}$, where $\delta(x)$ is a point probability mass at $x = 0$ 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 dataset at hand. Note that the minimum of the non-zero values cannot be too close to zero, because it would correspond to an interaction for which no genotype combination would show any significant association with the trait, which as we have seen would have lead to the "O" category for each genotype combination and therefore an exact-zero. For this reason, the distribution of the non-zero values has to be shifted to the right.

This is where Francesco comes in play ... write a text that justifies that a shifted gamma distribution is a good choice in general and especially here. Reference [**?**] gives for an instance a detailed explanation of how a test-statistic is computed when the trait is binary and no correction is made for the main effects. Start for instance by linking to this and saying simple that the final test-statistic returned by this proecdure is a chi-square. Explain that the family of gamma distribution is powerful enough to model such a test-statistic. Reference [**?**] shows for instance how MB-MDR can handle a continuous trait and correct for lower-order effects. At the end, the test-statistic is a student t-test. Show again how easily this will be fitted using a gamma.

All 8 analysis show that $g_X(x)$ is the density function of a shifted gamma distribution, i.e. $g_X(x) = \frac{(x-\gamma)^{k-1}e^{\frac{x-\gamma}{\theta}}}{\theta^k\Gamma(k)}$, where k, θ and γ are respectively the shape, scale and location parameters.

We show now the goodness of fitt in eight practical scenarios. We run MBMDR-4.2.0 on 4 different datasets and adapt the source code to store every single test-statistics generated by the program into a file. We consider four datasets:

- A simulated dataset D_1, for which the trait is expressed on a binary scale. This dataset is composed of 100,000 SNPs and 1000 individuals (500 cases

and 500 controls). It was generated using GAMETES, a fast, direct algorithm for generating pure, strict, epistatic models with random architectures [**?**].

- A real-life dataset D_2, for which the trait is expressed on a binary scale ... still to be discussed with Kristel...
- A simulated dataset D_3, for which the trait is expressed on a continuous scale. This dataset is composed of 100,000 SNPs and 1000 individuals. ... explain the model ...
- A real-life dataset D_4, for which the trait is expressed on a continuous scale ... still to be discussed with Kristel ... (so far, I used one of Elena's big datasets)

We analyze each datasets with two typical settings of MBMDR-4.2.0:

- Setting S_1 (default parameters): the programs makes a codominant correction for the main effects of each SNP.
- Setting S_2 (option "-a NONE"): the program does not make any adjustment for the main effects of the SNPs.

Text by Francesco to show that the non-zero values of the 8 possible combinations of datasets and settings, follow a shifted gamma distribution ... I will send the 8 files ... also explain that we are in fact interested in the tail of the distribution and that this is the part of the distribution that we need to fitt as precisely as possible, not the middle or the head ... please produce a figure composed of 8 plots, comparing each time the observed data and the fitted distribution ...

The gammaMAXT algorithm

Figure 1 describes the difference between three algorithms: the original maxT, Van Lishout's maxT and the new gammaMAXT. The different steps of Van Lishout's maxT are given below:

1 Compute the test-statistics for all m pairs, but store only the n highest ones. The result is a *Real data* vector where $T_{0,1} \geq T_{0,2} \geq \ldots \geq T_{0,n}$.

2 Initialize a vector a of size n with 1's.

3 Perform the following operations for $i = 1, \ldots, B$:
   (a) Generate a random permutation of the trait column.
   (b) Compute the test-statistics $T_{i,1}, \ldots, T_{i,n}$ and store them in a $Permutation_i$ vector.
   (c) Compute the maximum M_i of the test-statistics values $T_{i,n+1}, \ldots, T_{i,m}$.
   (d) Replace $T_{i,n}$ by M_i if $T_{i,n} < M_i$.
   (e) Force the monotonicity of the $Permutation_i$ vector: for $j = n-1, \ldots, 1$ replace $T_{i,j}$ by $T_{i,j+1}$ if $T_{i,j} < T_{i,j+1}$.
   (f) For each $j = 1, \ldots, n$, if $T_{i,j} \geq T_{0,j}$ increment a_j by one.

4 Divide all values of vector a by $B+1$ to obtain the *p-values* vector p. Force monotonicity as follows: for $j = 1, \ldots, n-1$, replace p_{j+1} by p_j if $p_{j+1} < p_j$.

