A Flexible Computational Framework Using R and Map-Reduce for Permutation Tests of Massive Genetic Analysis of Complex Traits

Behrang Mahjani, Salman Toor, Carl Nettelblad, and Sverker Holmgren

Abstract—In quantitative trait locus (QTL) mapping significance of putative QTL is often determined using permutation testing. The computational needs to calculate the significance level are immense, 10^4 up to 10^8 or even more permutations can be needed. We have previously introduced the PruneDIRECT algorithm for multiple QTL scan with epistatic interactions. This algorithm has specific strengths for permutation testing. Here, we present a flexible, parallel computing framework for identifying multiple interacting QTL using the PruneDIRECT algorithm which uses the map-reduce model as implemented in Hadoop. The framework is implemented in R, a widely used software tool among geneticists. This enables users to rearrange algorithmic steps to adapt genetic models, search algorithms, and parallelization steps to their needs in a flexible way. Our work underlines the maturity of accessing distributed parallel computing for computationally demanding bioinformatics applications through building workflows within existing scientific environments. We investigate the PruneDIRECT algorithm, comparing its performance to exhaustive search and DIRECT algorithm using our framework on a public cloud resource. We find that PruneDIRECT is vastly superior for permutation testing, and perform 2×10^5 permutations for a 2D QTL problem in 15 hours, using 100 cloud processes. We show that our framework scales out almost linearly for a 3D QTL search.

Index Terms—QTL mapping, RHadoop, PruneDIRECT, DIRECT, map-reduce, permutation testing

1 Introduction

Understanding the relation between genes and traits is a fundamental problem in genetics. Apart from basic understanding of genetics, such knowledge can eventually lead to, e.g., the identification of possible drug targets, treatment of heritable diseases, and efficient designs for plant and animal breeding. The aim of quantitative trait loci (QTL) mapping is to locate regions in the genome that can be associated to quantitative traits, i.e., traits where the individuals in a population exhibit continuous distributions. Indeed, most traits of medical or economical importance are quantitative, and such traits are normally affected by both genetic and environmental factors as well as their interactions. With accurate genetic and trait data, a powerful statistical model, efficient numerical algorithms, and a suitable environment for computational experiments, it is then possible to identify QTL and the corresponding statistical significance levels. For a review of QTL analysis, see, e.g., [6].

- *B. Mahjani and S. Holmgren are with the Division of Scientific Computing, Department of Information Technology, Uppsala University, Uppsala, Sweden. E-mail: {behrang.mahjani, sverker.holmgren}@it.uu.se.*
- *S. Toor is with the Division of Scientific Computing, Department of Information Technology, Uppsala University, Uppsala, Sweden, and the Helsinki Institute of Physics, University of Helsinki, Helsinki, Finland. E-mail: salman.toor@it.uu.se.*
- *C. Nettelblad is with the Science for Life Laboratory, Division of Scientific Computing, Department of Information Technology, Uppsala University, Uppsala, Sweden. E-mail: carl.nettelblad@it.uu.se.*

Manuscript received 16 Jan. 2015; revised 16 Nov. 2015; accepted 19 Jan. 2016. Date of publication 11 Feb. 2016; date of current version 22 Mar. 2017. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TCBB.2016.2527639

It is very rare that a quantitative trait is controlled by a single genetic locus. Rather, the reason for a trait being quantitative is often the property that multiple genes are involved [2]. Each QTL can affect a trait independently (additive) or in interaction (epistasis) with other QTL. In both cases, a multidimensional statistical model is needed to detect and assess the effects in an accurate way [34]. Mathematically, the search for the genetic positions of a putative QTL corresponds to solving a multidimensional global optimization problem where the objective function is the statistical model fit.

Historically, most of practical QTL analysis has been focused on locating a single QTL at a time. Recently, simultaneous mapping of two QTL has become more widely used, but such analyses are still often focused on identifying additive effects only. Accurate mapping of multiple QTL with epistatic interactions in larger networks than pairs is not yet established as a standard technique. However, based on the progress of genetics research there has been increasing interest in making tools for this type of analysis available, see, e.g., [15]. Standard tools for QTL analysis, using standard computational algorithms, are not able to cope with the massive computations needed. A main reason for this is that, after identifying a set of QTL locations, the corresponding statistical significance levels needs to be determined using permutation tests [21]. A high significance level corresponds to solving a large number of permuted problems, where each instance is equivalent to locating a set of putative QTL for a new set of permuted phenotypes. Accurate permutation tests for larger models with multiple loci cannot be performed using current standard algorithms and software. Significant development, both in terms of

