**An R Package for QTL mapping and hotspot detection**

**Ping-Yuan Chung, …. and Chen-Hung Kao**

**Abstract**

Many biologically and economically important traits in organisms are quantitative, not qualitative. Quantitative traits are showing continuous variations and may include the traditional traits(such as yield and height in crops, weight and body fat percentage in animals, diabetes and hypertensions in human) or molecular traits (such as gene expression or protein levels) in the individuals. Quantitative traits are governed by multiple genes and easily affected by environmental factors. The genes control quantitative traits are called QTL (quantitative trait loci). The identification of individual QTL (QTL mapping) has been an important task in understanding the genetic base and architecture of quantitative traits in order to manipulate and improve them. QTL hotspots (genomic locations enriched in QTL) are a common and notable feature when collecting many QTL for various traits in many areas of biological studies. The QTL hotspots are important and attractive since they are highly informative and may harbor genes for the quantitative traits. The current statistical methods for QTL hotspot detection use either the individual-level data from the genetical genomics experiments or the summarized data from public QTL databases to proceed with the detection analysis. Together with QTL mapping and QTL hotspot detection, we can build networks among QTL hotspots, genes and traits to decipher the genetic architectures of quantitative traits in genes, genomes and genetics studies.

We provide an R package that implements some commonly used and popular statistical methods for QTL mapping and QTL hotspot detection. For QTL mapping, it consists of several functions to perform various tasks, including simulating or analyzing data, computing the significance thresholds and visualizing the QTL mapping results. The single-QTL or multiple-QTL method that allows a host of statistical models to be fitted and compared are applied to analyze the data for the estimation of QTL parameters. The models include the linear regression, permutation test, normal mixture model and truncated normal mixture model. The Gaussian stochastic process is implemented to compute the significance thresholds for QTL detection onto a genetic linkage map in the experimental populations. Two types of data, the complete genotyping or selective genotyping data, from various experimental populations, including backcross, F2, recombinant inbred (RI) populations, advanced intercrossed (AI) populations, intermated recombinant inbred (IRI) populations, and immortalized F2 (IF2) populations, are considered in the QTL mapping analysis. For QTL hotpot detection, the R code for our proposed statistical framework proposed (Yang, Wu and Kao 2019; Wu, Yang and Kao 2020) that can handle both the individual-level data and summarized QTL data is developed for QTL hotspot detection. Both the numerical and graphical outputs are presented to summarize the detection results. This proposal intends to develop an R package for QTL mapping and hotspot detection and describes a comprehensive overview of the primary functions in this package.

*Keywords:*QTL mapping, normal mixture model, truncated model, Gaussian process, permutation test, hotspot detection, permutation test, R.

1. **Introduction**

Many biologically and economically important traits in organisms are quantitative, not qualitative. These quantitative traits may include the traditional traits (such as yields and quality in rice, weight and body fat percentage in animals, diabetes and hypertension in human) or molecular traits (such as gene expression and protein levels). Quantitative traits usually vary continuously over a range of values in a population. They are likely to be affected by many genes with small effects and easily affected by environmental factors. For these reasons, individual genes are very difficult to detect by the traditional approaches, such as Mendelian method, analyses of trait means, variances, covariances, and the correlation of relatives. The genes controlling quantitative traits are called quantitative trait loci (QTL). For a long time, researchers have tried to obtain QTL information in order to explore the genetic mechanism of the quantitative traits and manipulate them, and through that to improve traits. With the fine scale genetic marker maps for various organisms, it has become possible to systematically map for individual QTL by the use of more sophistical statistical methods. Statistical methods of QTL mapping that aims to understand the underlying genetic mechanism of the quantitative traits have been well established (Lander and Botstein 1989; Haley and Knott 1992; Zeng 1993; Jansen 1993; Zeng 1994; Xu and Atchley 1995; Kao, Zeng, and Teasdale 1999; Sen and Churchill 2001; Broman 2003; Li, Ho and Kao 2014). These methods analyze the data from a well-designed experimental population to estimate the QTL parameters, including the numbers, positions, various gene actions (additive, dominance and interactive) and variance components, that can provide important insights into the genetic mechanism of quantitative traits. The QTL mapping data usually contain two parts, a set of phenotypic traits of interest and a set of genetic marker genotypes along a fine-scale genetic marker map, in the individuals from an experimental population. Here, rather than focus only on the backcross and F2 populations, we consider the experimental populations derived from two inbred lines and , such as the backcross (BC), F2, RI, AI, IRI, or IF2 populations, to describe the statistical methods that aim to tackle the several central issues, including the detection of QTL, determination of threshold values and selective genotyping, in the QTL mapping study. A comprehensive overview of the primary R functions of these statistical methods is also provided for general use in the QTL mapping community.

**1.1 Detection of QTL**

As the QTL are unknown before mapping and potentially located in any positions along the genome, the central tasks of the statistical methods focus on searching for the individual QTLs and then fitting them all into the statistical model for the estimation of QTL parameters. Lander and Botstein (1989) first proposed a QTL mapping procedure, called interval mapping (IM), to systematically search the entire genome for QTLs. The approach of IM uses one marker interval (one flanking marker pair) at a time to construct a putative QTL at a within position for modeling the relationship between a quantitative trait and the putative QTL and then testing for the presence of the QTL. For a putative QTL, say , at a specific fixed position along the genome, the IM model for individual *i* with a phenotypic trait value may be written as

, (1)

where is the genotypic value contributed by the QTL genotype, and is a residual assumed to follow . For the individuals in a population derived from two inbred lines, such as the F2 population, Q can have three possible different genotypes, homozygote (), heterozygote () and homozygote (), or two (of the three) possible different genotypes. Several genetic models have been proposed to model the relationship between the genotypic values and the QTL effects, and define the QTL effects (Cockerham1954; Van Der Veen 1959; Weir and Cockerham 1977; Kao and Zeng 2002). We may model the relationship between the three genotypic values and the QTL effects as , and , respectively, where and are the additive and dominance effects of the QTL. Then, we can further construct an equivalent model of equation (1) for an individual *i* as

(2)

where or if the QTL genotype of is *QQ , Qq or qq*. If the putative QTL is located at the marker, the IM model is a regression model. If the putative QTL is located at the position x in the marker interval (M,N), the genotypes of the QTL are not observable and needed to be inferred from its flanking markers, and the statistical model of IM is generally a normal mixture model. Given the data with *n* individuals, the likelihood function of the IM model for θ= is

 (3)

where is a normal probability density function with mean and variance , ’s correspond to the genotypic values of the 3 different QTL genotypes (, and ’s are the mixing proportions (conditional probabilities) of three QTL genotypes inferred from the two flanking markers (see Kao and Zeng 2009 for obtaining ’s in various experimental populations). By treating the normal mixture model as an incomplete-data problem, the EM algorithm (Dempster *et al.* 1977) is readily implemented to obtain the maximum likelihood estimates (MLE) of the parameters, and a likelihood ratio test (LRT) can be performed to test for the null hypothesis of no QTL (H0 : and ) at the position x. With a fine-scale genetic marker map throughout the genome, IM can be performed at all positions covered by markers to produce a continuous LRT statistic profile along chromosomes. Given a predetermined LRT threshold, the position with the significantly maximum LRT statistic in a chromosome region is regarded as the estimated QTL location. In this way, QTLs can be systematically searched and identified at the genome-wide level for the genetic dissection of the quantitative traits. As the search process for QTL needs to be performed at every position of the genome, the iterative EM algorithm may be often regarded to be complex and computationally expensive for QTL mapping (Haley and Knott 1992; Kao 2000). Haley and Knott (1992) proposed regression (REG) interval mapping, which regresses the quantitative trait value on the conditional expected genotypic value, to approximate IM to save computation cost (Haley and Knott 1992).

The approach of IM considers only one putative QTL at a time in the model and, therefore, can bias identification and estimation of QTL when multiple QTLs are located in the same linkage group. (Lander and Botstein 1989; Haley and Knott 1992; Zeng 1994). To deal with multiple QTL problems, composite interval mapping (CIM) that combines IM with multiple regression analysis was proposed (Jansen 1993; Zeng 1994). The approach of CIM is that, when testing for the putative QTL in an interval, other markers are used as covariates to remove the interference of other QTL and to reduce the residual variance such that the test can be improved. To further improve QTL mapping, Kao et al. (1999) proposed the multiple interval mapping (MIM) approach that tries to utilize multiple marker intervals simultaneously to fit multiple putative QTL in the model for QTL mapping. For, say, *m* putative QTL, Q1, Q2, ..., and Qm, without epistasis at given positions within the *m* separate marker intervals, (M1,N1), (M2,N2), ..., and (Mm,Nm), fitted into the MIM model, the one-QTL model in equation (2) can be straightforwardly expanded to a multiple-QTL model with *m* QTL as

(4)

For *m* putative QTL in the model, there are possible QTL genotypes, and the likelihood function of the MIM model for θ= becomes a normal mixture model

 (5)

(simply replacing the number 3 by in equation (3)), where ’s are the conditional probabilities of the possible QTL genotypes given the flanking marker genotypes. The general formulas by Kao and Zeng (1997) formulated based on the EM algorithm can be used to estimate the parameters of the *m* QTL. To avoid using the iterative EM algorithm, alternative approximate methods considering multiple QTL in the model for QTL mapping include the REG interval mapping (Haley and Knott 1992) and multiple imputation by Sen and Churchill (2001). Although the two approximate methods are faster in computation, their differences to the MIM method in the analysis can be significant in certain situations as given by Kao (2000) and Sen and Churchill (2001) and shown by the example below. Later, Kao (2004, 2006) and Kao and Zeng (2009) considered the specific genome structures present in the advanced populations to extend the MIM approach to various advanced populations for QTL mapping. Lee et al. (2014) developed the MIM method for the selective genotyping design (see below). With the MIM approach, the precision and power of QTL mapping could be improved. Also, epistasis between QTL, genotypic values of individuals, and heritabilities of quantitative traits can be readily estimated and analyzed to study the genetic basis of quantitative traits in various advanced populations.