For big datasets, the bottelneck of this procedure is step 3 (c). Consider for instance a dataset of 1 million SNPs, i.e. m=5x10^{11} interactions, analyzed with the default settings of the software ($n = 1000$, $B = 999$). Steps involving n but not m are not really time consuming, since $n << m$. Therefore, only step 1 and 3(c) remains. The former is performed only once and will require $O(10^{11})$ test-statistic computations, whereas the later is performed 999 times and will require $O(10^{14})$.

Therefore, the gammaMAXT algorithm is exactly the same as Van Lishout's implementation of maxT, except for step 3 (c) which is replaced by the following operation:

3 (c) Estimate the maximum M_i of the test-statistics values $T_{i,n+1}, \ldots, T_{i,m}$.

- i Initialize an integer z to 0 and create a vector *sample* of size N = 100,000.
- ii Select an index r at random in $[n+1, m]$ and compute $T_{i,r}$
- iii If $T_{i,r} = 0$, increase z, otherwise, store $T_{i,r}$ in an empty cell of to the vector *sample*
- iv Repeat steps ii. and iii. until the vector *sample* is full.
- v Estimate the parameter π by $\frac{N}{z+N}$. As a consequence, the amount of non-zeros in $T_{i,n+1}, \ldots, T_{i,m}$ is estimated by $\pi \times (m-n)$
- vi Estimate the location parameter γ by the minimum of the vector *sample*
- vii Estimate the shape k and the scale θ using the maximum likelihood estimation method (see below for the exact procedure).
- viii Estimate the maximum M_i that would be obtained, if we would take a sample of size z from a random variable following the fitted shifted gamma distribution (see below for the exact procedure).

To estimate the parameters k and θ, knowing the location parameter γ, we define $s = ln(\frac{1}{N}\sum_{i=1}^{N}(x_i - \gamma)) - \frac{1}{N}\sum_{i=1}^{N} ln(x_i - \gamma)$, then $k \approx \frac{3-s+\sqrt{(s-3)^2+24s}}{12s}$ is within 1.5% of the correct value [**?**]. A Newton-Raphson update of this initial guess is then computed by $k \leftarrow k - \frac{ln(k)-\psi(k)-s}{\frac{1}{k}-\psi'(k)}$, where $\psi(k)$ and $\psi'(k)$ are respectively the digamma and trigamma functions [**?**]. Finally, the maximum likelihood estimator of θ is given by $\frac{1}{kN}\sum_{i=1}^{N}(x_i - \gamma)$. At this point, we test this procedure on the 4 datasets $D_1, ..., D_4$ of the previous section, using the settings S_1 and S_2. In all case, we observe that the fitted parameters are approximately the same for all permutations. An analogous observation was noticed in a similar work, based on hypothesis testing using an extreme value distribution [**?**]. With this in mind, we adapt the gammaMAXT algorithm such that it does not fitt new parameters for each single permutation, but only for 1 out of 20. This is a compromise between winning computing-time (an order of magnitude) and being robust (not relying on a single fitting).

The cumulative distribution function of a shifted gamma distribution is given by $F(x) = \frac{\Gamma_x(k, \frac{x-\gamma}{\theta})}{\Gamma(k)}$, where $\Gamma_x(k, \frac{x-\gamma}{\theta})$ is the lower incomplete gamma function, defined by $\int_0^{\frac{x-\gamma}{\theta}} t^{k-1}e^{-t}dt$. By definition, if we sample one value from this distribution, the probability that its value is lower than a particular threshold x_t is given by $\frac{\Gamma_x(k, \frac{x_t-\gamma}{\theta})}{\Gamma(k)}$. Therefore, if we sample z independent and identically distributed (i.i.d.) values, the probability that the maximum of the $(x_1, x_2, ..., x_z)$ sample is lower than x_t is given by $P[(x_0 \leq x_t) \wedge (x_1 \leq x_t) \wedge ... \wedge (x_z \leq x_t)] = [\frac{\Gamma_x(k, \frac{x_0-\gamma}{\theta})}{\Gamma(k)}]^z$. A first attempt to predict M_i is thus to compute its expected median m_t, defined by $m_t : [\frac{\Gamma_x(k, \frac{m_t-\gamma}{\theta})}{\Gamma(k)}]^z = 0,5$. The problem of this approach is that all permutations will lead to approximately the same value, since all shifted gamma distributions are approximately the same. This would not mimic the maxT algorithm at all! The idea of the later is to produce a sample representing the distribution of the maxima under the null, against which the maximum of the original data can be compared

to. To achieve a similar behaviour, we chose to generate a particular sample: the B-quantiles (999-quantiles with the default settings). In this way, we describe the distribution of the maxima under the null smoothly. Our final prediction of M_i is thus the i^{th} B-quantile, defined by $M_i : [\frac{\Gamma_x(k, \frac{M_i - \gamma}{\theta})}{\Gamma(k)}]^z = \frac{i}{B+1}$. Solving this equation is far from trivial. However, the gamma and the lower incomplete gamma functions are pre-implemented in C++. For this reason, we have implemented a dichotomous search for M_i:

(a) Initialize a variable y to a value that is obviously much higher than M_i (default: 1000) and a variable *step* to half of this value (default: 500).
(b) Compute $p_g = [\frac{\Gamma_x(k, \frac{y - \gamma}{\theta})}{\Gamma(k)}]^z$.
(c) If p_g is lower (higher) than $\frac{i}{B+1}$, increase (decrease) y by *step*.
(d) Divide *step* by 2.
(e) Repeat steps (b), (c) and (d) until *step* is below the desired precision (default: 0.000001).
(f) Return the final value of y, our final prediction of M_i.

For big datasets, the bottelneck of the gammaMAXT algorithm is now step 1. Indeed, consider again a dataset of 1 million SNPs analyzed with the default settings of the software. Step 1 did not change and still requires $O(10^{11})$ test-statistic computations. Step 3 (c) however, is still performed 999 times, but only 1 out of 20 permutations, i.e. 50, will lead to a fitting of the shifted gamma distribution involving 100,000 test-statistic computations. This implies $O(10^6)$ instead of $O(10^{14})$ with Van Lishout's implementation of maxT!

Parallel workflow

The parallel workflow of Van Lishout's implementation of maxT only parallelizes step 3 [**?**]. Since the bottelneck of the new algorithm is step 1, we will now parallelize this step as well. Figure 2 describes the four steps of the new parallel workflow:

1 Split the computation of the m test-statistics of step 1 between C machine. To achieve an approximately homogeneous split, compute on each machine $c = 1 \dots C$ the pairs for which the modulo of the index of the first SNP is equal to $c - 1$ and save the n highest results ones into a file *topc.txt*.
2 When all machines have terminated their computations, read the files *top*1.*txt* ... *topC.txt* on one machine and retrieve the n highest values amoung these files. Save the result into a file *topfile.txt*. This file will contain the information of the *Real Data* vector of Figure 1.
3 Split the computation of the permutations homogeneously between the C machines. On each machine $c = 1 \dots C$, perform the following operations:
   (a) Read the file *topfile.txt*
   (b) Initialize a vector p of size n with 0's.
   (c) Execute step 3 of Van Lishout's *maxT* algorithm for each permutation assigned to c (using vector p instead of a).
   (d) Save the p vector into a file *permutc.txt*.
4 When all machines have terminated their work, sum all vectors of the files *permut*1.*txt* ... *permutC.txt* to obtain a vector p. Add 1 to all elements of this vector. Perform step 4 of the *gammaMAXT* algorithm on p.

Competing interests
The authors declare that they have no competing interests.

Author's contributions
Text for this section ...

Acknowledgements
Text for this section ...

Figures

Figure 1 Classical versus Van Lishout's implementation of maxT and gammaMAXT In the classical *maxT* implementation, all $T_{i,j}$ values are computed and put in memory. In Van Lishout's implementation of *maxT*, all $T_{i,j}$ values are computed but only the maximum $M_1, \ldots, M_B$ of the $[T_{1,n+1}, \ldots, T_{1,m}], \ldots, [T_{B,n+1}, \ldots, T_{B,m}]$ are stored in memory. In gammaMAXT, only a sample from each $[T_{1,n+1}, \ldots, T_{1,m}], \ldots, [T_{B,n+1}, \ldots, T_{B,m}]$ is computed and used to predic the maximum $M_1, \ldots, M_B$.

Figure 2 MBMDR-4.2.0 parallel workflow
The computation of the test-statistics is first split between the available machines ... Finally, *MBMDR-4.2.0* reads the produced $permut?.txt$ files to create the final output file.

Tables

Table 1 Sample table title. This is where the description of the table should go.

	B1	B2	B3
A1	0.1	0.2	0.3
A2	...	..	.
A3	..	.	.