algorithms and software, is required. In this paper, we take an important step in this direction by presenting a framework for analysis of multiple QTL based on a highly efficient search algorithm used both for finding a putative set of QTL and establishing statistical confidence levels using permutation testing. Our QTL analysis workflow is implemented by integrating different software building blocks into a standard R program, resulting in an environment for computational QTL analysis that feels familiar to many geneticists. From the R program, cloud computational resources are accessed using RHadoop [36], transparently performing the demanding computations in parallel. The power of R compared to more specialized environments is the ability for the end user to freely adapt and extend the methods and workflows. This approach is similar to a pilot implementation presented in [9] for accessing grid, cluster and HPC resources. However, using modern software and services, we now show that this type of tool can be easily implemented in a portable and easy-to-use way, allowing even inexperienced users of larger-scale computational resources to extend and modify their QTL analysis toolbox. We claim that this will enable new types of QTL analysis, potentially leading to new results in genetics. In summary, the key contributions in this paper are: first, analysing PruneDIRECT QTL scan for permutation testing more in details [5]; second, developing a parallel computing application in R to calculate the statistical significance of a QTL point using permutation testing.

In Section 2 we introduce the statistical model used in QTL mapping. In Section 3, we explain how many permutations are needed to have a reasonable statistical significance level. In Section 4, we review the QTL search algorithms DIRECT and PruneDIRECT, and analysing PruneDIRECT for permutation testing more in details. In Section 5, our new framework for parallel permutations testing is introduced. Sections 6 and 7 describe the computational experiments and the related results. Section 8 gives an overview of a few of the related works, highlighting the importance of our work.

The RHadoop configurations used in this article and the R code for PruneDIRECT can be obtained from:

https://github.com/PruneDIRECT/RVersion

2 QTL Analysis

A widely used statistical model used in QTL mapping is based on linear regression [7]. This is also the type of model used in this paper. However, note that the search algorithms and software can be adapted to other types of statistical models. A brief summary of the linear regression model is given below.

A set of d locations in the genome can be identified by the vector $\mathbf{x} = [x_1 \quad x_2 \dots x_d]$, where $x_i \in [0, G]$, G being the size of the genome (normally measured in centi-Morgans [cM]). Let n be the number of individuals in the population, $\mathbf{y}$ the vector of n phenotype observations, $k \geq d + 1$ the total number of parameters in the model, and $\mathbf{b}$ the vector of k effects. A generic linear regression QTL model can then be formulated in matrix form,

$$\mathbf{y} = A(\mathbf{x})\mathbf{b} + \epsilon, \tag{1}$$

where the $n \times k$-matrix $A(\mathbf{x})$ is the design matrix and ϵ is the error vector. In the QTL mapping procedure, this model is evaluated for different combinations of locations $\mathbf{x}$ in search of the best model fit. This corresponds to solving the optimization problem

$$RSS_{opt} = \min_{\mathbf{b},\mathbf{x}} (A(\mathbf{x})\mathbf{b} - \mathbf{y})^T (A(\mathbf{x})\mathbf{b} - \mathbf{y}). \tag{2}$$

The problem (2) can be separated into two parts. The first, inner problem is to evaluate the objective function for a given position (3), i.e., to compute the model fit. The second, outer problem is a global optimization over the complete genome (4):

$$RSS(\mathbf{x}) = \min_{\mathbf{b}} (A(\mathbf{x})\mathbf{b} - \mathbf{y})^T (A(\mathbf{x})\mathbf{b} - \mathbf{y}), \tag{3}$$

$$RSS_{opt} = \min_{\mathbf{x}} RSS(\mathbf{x}). \tag{4}$$

When searching for d QTL, the search space for the outer problem (4) is in principle a d-dimensional hypercube where the side is G. However, in practice, symmetries can be used to reduce the size of the search space somewhat.

From actual experiments, genetic information is only available at a set of marker locations, determined by the experimental setting used for analyzing the genetic samples. Today, dense genetic maps with cost-effective experimental methods are available for many species. However, the genetic map might not be dense enough to cover all regions along chromosomes. Also, experimental data might also be missing or inconclusive for some individuals in some places. For these reasons, it is necessary to solve a missing data problem to account for QTL in arbitrary positions.

3 Permutation Testing for QTL Applications

In order to ascertain the genome-wide significance for a QTL optimum it is common practice to permute the phenotype vector, keeping identical genotypes [6], [14]. The point of this procedure is to replicate the overall properties of the population, while removing all genetically correlated contributions to the phenotypes. Thus, the empirical null distribution of the objective function is determined. The accuracy of the significance level depends on the number of permutations. To get arrive at a significance level α with a Monte Carlo error limit of $< C$ (as a fraction of unity, i.e., $100C$ percent), N permutations are needed, where N is [21]:

$$N = \frac{1 - \alpha}{\alpha C^2}. \tag{5}$$

Here, C is the coefficient of variation which can be interpreted as the Monte Carlo error affecting the estimate, see, e.g., [27], pp. $208-210$.

To calculate a 95 percent significance level with error < 0.01 percent, we need $N = 1.9 \times 10^5$ permutations, and for 99.9 percent significance with error < 0.01 percent, we need around $N = 9.9 \times 10^5$ permutations. This obviously poses a great computational challenge for the use of permutation testing in practice.