**1.2 Determination of threshold values**

Under the framework of interval mapping procedure, a sequence of (correlated and uncorrelated) null hypotheses at all the genomic positions is tested by using LRT statistics. When determining the threshold values for claiming significant QTL detection (rejecting the null hypothesis), the multiple testing problem for controlling genome-wide error rates should be considered in QTL mapping. Also, it has been recognized that various factors, such as the number and size of intervals, population genome structures, and marker density, will involve in and should be considered in determining the threshold value for claiming QTL detection. Several analytical, empirical and numerical approaches, including Bonferroni adjustment, Ornstein-Uhlenbeck process, numerical simulation, permutation test and Gaussian process, have been suggested to obtain the threshold values (Lander and Botstein 1989; Churchill and Doerge 1994; Rebai *et al*. 1994; Piepho 2001; Zou et al. 2004; Chang et al. 2009; Guo 2011; Kao and Ho 2012). In general, the numerical approach, such as the permutation test or numerical simulation, needs to handle the problem of computational intractability. The analytical approach has a very cheap computational cost and requires some assumptions such as normality in derivation. We found that the Gaussian process is found to be about 7700 times faster than the permutation method in obtaining the thresholds. Notably, Chang et al. developed a score test for the detection of QTL in the backcross population, and showed that the asymptotic distribution of the score test statistics, , , at all the *k* sequential positions in the genome can be described by Gaussian stochastic process with mean zero and well-structured variance-covariance matrix. Also, since the squared score statistic,, is asymptotically equivalent to the LRT statistic (Cox and Hinkley 1974; Chang et al. 2009), the distribution of along the genome ~~under the null hypothesis~~ can be also used to assess the threshold value of the LRT statistic in QTL mapping. Guo (2011) and Kao and Ho (2012) derived more general score test statistics and Gaussian processes to evaluate the threshold values for the F2 population and the progeny populations after backcross or F2 (advanced populations). Now consider the case of the F2 population. Each of the *k* positions is associated with two score test statistics, one for additive effect and the other for the dominance effect. Let *U* denote a vector whose elements are the score test statistics at the *k* genomic positions. Then, the *U* vector has length of 2*k*. The asymptotic distribution of *U* is a Gaussian stochastic process, *i.e.* *U* is a multivariate normal distribution with probability density function

, (6)

where =0 is the mean, and is the variance-covariance matrix. Note that the structure of is associated with the population genome structure and is usually well-organized in the experimental populations. Obtaining the elements in needs to use the genotypic distributions of one, two, three and four genes of the populations. In the backcross and F2 populations, as the genome structure has the Markovian structure under Haldane map function (Haldane 1919), the genotypic distributions of three and four genes can be obtained by the genotypic frequencies of pairwise genes. For the advanced populations, their genomes no longer possess the Markovian property and are relatively more complicated so that the genotypic distributions of two, three and four genes cannot be obtained directly. The sets of transition equations proposed by Haldane and Waddington (1931), Geiringer (1944) and Kao and Zeng (2010) can be readily used to obtain the genotypic frequencies of two, three and four genes for the construction of the variance-covariance matrix. The general frameworks of the score test statistics and Gaussian processes provided by Guo (2011) and Kao and Ho (2012) can be readily used to study the behaviors of thresholds values in the genomes with different sizes and various marker densities in the different experimental populations, including the backcross, F2 and their more advanced populations, with very low computational cost. In general, given a significance level and a genome size, the threshold values should be enhanced in denser marker maps and in more advanced populations.

**1.3 Selective genotyping**

The cost of producing QTL mapping data includes both phenotyping and genotyping costs. If the total cost is not of primary concern in QTL experiments, the complete genotyping approach, where all individuals in the entire sample are genotyped and phenotyped, will be conducted for QTL analysis. However, researchers still face the situation of insufficient budgets to fully cover the expense of complete genotyping, although the cost of marker genotyping has been dropping recently. Due to a limited budget, and researchers may not be allowed to conduct complete genotyping for a large amount of individuals in QTL analysis. The selective genotyping approach has been known as a cost-saving strategy to reduce genotyping work and can still maintain nearly equivalent efficiency to complete genotyping in QTL mapping (Lebowitzet al.1987; Lander and Botstein 1989; Xu and Vogl 2000; Lee *et al.* 2014). This approach is to select individuals from the high and low extremes of the trait distribution for genotyping and keep the remaining individuals ungenotyped in the entire sample.

Suppose that, among the *n* individuals in the sample, individuals with extreme trait values (each from the upper and lower extremes) are selected for marker genotyping, and the remaining ( individuals are not genotyped. Statistical QTL mapping methods for analyzing selective genotyping data can either consider all the individuals (full data) or consider just the genotyping individuals (genotyping data) in their models for QTL detection. If only the genotyping data are utilized in the analysis, data of this sort are called centrally truncated data. Xu and Vogl (2000) and Lee *et al.* (2014) incorporated the truncated model into the mixture structure of interval mapping framework to propose a truncated normal mixture model for QTL analysis. For genotyped individuals, the likelihood function of the *m* QTL model for is

 (7)

where

is the cumulative density with trait values greater than (right truncated point) and lower than (left truncated point) so that . The details of the EM algorithm for obtaining the MLE of the parameters can be found in Lee et al. (2014). If full data are fitted into the statistical model for QTL analysis, the model likelihood can be written as

 (8)

where the first and second terms on the right-hand side are the likelihoods for the genotyped and for the ungenotyped individuals, respectively. Note that ’s are obtained from the conditional probabilities of the QTL genotypes given its flanking marker genotype, and ’s are the proportions of QTL genotypes in the ungenotyped individuals (Lee et al. 2014). In the parameter estimation, the same EM algorithm for complete genotyping (Kao and Zeng 1997) can be directly applied to obtain the MLE. It is found that the analysis using full data by model (6) performs better than that using only genotyping data by model (5), because additional information from the ungenotyped individuals is incorporated into the analysis (Xu and Vogl 2000; Lee et al. 2014). Also, selective genotyping using larger genotyping proportions, say , may maintain roughly equivalent power to complete genotyping and that using smaller genotyping proportions has difficulties doing so (Lee et al. 2014). Here, we further extend the models in equation (7) and (8) to mapping for QTL using the selective genotyping data from the various advanced populations. The key to the extension needs to consider the specific genome structures of the advanced populations to compute the proportion ’s for the model in Equation (7) and both the proportions, ’s and ’s, for the model in Equation (8). The details of the EM algorithm for obtaining the MLE of the parameters in the truncated normal mixture model and the normal mixture model in equations (7) and (8) are described in Lee *et al.* (2014).

**1.4 QTL hotspot detection**

Genome-wide QTL hotspot detection first needs to collect data with many QTL to proceed with the detection analysis. So far, both the genetical genomics experiments and public QTL databases can provide the data sets with many QTL for the hotspot analysis, but note that these two data sources have different structures. The genetical genomics experiment contains individual-level data (containing the original marker genotypes and many molecular traits) that allow to detect thousands of QTL in a single experiment. And public database (such as GRAMENE, Q-TARO, Rice TOGO browser, PeanutBase and MaizeGDB) curates thousands of summarized QTL data (containing the detected QTL, trait names and reference sources without any individual-level data) for various traditional traits from numerous independent QTL experiments. Using these two types of data, several statistical methods mainly based on permutation tests have been proposed to detect QTL hotspots. West *et al.* (2007), Wu *et al.* (2008), Li *et al.* (2010), Breitling *et al.* (2008) and Neto *et al.* (2012) developed statistical methods to detect QTL hotspots using the genetical genomics experiments. These methods may suffer from the problems of ignoring the correlation structure among traits, neglecting the magnitude of LOD scores for the QTL, or paying a very high computational cost, which often lead to detection of excessive spurious hotspots, failure to discover biologically interesting hotspots composed of a small to moderate number of QTL with strong LOD scores, and computational intractability, respectively, during the detection process. We (Yang, Wu and Kao 2019; Wu, Yang and Kao 2020) introduce a general statistical framework that can handle both types of data as well as take care of all the above concerns, including the correlation structure among traits, the magnitude of LOD scores of QTL in a hotspot and computational cost, for QTL hotspot detection. Our statistical framework operates on the QTL matrix or the EQF matrix and hence is very cheap in computation. By taking the advantages of using the individual-level data in genetical genomics experiment, the estimates of QTL parameters and the LOD scores at every position for all traits can be obtained by the QTL mapping technique and used to benefit the QTL hotspot analysis. Our statistical framework attempts to take the QTL mapping results into account to address the concerns and facilitate the hotspot detection. Two special devices, trait grouping and top profile, are deployed in the framework. In trait grouping, we first show that traits controlled by the tightly linked or pleiotropic QTL (tightly linked or pleiotropic traits) may have arbitrary values at their phenotypic or genetic correlations, and hence we use the estimated QTL positions, rather than the phenotypic or genetic correlations among traits, to make inference about the tightly linked and/or pleiotropic traits for trait grouping. Then, the permutation algorithm of Yang *et al.* (2019) is deployed to randomly shift the tightly linked and/or pleiotropic QTL together along the genome separately by trait group, accounting for the correlation structure among traits, to compute a series of EQF thresholds, ’s, for hotspot detection. Two permutation schemes, the QTL-interval permutation and the EQF-bin permutation, are devised to compute the EQF thresholds for assessing the significance of QTL hotspots.The QTL-interval permutationoperateson the QTL matrix, and then considers the genome to be circular (Cabrera *et al.* 2012) and randomly swaps the QTL intervals in the circular genome. On the contrary, the EQF-bin permutation works on the EQF matrix, and then breaks a QTL interval into several EQF bins and randomly shifts the EQF bins along the genome. The algorithm of the QTL-interval permutation with trait grouping for computing the threshold that can control GWER of at a fixed α level is described below.

Using the *m* different LOD thresholds,the LOD score matrix can be converted into the *m* QTL matrices and then *m* EQF matrices, respectively. For an EQF matrix, **,** we first obtain the EQF sum over all traits for the *W* bins and order them from highest to lowest, , , …, . Then, we define as the *n*th EQF sum of the *W* ordered observed EQF sums, and use as a test statistic for at least *n* spurious hotspots under the null hypothesis that the QTL are randomly distributed in the genome.

1. The traits with tightly linked or pleiotropic QTL are grouped together. After grouping, there are, say *R*, trait groups containing , , , , traits, respectively (.
2. Generate a new permuted QTL matrix by performing permutation in each trait group as follows: First consider the genome to be circular. For each trait group, the QTL intervals of the tightly linked or pleiotropic QTL are permuted together, and the other QTL intervals are permuted alone to obtain a permuted QTL matrix.
3. Obtain the EQF matrix, , from the permuted QTL matrix, and compute the total EQF sums over all rows for the *W* locations in, *i.e.* for . Then, order the *W* EQF sums (from highest to lowest as, , …,.
4. For a fixed hotspot number *n*, obtain and store corresponding to the *n*th EQF sum of the *W* ordered EQF sums for .
5. Repeat steps 1–4 times so that there are new permuted matrices (namely, , ,…,) for obtaining the , ,…,. The -permutation samples of , is an estimate of the null distribution of the test statistic for at least *n* spurious hotspots anywhere in the genome under the null.
6. The upper (1-)-quantile of the *B*-permutation samples generated in step 5 is the EQF threshold, denoted by , for for assessing at least *n* spurious hotspots.

If the EQF-bin permutation with trait grouping is considered, the permutation is performed on the original EQF matrix, **,** directly. In steps 2 and 3, for each trait group, the EQF bins associated with the tightly linked or pleiotropic QTL are permuted together, and those of the other QTL intervals are permuted alone to obtain a permuted EQF matrix, (see also Yang et al. 2019). Then, we order the *W* EQF sums (from highest to lowest as, , …, to obtain the threshold by using step 4 to step 6. In this way, the proposed algorithm can deploy both the EQF-bin permutation and the QTL-interval permutation to compute a series of thresholds, ’s, for ’s to assess the significance of QTL hotspots. For , 2,, *k*, where *k* is determined by (*β* is the threshold value obtained without trait grouping, *i.e.* by using the Q-method), a series of ’s ranging from the most conservative () to the most liberal () can be obtained and used for assessing the significance of different numbers of QTL hotspots.

The permutation algorithm allows to compute a series of EQF thresholds, ’s, for each of the *m* EQF matrices. Now, we denote as the EQF matrix constructed using LOD threshold and ’s as the corresponding thresholds for . We define the top threshold for a bin *w* as

Top , (5)

where is the EQF sum of the bin *w* over all traits in the matrix. That is the topthreshold is the largest EQF threshold (with the smallest *n*) for bin *w* to be significant as a QTL hotspot in the matrix. For a hotspot, the smaller the value of *n* in the top threshold is, the relatively more significant it is. Therefore, in a specific EQF architecture, the top threshold of a hotspot can be used to characterize its significance status compared to the other hotspots. For a hotspot, there are *m* top thresholds across all the *m* EQF architectures. The pattern of (the *n* values in) the *m* top thresholds can outline how the relative significance status of a hotspot changes over the different EQF architectures. For each hotspot, we profile the top thresholds and use the profile to outline the LOD-score pattern across thedifferent LOD thresholds. The top profile can then serve to characterize the types of hotspots with varying sizes and LOD-score distributions, so as to have the ability to assess the small and moderate hotspots with strong LOD scores. In this way, our framework can overcome the underestimation of threshold arising from ignoring the correlation structure among traits, and also identify the different types of hotspots with very low computational cost during the detection process.

Currently, the R/qtl software provided by Broman *et al.* (2003) are the most popular for QTL mapping and QTL hotspot detection. There are several differences between our R package and R/qtl package. First, when multiple QTL are simultaneously considered in the model, R/qtl implements the resampling approach of Sen and Churchill (2001) to approximate the MIM model, which may lose power in detection and precision in estimation greatly as compared to our approach (implicitly pointed out in Sen and Churchill 2001 and validated in our study). Second, our R package uses an efficient Gaussian process to compute the significance threshold for QTL mapping, and the R/qtl package uses the computationally expensive permutation tests to do so. Our approach is 7700 times faster than the R/qtl package in the computation of the threshold value. Third, our R package deal with more advanced populations from two inbred lines as compared to the R/qtl package. Fourth, our R package uses the statistical framework developed by (Yang, Wu and Kao 2019; Wu, Yang and Kao 2020) for QTL hotspot detection, so that it can handle two types of data. The R/qtl package can only handle individual-level data. Fifth, so far, the R/qtl package does not consider the case of selective genotyping in QTL mapping. We have considered selective genotyping approach in the R package. As a result, our R package can provide greater power and precision with cheaper computational cost for QTL mapping and QTL hotspot detection.

1. **The R package QTL.EMmapping**

We provide the R package that produces both the numerical and graphical summaries of QTL mapping. The function set **QTL.EMmapping** contains 8 main functions for QTL analyzation and some minor functions to support the main functions. It can handle various populations from the different breeding programs, including backcross (BC), F2, recombinant inbred (RI) populations, advanced intercrossed (AI) populations, intermated recombinant inbred (IRI) populations and immortalized F2 (IF2) populations. The EM algorithm is used to obtain the maximum likelihood estimates (MLEs) of the QTL effects and LRT statistic for the interval mapping (IM) in **EM.IM()** and for the multiple interval mapping (MIM) in **EM.MIM()**. For selective genotyping, the function **EM.IM2()** and **EM.MIM2()** are usedto obtain the MLEs for IM and MIM through the proposed model or the truncated model. The function **LRTthre()**, **D.make()**, and **Q.make()** provide the likelihood ratio test (LRT) threshold of IM, design matrix of QTL effects, and conditional probability matrix of QTL genotypes respectively. The function **progeny()** is used to simulate the phenotype and genotype data for a given generation.

2.1 Inputs

In order to build suitable data for the functions in **QTL.EMmapping**, it is necessary to load the data in the R environment and revise the data format or generate data with appropriate format in R. Csv files or txt files are loaded easily into R environment by the function **read.csv()** or **read.table()** respectively. **QTL.EMmapping** takes four main inputs: QTL information (**QTL**), marker information (**marker**), genotypes (**geno**), and phenotypes (**y**).

The marker information data is a *k*\*2 matrix, where *k* is the number of markers. The first column is the chromosome number of the marker, and the second column is the marker position. The unit of position can be morgan (M) or centimorgan (cM). Table 1 shows the format of marker information data.

The QTL information data should be a *q*\*2 matrix, where *q* is the number of QTLs. The format is similar to the marker information data and can be seen in Table 1.

The genotype data is an *n\*k* matrix, *n* is the number of individuals and *k* is the number of markers. The coding for the genotypes are 2 for AA, 1 for Aa, 0 for aa, and *NA* for missing value, respectively. Table 2 shows the format of the coding for marker genotypes.

The phenotype data is a vector with length *n*, where *n* is the number of individuals. The elements of phenotype data are numerical values or *NA* for missing values.

Table 1. The format example of marker information data

|  |  |
| --- | --- |
| chr | position |
| 1 | 0 |
| 1 | 24 |
| 1 | 40 |
| 1 | 103 |
| ⁝ | ⁝ |
| 12 | 72 |
| 12 | 126 |

Table 2. The format example of genotype data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | ind1 | ind2 | ind3 | ind4 | ind5 | … | indn |
| marker1 | 2 | 1 | 1 | 2 | 0 | … | 2 |
| marker2 | 2 | 1 | 0 | 0 | 1 | … | 1 |
| marker3 | 2 | 2 | NA | 1 | 1 | … | 0 |
| marker4 | 0 | 0 | 1 | 0 | NA | … | 2 |
| marker5 | 1 | 0 | 0 | 2 | 2 | … | 2 |
| ⁝ | ⁝ | ⁝ | ⁝ | ⁝ | ⁝ | ⁝ | ⁝ |
| markerk | 1 | 1 | 0 | 0 | 1 | … | 0 |

2.2 Command and options

Once the input data are already, the IM analyze can be run through the command

***R> IM.out=EM.IM(marker ,geno, y, method="EM", type="RI" ,ng=2, D.matrix=NULL,  
+ cM=T, speed=1, conv = 10^-5, d.eff=T, LRT.thre=T, simu=1000,  
+ lv=0.05, detect=T, minQTLdist=15, chart = "chr", console = F)***

The arguments are described in Table 3 and the numerical results are shown below.

Table 3: The list of parameters used in **EM.IM()** and **EM.IM2()**.

|  |  |
| --- | --- |
| Argument | Description |
| **# yu** | The phenotype value of the individuals that are without genotyping. |
| **# sele.g** | If **sele.g="n"**, it will consider that the data is not a selective genotyping data but the complete genotyping data. If **sele.g="p"**, it will consider that the data is a selective genotyping data, and use the proposed model (Lee 2014) to analyze. If **sele.g="t"**, it will consider that the data is a selective genotyping data, and use the truncated model (Lee 2014) to analyze. Note that the **yu** must be input when **sele.g="p"**. |
| **# tL** | To decide the lower truncation point when **sele.g="t"**. Note that when **sele.g="t"** and **tL=NULL**, the **yu** must be input and the function will consider the minimum of **yu** as the lower truncation point. |
| **# tR** | To decide the upper truncation point when **sele.g="t"**. Note that when **sele.g="t"** and **tR=NULL**, the **yu** must be input and the function will consider the maximum of **yu** as the upper truncation point. |
| **method** | Two methods, the interval mapping (Lander and Botstein 1989) based on the EM algorithm (**method="EM"**) and the regression interval mapping method of Haley and Knott (1992) (**method="REG"**), are considered. |
| **type** | The types of populations for the dataset. The codes include backcross (**type="BC"**), AI populations (**type="AI"**), and RI populations (**type="RI"**). |
| **ng** | The generation number of the population. For example, the BC1 population is **type="BC"** with **ng=1**; the AI F3 population is **type="AI"** with **ng=3**. |
| **D.matrix** | |  |  |  | | --- | --- | --- | | AI or RI |  |  | |  | a1 | d1 | | 2 | 1 | 0.5 | | 1 | 0 | -0.5 | | 0 | -1 | 0.5 |   The design matrix of the IM model. If **D.matrix=NULL**, the design matrix will be the constructed using Cockerham’s model) for the BC, AI or RI populations see below. Users can also define the design matrix using other models with the same dimension.   |  |  | | --- | --- | | BC |  | |  | a1 | | 2 | 0.5 | | 1 | -0.5 | |
| **cM** | A logical parameter. If the unit of marker position is centi-Morgan that **cM=T**. Or if the unit of marker position is morgan that **cM=F**. |
| **speed** | The walking speed for QTL search in cM. |
| **conv** | The convergent criterion of the EM algorithm. The E and M steps will be iterated until a convergent criterion is satisfied. |
| **d.eff** | A logical parameter in the AI or RI population to specify whether the dominant effect will be considered in the parameter estimation or not. |
| **LRT.thre** | If **LRT.thre = T**, the LRT threshold will be computed based on the Gaussian stochastic process (Kao and Ho 2012). Or users can input a numerical value as the LRT threshold. |
| **simu** | To decide how many simulation samples will be used to compute the LRT threshold using the Gaussian process. |
| **lv** | The type I error rate for the LRT threshold. |
| **detect** | A logical parameter to decide whether the significant QTLs whose LRT statistic is larger than the LRT threshold will be shown in the output dataset or not. |
| **minQTLdist** | The minimum distance cM of different QTLs being claimed to be significant on the same chromosome. |
| **chart** | To decide the type of plot in output. Two types of plot are considered. If **chart = "chr"**, it produces plots for individual chromosomes. Or if **chart = "all"**, it produces one plot for all chromosomes. Otherwise, no plot will be made. |
| **console** | A logical parameter. To decide whether the process of algorithm will be shown in the R console or not. |

***R> str(IM.out)*List of 3  
 $ effect :'data.frame': 600 obs. of 5 variables:  
 ..$ chr: num [1:600] 1 1 1 1 1 1 1 1 1 1 ...  
 ..$ cM : num [1:600] 1 2 3 4 5 6 7 8 9 10 ...  
 ..$ a1 : num [1:600] 2.95 3.2 3.45 3.68 3.89 ...  
 ..$ d1 : num [1:600] 0.767 0.803 0.836 0.862 0.881 ...  
 ..$ LRT: num [1:600] 19.6 22.5 25.9 29.5 33.3 ...  
 $ LRT.threshold: Named num 13.9  
 ..- attr(\*, "names")= chr "95%"  
 $ detect.QTL :'data.frame': 4 obs. of 5 variables:  
 ..$ chr: num [1:4] 1 1 4 4  
 ..$ cM : num [1:4] 25 52 109 134  
 ..$ a1 : num [1:4] 5.71 5.67 2.82 3.07  
 ..$ d1 : num [1:4] 1.364 0.914 1.316 -0.677  
 ..$ LRT: num [1:4] 141.4 129.7 15.2 19.8**

Where **effect** contains the LRT statistics and effects at each position of the chromosomes; **LRT.threshold** is the LRT threshold which obtained by the Gaussian stochastic process (Kuo 2011; Kao and Ho 2012); **detect.QTL** is the information about the detected QTLs. And the figure results depend on the parameter setting in “**chart**” whose upper part is the LRT profile and lower part is the profile for the effects.

The function **EM.IM2()** is used for selective genotyping, and it can be run through the command

***R> IM.out2=EM.IM2(marker, geno, y, yu=NULL, sele.g="n", tL=NULL, tR=NULL,  
+ method="EM", type="RI", ng=2, D.matrix=NULL, cM=T, speed=1,  
+ conv = 10^-5, d.eff=T, LRT.thre=T, simu=1000, lv=0.05,   
+ detect=T, minQTLdist=15, chart = "chr", console = F)***

The parameters are similar with **EM.IM()** and shown in Table 3, where the arguments only for **EM.IM2()** are shown with the # mark. Note that, in this function, the input of phenotype is divided into two parts that the individuals with genotyping (**y**) and the individuals without genotyping (**yu**).

The results of **EM.IM2()** are similar to **EM.IM()** that contain the numerical results and figure results. Furthermore, it will be shown the model used for analyzing the selective genotyping data.

The LRT threshold of IM is able to be obtained in the analyze of **EM.IM()** and **EM.IM2()**. Also, it can be calculated separately through the command

***R> LRTthre(marker, type="RI", ng=2, ns=200, gv=25, cM=T, d.eff=T, simu=1000, speed=1,  
+ lv=0.05, console=F)* 95%   
13.83813**

The parameters are similar to **EM.IM()** and shown in Table 3. The parameters **ns** and **gv** present the number of individuals and the genetic variance respectively. Note that, under normal circumstances, these two parameters may not affect the result significantly.

For MIM analyze, it can be run through the command

***R> MIM.out=EM.MIM(D.matrix, cp.matrix, y, E.vector=NULL, X=NULL, beta=NULL,  
+ variance=NULL, conv=10^-5, console=F)***

The arguments of **EM.MIM()** are described in Table 4.

Table 4: The list of parameters used in **EM.MIM()** and **EM.MIM2()**.

|  |  |
| --- | --- |
| Argument | Description |
| **# yu** | The phenotype value of the individuals that are without genotyping. |
| **# sele.g** | If **sele.g="n"**, it will consider that the data is not a selective genotyping data but the complete genotyping data. If **sele.g="p"**, it will consider that the data is a selective genotyping data, and use the proposed model (Lee 2014) to analyze. If **sele.g="t"**, it will consider that the data is a selective genotyping data, and use the truncated model (Lee 2014) to analyze. Note that the **yu** must be input when **sele.g="p"**. |
| **# tL** | To decide the lower truncation point when **sele.g="t"**. Note that when **sele.g="t"** and **tL=NULL**, the **yu** must be input and the function will consider the minimum of **yu** as the lower truncation point. |
| **# tR** | To decide the upper truncation point when **sele.g="t"**. Note that when **sele.g="t"** and **tR=NULL**, the **yu** must be input and the function will consider the maximum of **yu** as the upper truncation point. |
| **# cM** | A logical parameter. If the unit of marker position is centi-Morgan that **cM=T**. Or if the unit of marker position is morgan that **cM=F**. |
| **# type** | The types of populations for the dataset. The codes include backcross (**type="BC"**), AI populations (**type="AI"**), and RI populations (**type="RI"**). |
| **# ng** | The generation number of the population. For example, the BC1 population is **type="BC"** with **ng=1**; the AI F3 population is **type="AI"** with **ng=3**. |
| **E.vector** | The initial value for QTL effects. The number of elements corresponds to the column dimension of the design matrix. If **E.vector=NULL**, the initial value will be 0 for all effects. |
| **X** | The initial value for design matrix of the fixed factors except for QTL effects. If **X=NULL**, the initial value will be numerical values 1. |
| **beta** | The initial value for effects of the fixed factors except for QTL effects. If **beta=NULL**, the initial value will be 0. |
| **variance** | The initial value for variance. If **sigma=NULL**, the initial value will be the standard deviation of phenotype values. |
| **conv** | The convergent criterion of EM algorithm. The E and M steps will be iterated until a convergent criterion is satisfied. |
| **console** | A logical parameter. To decide whether the process of algorithm will be shown in the R console or not. |

The **D.matrix** is the design matrix of QTL effects which is a *g\*p* matrix, where *g* is the number of possible QTL genotypes, and *p* is the number of effects considered in the MIM model. The design matrix is constructed according to the Cockerham’s model. The row elements are the coded variables for each of the all possible QTL genotypes that can be seen in Table 5. For example, in Table 5, the first row is for the genotype with double homozygotes 22. The first four elements are 1, -1/2, 1 and -1/2 for the marginal effects, and the latter four elements are 1, -1/2, -1/2 and 1/4 are for the epistatic effects. The row names and column names of the design matrix should be assigned as the possible QTL genotypes (22, 21, 20, etc.) and the name of effects (a1, d1, a1:a2, etc., or other appropriate names depend on users). The design matrix can be generated easily through the command

***R> D.matrix=D.make(nQTL, type="RI", a=1:nQTL, d=1:nQTL, aa=NULL, dd=NULL,  
+ ad=NULL)***

Where **nQTL** is the number of QTLs and the arguments of **D.make ()** are described in Table 7. The output of **D.make()** is a design matrix whose format can be seen in Table 5.

The **cp.matrix** is the conditional probability matrix which is an *n\*g* matrix, where *n* is the number of individuals, and *g* is the number of possible genotypes of QTLs. The row elements of the conditional probability matrix are the conditional probabilities of QTL genotypes for the *n* individuals. And their sum must be 1 that seen in Table 6. The conditional probability matrix can be generated easily through the command

***R> cp.matrix=Q.make(QTL, marker, geno, interval=F, cM=T, type="RI", ng=2)[[nQTL+1]]***

The parameters are similar to **EM.IM()** and shown in Table 3. Where the parameter **interval** is a logical parameter which presents that when the QTL with the same position of a marker, whether the marker will be skipped and not be regarded as a flanking marker or not. The output of **Q.make()** contains *k* conditional probability matrices for the *k* flanking marker pairs (the *k* Q-matrices) and a conditional probability matrix of each QTL for all individuals (the cp-matrix, if the genotype data of testing population is input). Note that, if **geno=NULL**, the function can be run too and the result will contain *k* Q-matrices but no cp-matrix.

After input the design matrix conditional probability matrix, the result of MIM can be obtained by the function **EM.MIM()** that shown below.

***R> str(MIM.out)*  
List of 6  
 $ E.vector : num [1:8, 1] 3.855 0.433 2.897 0.671 0.408 ...  
 ..- attr(\*, "dimnames")=List of 2  
 .. ..$ : chr [1:8] "a1" "d1" "a2" "d2" ...  
 .. ..$ : NULL  
 $ beta : num [1, 1] 0.577  
 $ variance : num [1, 1] 9.39  
 $ PI.matrix : num [1:100, 1:81] 0 0 0 0 0 ...  
 ..- attr(\*, "dimnames")=List of 2  
 .. ..$ : NULL  
 .. ..$ : chr [1:81] "2222" "2221" "2220" "2212" ...  
 $ LRT : num 236  
 $ iteration.time : num 24**

Where **E.vector** is the effect of QTLs designated in the design matrix; **beta** is the estimate of the fixed factors; **variance** is the estimate of variance; **PI.matrix** is the posterior probabilities matrix; **LRT** is the LRT statistic; and **iteration.time** is the iteration time.

Table 5. The format example of design matrix

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a1 | d1 | a2 | d2 | a1:a2 | d1:d2 | a1:d2 | a2:d1 |
| 22 | 1 | -0.5 | 1 | -0.5 | 1 | 0.25 | -0.5 | -0.5 |
| 21 | 1 | -0.5 | 0 | 0.5 | 0 | -0.25 | 0.5 | 0 |
| 20 | 1 | -0.5 | -1 | -0.5 | -1 | 0.25 | -0.5 | 0.5 |
| 12 | 0 | 0.5 | 1 | -0.5 | 0 | -0.25 | 0 | 0.5 |
| 11 | 0 | 0.5 | 0 | 0.5 | 0 | 0.25 | 0 | 0 |
| 10 | 0 | 0.5 | -1 | -0.5 | 0 | -0.25 | 0 | -0.5 |
| 02 | -1 | -0.5 | 1 | -0.5 | 1 | 0.25 | -0.5 | -0.5 |
| 01 | -1 | -0.5 | 0 | 0.5 | 0 | -0.25 | 0.5 | 0 |
| 00 | -1 | -0.5 | -1 | -0.5 | -1 | 0.25 | -0.5 | 0.5 |

Table 6. The format example of conditional probability matrix

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 22 | 21 | 20 | 12 | 11 | 10 | 02 | 01 | 00 |
| ind1 | 0 | 0.002 | 0.002 | 0.002 | 0.495 | 0.495 | 0 | 0.002 | 0.002 |
| Ind2 | 0.005 | 0.985 | 0.005 | 0 | 0.005 | 0 | 0 | 0 | 0 |
| Ind3 | 0.002 | 0.495 | 0.002 | 0.002 | 0.495 | 0.002 | 0 | 0 | 0 |
| Ind4 | 0.496 | 0.497 | 0.001 | 0.002 | 0.002 | 0.001 | 0 | 0.001 | 0 |
| Ind5 | 0 | 0 | 0 | 0.005 | 0 | 0 | 0.990 | 0.005 | 0 |
| ⁝ | ⁝ | ⁝ | ⁝ | ⁝ | ⁝ | ⁝ | ⁝ | ⁝ | ⁝ |
| Indn | 0.249 | 0.249 | 0.001 | 0.249 | 0.249 | 0.001 | 0.001 | 0.001 | 0 |

Table 7: The list of parameters used in **D.make()**.

|  |  |
| --- | --- |
| Argument | Description |
| **type** | The population type of the dataset. Include backcross (**type="BC"**), AI population (**type="AI"**), and RI population (**type="RI"**). Note that if **type="BC"**, the design matrix contain only additive effect and additive by additive interaction. If **type="AI"** or **type="RI"**, that will contain additive and dominance effects and all interaction. |
| **a** | The additive effects. Users can input a vector to decide the additive effects of which QTL will be considered in this design matrix. |
| **d** | The dominant effects. Users can input a vector to decide the dominant effects of which QTL will be considered in this design matrix. |
| **aa** | The additive-by-additive interaction. Tow format can be used in this parameter. One format is vector, in which every two elements indicate a combination of additive-by-additive interaction. For example, **aa=c(1,3,2,4,5,6)**, indicates that the interaction between QTL1 and QTL3, the interaction between QTL2 and QTL4, and that between QTL5 and QTL6 will be considered in the design matrix. The other format is a *2\*i* matrix, where *i* is the number of combination of interaction, and each column indicates the two interacting QTL. For example, the matrix format for the interactions between QTL1 and QTL3, between QTL2 and QTL4, and between QTL5 and QTL6 is shown below, and the code is **aa=matrix(c(1,3,2,4,5,6),2,3)**.   |  |  |  |  | | --- | --- | --- | --- | |  | 1 | 2 | 5 | |  | 3 | 4 | 6 |   Besides, if **aa=”all”**, all combinations of additive-by-additive interaction will be considered. |
| **dd** | The dominant-by-dominant interaction. The format is the same as that in **aa**. |
| **ad** | The additive-by-dominant interaction. The format is the same as that in **aa**. Note that, in each pair of QTLs, the first element indicates the additive effect, and the second element indicates the dominant effect. |

For the MIM analyze of selective genotyping, it can be run through the command

***R> MIM.out2=EM.MIM2(QTL, marker, geno, D.matrix, cp.matrix=NULL, y, yu=NULL,  
+ sele.g="n", tL=NULL, tR=NULL, cM=T, type="RI", ng=2,  
+ E.vector=NULL, X=NULL, beta=NULL, variance=NULL,  
+ conv=10^-5, console=F)***

The parameters are similar with **EM.MIM()** and shown in Table 3, where the arguments only for **EM.MIM2()** are shown with the # mark.. Note that, in this function, the input of phenotype is divided into two parts that the individuals with genotyping (**y**) and the individuals without genotyping (**yu**). If the input **cp.matrix** is null, the conditional probability matrix will be calculated for the given population.

The results of **EM.MIM2()** are similar to **EM.MIM()** and it will be shown the model used for analyzing the selective genotyping data.

2.3 The simulation example

For the simulation analyze, the simulation data of specific population can be generated by the command

***R> simulate.data=progeny(QTL, marker, type="RI", ng=2, E.vector=NULL, h2=0.5,  
+ size=200)***

The arguments are described in Table 8 and the output are the information of simulative data and shown below. Where **phe** is the phenotype values; **E.vector** is the QTL effects; **marker.prog** is the marker genotype; **QTL.prog** is the QTL genotype.

***R> str(simulate.data)*List of 4  
 $ phe : num [1:200] 2.36 3.85 1.31 4.13 3.48 ...  
 $ E.vector : Named num [1:32] 1 1 1 1 1 1 1 1 1 1 ...  
 ..- attr(\*, "names")= chr [1:32] "a1" "d1" "a2" "d2" ...  
 $ marker.prog: num [1:200, 1:44] 1 1 1 2 1 1 1 2 2 2 ...  
 $ QTL.prog : num [1:200, 1:4] 1 2 1 2 2 1 0 1 1 2 ...**

The process of the simulation analyzation example in R environment is shown below. First, load the functions and set seed.

***R> load("QTL.EMmapping.RDATA")  
R> set.seed(4586)***

Table 8: The list of parameters used in **progeny()**.

|  |  |
| --- | --- |
| Argument | Description |
| **type** | The population type of the dataset. Include backcross (**type="BC"**), AI population (**type="AI"**), and RI population (**type="RI"**). |
| **ng** | The generation number of the population. For example, the BC1 population is **type="BC"** and **ng=1**; the AI F3 population is **type="AI"** and **ng=3**. |
| **E.vector** | Set the effect of QTLs. It should be a named vector, and the name of elements should be the effects of QTLs and their interaction. For example, the additive effect of QTL1 is coded to “a1”; the dominant effect of QTL2 is coded to “d2”; and the interaction of the additive effect of QTL2 and the dominant effect of QTL1 is coded to “a2:d1”. So that, if the additive effect of QTL1 is 2, the dominant effect of QTL2 is 5, and the interaction of the additive effect of QTL2 and the dominant effect of QTL1 is 3, the user should input **E.vector=c("a1"=2, "d2"=5, "** **a2:d1"=3)**. If **E.vector =NULL**, there will be 1 for all effects. |
| **h2** | Set the heritability for simulated phenotypes. It should be a number between 0 and 1. |
| **size** | The population size of simulated progeny. |

And then, build the genotype data and QTL effects.

***R> marker=cbind(rep(1:3,each=11),rep(seq(0,100,10),3))  
R> QTL=cbind(c(1,1,2),c(23,77,55))  
R> eff=c("a1"=-10,"a2"=12,"a3"=8,"a1:a3"=1)***

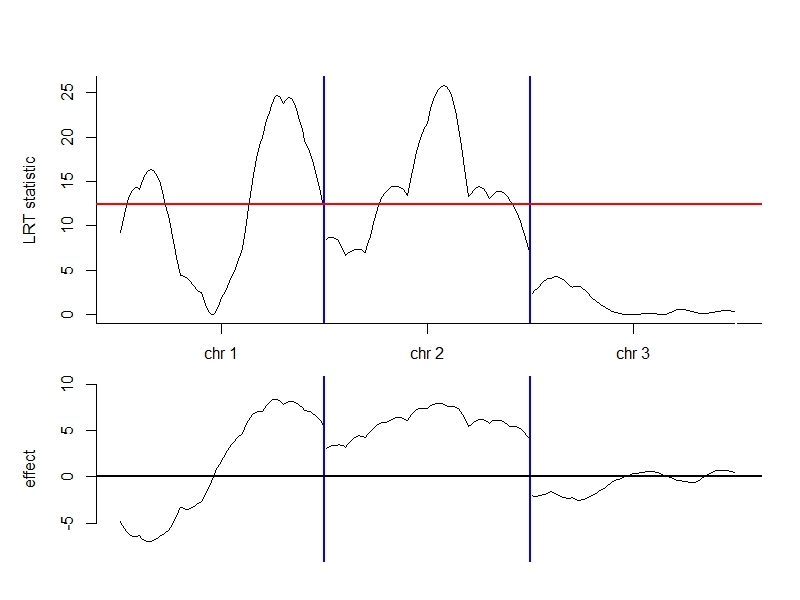
By the data shown above, the simulative phenotype and genotype data of F2 population can be obtained through the command

***R> testdata=progeny(QTL, marker, type="RI", ng=2, E.vector=eff, h2=0.5, size=200)  
R> y=testdata$phe  
R> geno=testdata$marker.prog***

This way, the IM analyze can be carried out.

***R> IMtest=EM.IM(marker, geno, y, method="EM", type="RI", ng=2, chart="all")  
R> detQTL=IMtest$detect.QTL***

The figure result can be seen in the Figure 1, where the red line in the upper part is the LRT threshold. And the results of detected QTLs are shown below.

  
Figure 1: The figure result of simulation data by **EM.IM()**.

***R> detQTL* chr cM a1 d1 LRT  
16 1 16 -6.994105 -2.6426981 16.32022  
77 1 77 8.390089 1.0414881 24.67004  
158 2 58 7.863031 -0.3525851 25.75889**

The detected QTLs are used in the MIM analyze to estimate the epistasis effects of QTLs. And the results are shown below.

***R> D.matrix=D.make(3, type="RI", aa="all")  
R> cp.matrix=Q.make(detQTL[,1:2], marker, geno, type="RI", ng=2)[[4]]  
R> MIMtest=EM.MIM(D.matrix, cp.matrix, y, conv=10^-7)  
R*** ***> MIMtest$E.vector  
 a1 d1 a2 d2 a3   
-10.1357903 -1.5855165 10.9840691 2.6727373 8.9346010   
 d3 a1:a2 a1:a3 a2:a3   
 -0.5602120 0.7881691 -4.0567548 2.7760219  
R> MIMtest$beta* [,1]  
[1,] 0.4868954**

***R> MIMtest$variance* [,1]  
[1,] 156.8938 *R> MIMtest$LRT*[1] 141.6738 *R> MIMtest$iteration.time*[1] 31**

1.4 The real data example

The real data example uses the yeast data (Brem and Kruglyak 2005) to illustrate the QTL mapping for selective genotyping data. The yeast data is backcross population which contains 5740 traits and 1072 markers. It has been pre-processed and can be loaded in to the R environment.

***R> load("yeast.process.RDATA")  
R> geno=yeast.process$geno  
R> marker=yeast.process$marker  
R> pheno=yeast.process$pheno***

The 124th trait is used to be the example of selective genotyping. To build the selective genotyping data, the genotype data of the individuals with the upper or lower 25% phenotype value are kept and the genotype data of others are removed.

***R> y0=pheno[,124]  
R> quantile(y0)* 0% 25% 50% 75% 100%   
 -2.3718533 -0.6606779 0.0000000 0.6606779 2.3718533  *R> y=y0[y0>quantile(y0)[4] | y0<quantile(y0)[2]]  
R> yu=y0[y0>=quantile(y0)[2] & y0<=quantile(y0)[4]]  
R> geno.s=geno[y0>quantile(y0)[4] | y0<quantile(y0)[2],]***

Load the functions and set seed to carry out the IM for this selective genotyping data. The result of detected QTLs is shown below and the figure result can be seen in Figure 2. Where the red line in the upper part is the LRT threshold.

As the contrast, the IM analyze of all genotype data is carried out by the function **EM.IM()**. The result of detected QTLs is shown below and the figure result can be seen in Figure 3.

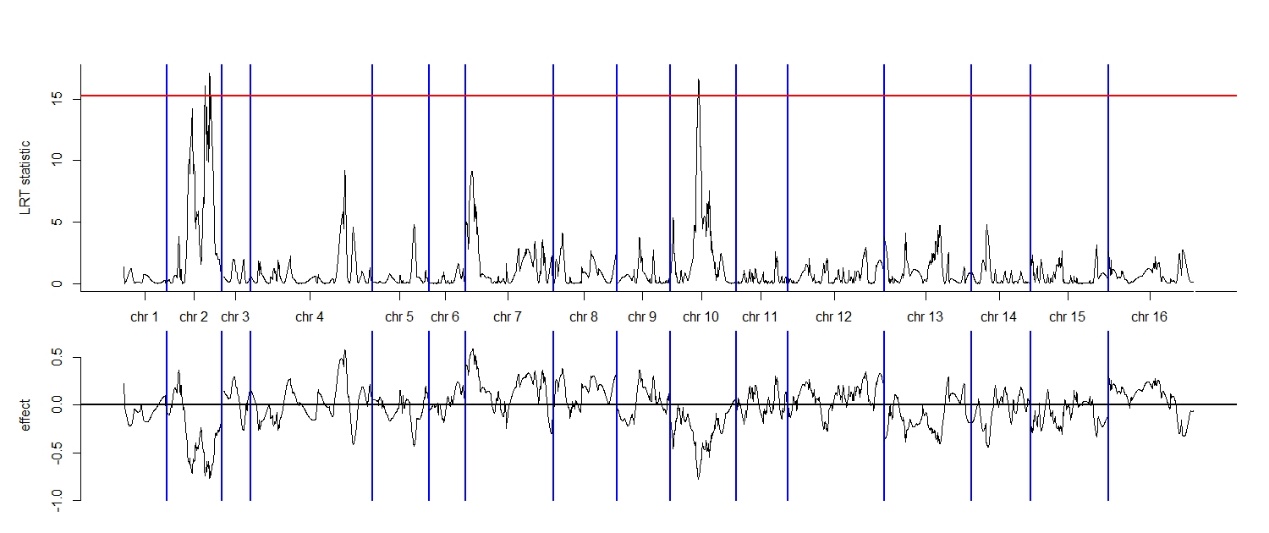


Figure 2: The figure result of selective genotyping yeast data by **EM.IM2()**.

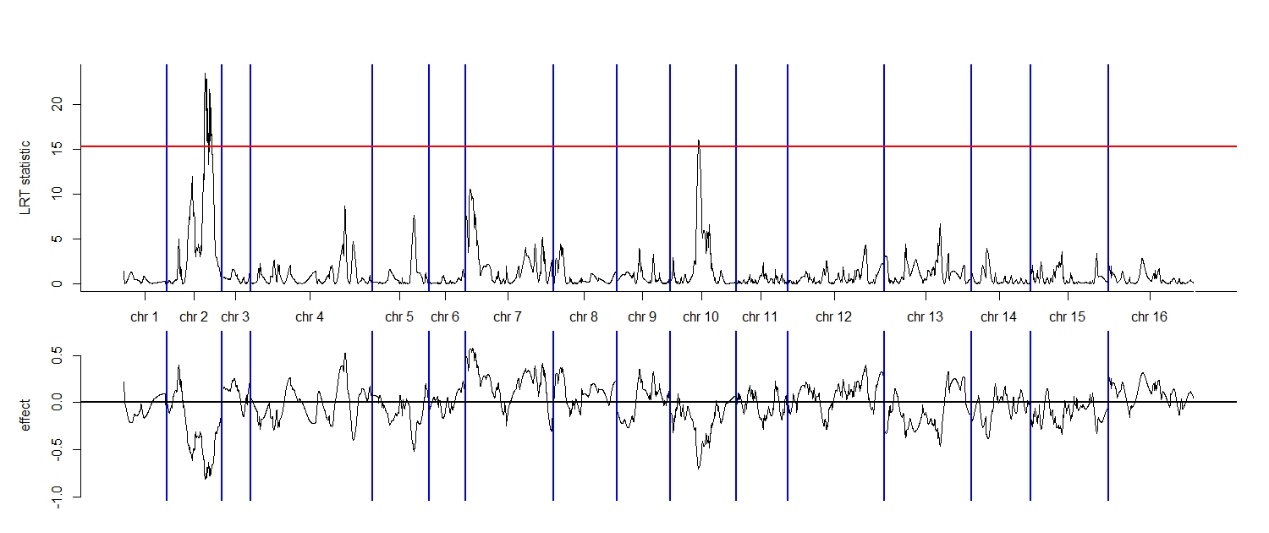


Figure 3: The figure result of complete genotyping yeast data by **EM.IM()**.

***R> load("QTL.EMmapping.RDATA")  
R> set.seed(2500)  
R> IMtest2=EM.IM2(marker, geno.s, y, yu, sele.g="p", type="BC", ng=1, chart = "all")  
R> dectQTL2=IMtest2$detect.QTL  
R> dectQTL2* chr cM a1 LRT  
480 2 224 -0.7436974 16.11362  
508 2 252 -0.7658264 17.11409  
3392 10 166 -0.7712131 16.63349  
*R> IMcon=EM.IM(marker,geno,y0,type="BC",ng=1,chart = "all")  
R> IMcon$detect.QTL*  
 chr cM a1 LRT  
480 2 224 -0.8107525 23.47312  
508 2 252 -0.7824505 21.67188  
3392 10 166 -0.6971368 16.01357**

It can be seen obviously that the results of detected QTL from the complete genotyping data and the selective genotyping data are very similar.

The detected QTLs are used in the MIM analyze to estimate the epistasis effects of QTLs. And the results are shown below.

***R> D.matrix=D.make(3, type="BC", aa="all")  
R> MIMtest2=EM.MIM2(dectQTL2[,1:2], marker, geno.s, D.matrix, y=y, yu=yu,   
+ sele.g="p", type="BC", ng=1)  
R> MIMtest2$E.vector* a1 a2 a3 a1:a2 a1:a3 a2:a3   
 -0.4594344 -0.5236613 -0.8155428 -0.6173572 -0.4524002 -0.0372447  *R> MIMtest2$beta*[1] 0.1053326 *R> MIMtest2$varence*[1] 0.5684126 *R> MIMtest2$LRT*[1] 47.23932 *R> MIMtest2$time*[1] 40**

1. **The R package of QTL hotspot detection**

3.1Inputs

To carrying out the R function set **QTLhotsot**, the LOD data and the information of bins on the chromosomes should be input. The LOD data is a *t\*p* matrix, where *t* is the number of traits; *p* is the number of bin on the chromosomes; the elements of this matrix are the LOD score for each bin over the traits. It can be seen in Table 9. The bin information is an *n*\*2 matrix, where *n* is the number of chromosomes. The first column is the serial number of chromosomes; the second column donates how many bins in each chromosome shown in Table 10.

Table 9. The format example of LOD data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | bin1 | bin 2 | bin 3 | bin 4 | bin 5 | … | bin n |
| trait1 | 0.047 | 0.116 | 0.209 | 0.313 | 0.342 | … | 0.358 |
| trait 2 | 0.095 | 0.176 | 0.274 | 0.376 | 0.301 | … | 0.342 |
| trait 3 | 0.798 | 0.670 | 0.533 | 0.394 | 0.342 | … | 0.284 |
| trait 4 | 0.363 | 0.321 | 0.272 | 0.219 | 0.192 | … | 0.149 |
| trait 5 | 0.017 | 0.010 | 0.005 | 0.002 | 0.001 | … | 0.000 |
| ⁝ | ⁝ | ⁝ | ⁝ | ⁝ | ⁝ | ⁝ | ⁝ |
| trait*t* | 0.683 | 0.593 | 0.471 | 0.336 | 0.304 | … | 0.271 |

Table 10. The format example of bin information

|  |  |
| --- | --- |
| chromosome | # of bin |
| 1 | 256 |
| 2 | 324 |
| 3 | 160 |
| 4 | 723 |
| ⁝ | ⁝ |
| 15 | 463 |
| 16 | 513 |

* 1. Command and options

Once the input data are already, the QTL detection can be run through the command

***R>*** ***LOD.QTLdetect.result =LOD.QTLdetect(LOD, chro, thre=3, gape=20)***

The arguments are described in Table 11 and the numerical results are shown below.

Table 11: The list of parameters used in **LOD.QTLdetect()**.

|  |  |
| --- | --- |
| Argument | Description |
| **LOD** | The LOD data as Table 9. |
| **chro** | The bin information as Table 10. |
| **thre** | The LOD threshold. The LOD score under this threshold will be calculated as 0. |
| **gape** | The peak of LOD score on the chromosome over the gape of bins will be detected as a QTL. |

***> str(LOD.QTLdetect.result)***

**List of 6**

**$ detect.QTL.number : num [1:5740] 1 2 0 1 2 2 1 0 0 3 ...**

**$ QTL.matrix : num [1:5740, 1:6336] 0 0 0 0 0 0 0 0 0 0 ...**

**$ EQF.matrix : num [1:5740, 1:6336] 0 0 0 0 0 0 0 0 0 0 ...**

**$ linkage.QTL.number: 'table' int [1:5(1d)] 86093 5031 654 56 6**

**..- attr(\*, "dimnames")=List of 1**

**.. ..$ link: chr [1:5] "0" "1" "2" "3" ...**

**$ threshole : num 3**

**$ chro : num [1:16, 1:2] 1 2 3 4 5 6 7 8 9 10 ...**

**..- attr(\*, "dimnames")=List of 2**

**.. ..$ : NULL**

**.. ..$ : chr [1:2] "chromosome" "# of bins"**

**detect.QTL.number** donates the number of detected QTL of traits. **QTL.matrix** donates the QTL position, where the elements 1 donate the position of QTL; elements 0 donate the bins whose LOD score is under the LOD threshold; other positions is shown as NA. **EQF.matrix** donates the EQF value of each bin. **linkage.QTL.number** donates the linkage QTL number of all detected QTL. **threshole** and **chro** are as the arguments the user input. And the figure result of the EQF values over the bins can be seen in Figure 4.

Once the user obtains the detection result, the hotspot detection by cluster permutation process can be run through the command

***R> permutation=EQF.permu(LOD.QTLdetect.result, ptime=1000, alpha=0.05, Q=TRUE,***

***+ plot.all=TRUE, plot.cr=TRUE)***

The arguments are described in Table 12 and the numerical results are shown below.

Table 12: The list of parameters used in ***EQF.permu* ()**

|  |  |
| --- | --- |
| Argument | Description |
| **LOD.QTLdetect.result** | The output data of **LOD.QTLdetect()**. |
| **ptime** | The permutation time. |
| **alpha** | The type 1 error rate of detecting the hotspot. |
| **Q** | If being TURE, the function will further carrying out the population of Q method as the control group and shown as β in the figure result. |
| **plot.all** | If being TURE, output one figure of the EQF values over the bins. |
| **plot.cr** | If being TURE, output the figures the EQF values over the bins of each chromosome. |

***> str(permutation)***

**List of 5**

**$ cluster.number : Named num [1:522] 520 3917 3236 3 8 ...**

**..- attr(\*, "names")= chr [1:522] "ngroup" "nQTL" "ng1" "ng2" ...**

**$ cluster.id :List of 520**

**..$ : num [1:3236] 2 4 5 6 7 10 11 12 14 18 ...**

**..$ : num [1:3] 1913 2551 2552**

**..$ : num [1:8] 182 946 3905 3975 4145 ...**

**……**

**$ cluster.matrix : num [1:520, 1:6336] 0.251 2.573 0 0 0 ...**

**$ permu.matrix.cluster: num [1:1000, 1:6336] 144 144 144 144 145 ...**

**$ permu.matrix.Q : num [1:1000, 1:6336] 4.06 3.91 3.56 4.4 3.79 ...**

**cluster.number** donates the number of QTLs in each cluster group. **cluster.id** donates the serial number of traits in each cluster group. **cluster.matrix** donates the EQF matrix from the clustering process. **permu.matrix.cluster** donates the permutation result of the clustering method which has been sorted by order. **permu.matrix.Q** donates the permutation result of the Q method which has been sorted by order. And the figure result of the plot through chromosomes is shown in Figure 5, the plots by each chromosome are shown in Figure 6.

* 1. The real data example

The real data example uses the yeast data (Brem and Kruglyak 2005). We provide the data which has been calculated the LOD score contains 5740 traits and 6336 bins. It can be loaded in to the R environment through the command

***> load("QTLhotspot.RDATA")***

***> yeastLODdata[1:5,1:5]***

**1\_0 1\_0.01 1\_0.02 1\_0.03 1\_0.04**

**[1,] 0.04707874 0.11560524 0.208530848 0.312990065 0.342238826**

**[2,] 0.09527598 0.17633589 0.273689630 0.375525717 0.360925870**

**[3,] 0.79761935 0.66959528 0.532599457 0.394223935 0.342336239**

**[4,] 0.36368204 0.32132689 0.272208652 0.218861000 0.192263065**

**[5,] 0.01665361 0.01048611 0.005469565 0.002032152 0.001140413**

***> yeastchromosome[1:5,]***

**chromosome # of bins**

**[1,] 1 256**

**[2,] 2 324**

**[3,] 3 160**

**[4,] 4 723**

**[5,] 5 334**

The process of detection can be run through the command

***R>*** ***LOD.QTLdetect.result =LOD.QTLdetect(yeastLODdata, yeastchromosome, thre=3,***

***+ gape=20)***

***R> permutation=EQF.permu(LOD.QTLdetect.result, ptime=1000, alpha=0.05, Q=TRUE,***

***+ plot.all=TRUE, plot.cr=TRUE)***

and the figure result are shown below.

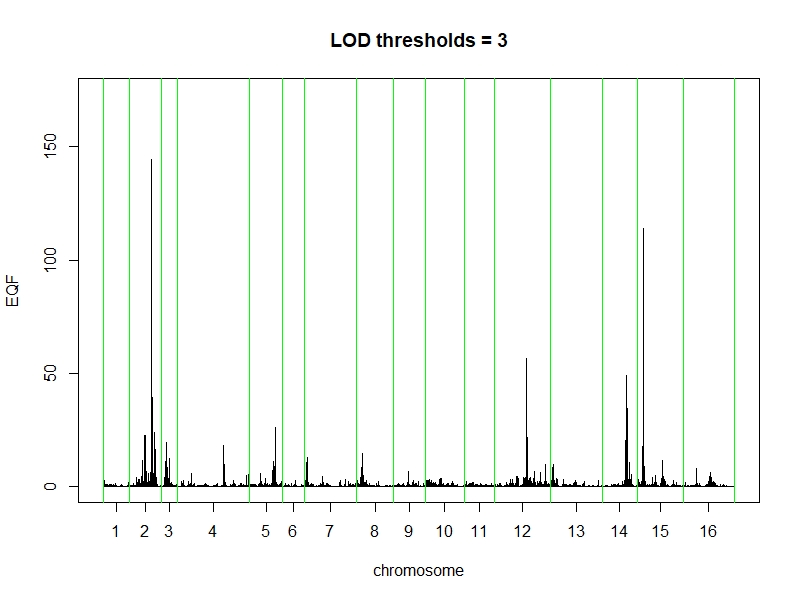


Figure 4: The figure result from **LOD.QTLdetect()**.

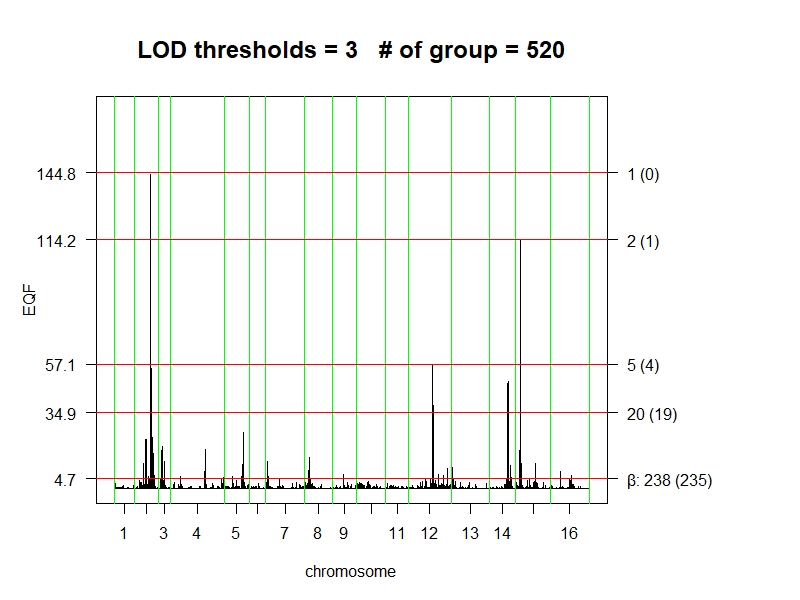


Figure 5: The figure result of the plot through chromosomes from **LOD.QTLdetect()**.

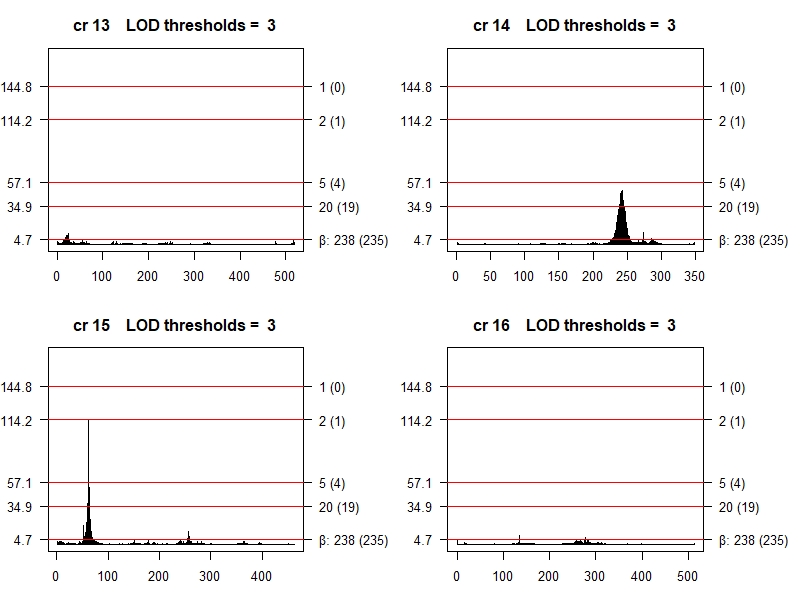
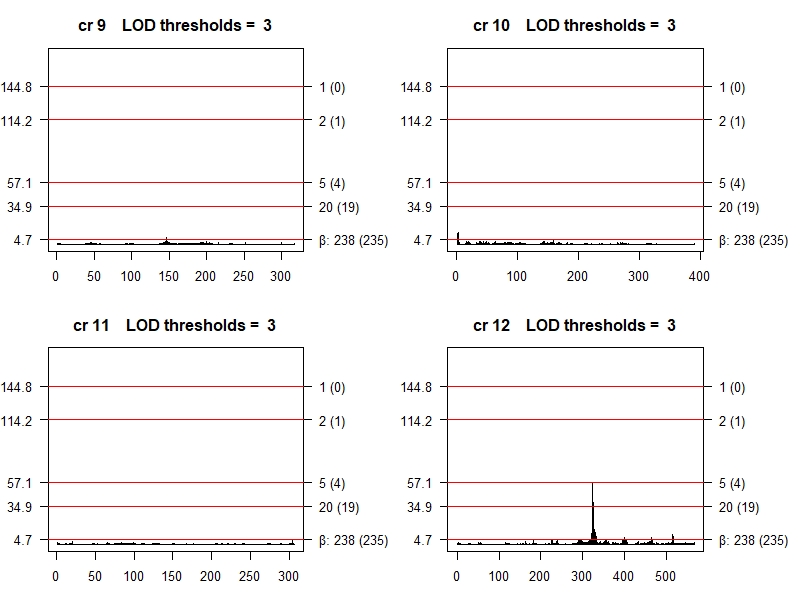
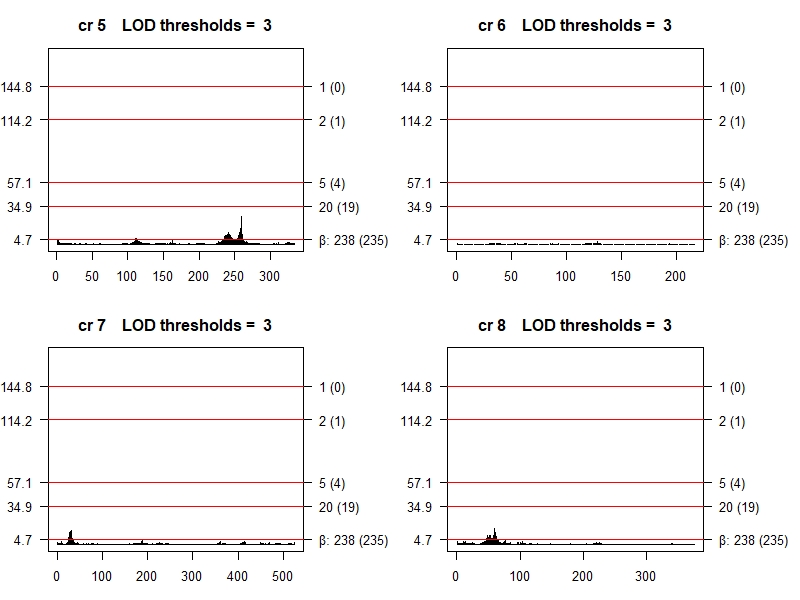
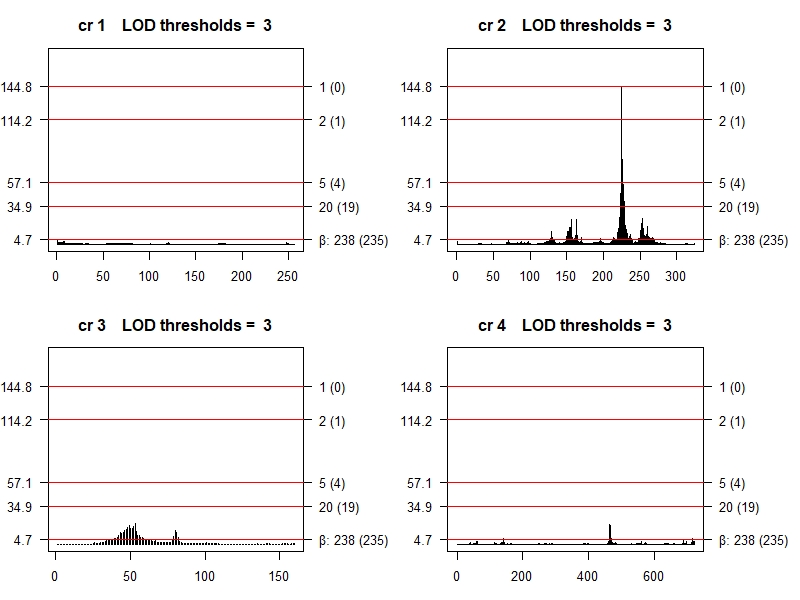


Figure 6: The figure result of the plots by each chromosome from **LOD.QTLdetect()**.