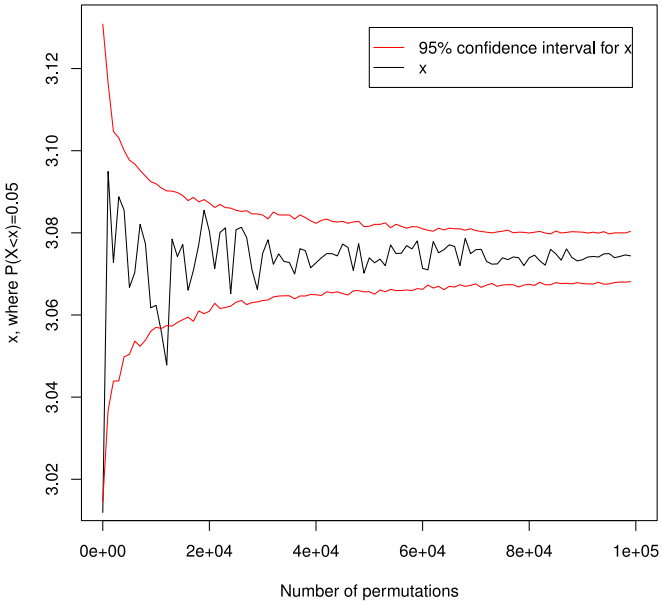

Fig. 1. The number of permutations needed to get an accurate estimate of the 95 percent significance level.

We calculated the empirical distribution of the objective function in PruneDIRECT for a backcross model of a 2D search. The details of the model and the data are explained in the section Computational Experiments. The objective function is a function of the residual sum of squares, which indicates the goodness of fit for the (permuted) phenotypes against the genotypes in a specific candidate position. As discussed above, $N = 1.9 \times 10^5$ permutations are needed for establishing a 95 percent significant threshold with an error for the threshold estimate < 0.01percent.

Fig. 1 illustrates the identified 95 percent significance threshold for different permutation counts. The red line is the 95 percent bootstrap confidence interval of size $1{,}000$. As the plot shows, the variance of the estimated significance threshold decreases very slowly. We know from equation 5 that the variance is proportional to $1/\sqrt{N}$.

4 Global Optimization Algorithms for QTL Search

Traditionally, solvers for the search problem (4) have been implemented as an exhaustive search over a d-dimensional hypercube. This is equivalent to a model fitting problem in a d-dimensional hypercube of genome positions. Since the evaluation of the statistical model fit is relatively demanding, the computational requirement for this optimization procedure can be very large, increasing exponentially with d. Furthermore, when using permutation testing to determine a significance level, the d-dimensional search must be repeated, e.g., 10^5 times for different permuted data sets. Using exhaustive search combined with permutation testing results in accurate and reliable QTL locations (within the framework of the statistical model used), but the computational demands are often considered to be prohibitive already for $d = 2$. For $d > 2$, truly multidimensional QTL mapping with accurate permutation testing has to our knowledge not been used in published genetics research.

Alternative search algorithms for QTL mapping have been studied, e.g., genetic algorithms [3] and DIRECT [4]. Using these algorithms, potential QTL locations can often be determined orders of magnitude faster than when using exhaustive search, but instead accuracy becomes a problem. For this type of black-box global optimization algorithms it is impossible to define a stopping criterion that combines rapid termination with a guarantee that the correct QTL locations are found. Specially, for the permutation tests the search space is by construction very unstructured. Since the search algorithm is used many times for different parameters, even a small error can bias the final result, see, e.g., [4].

In the experiments performed in this paper, we compare the permutation testing using standard exhaustive search to the version of DIRECT presented in [4] and to a novel search scheme named PruneDIRECT [5] using our new parallel framework for QTL permutations.

4.1 DIRECT

The DIviding RECTangles (DIRECT) optimization scheme was first suggested in [10]. The algorithm could be seen as a further developed branch-and-bound strategy, where the difference is that while a traditional scheme of this type requires a known bound on the variation within any subregion, DIRECT does not. Rather, it explores the search space opportunistically. This is done by splitting the space into successively smaller boxes, starting out from a single box covering the entire search space.

Within each iteration of the DIRECT method, a subset of boxes to split is identified. This subset is defined as a *convex hull* of the existing boxes in a radius-objective function space. Thus, a box is split if it is larger or has a smaller objective function value. For two boxes with identical objective function values in their respective centroids, the larger one is preferable, since the minimum value possibly existing within that box is smaller, given any assumption on the maximum variation. For boxes of the same size, a smaller objective function value in the centroid is naturally preferable. In effect, the slopes between the boxes in the convex hull will make an assumption on the value of the Lipschitz constant K. As the method proceeds, this assumption will get less and less strict. In Fig. 2, an example of the search space for DIRECT is shown.

Asymptotically, K will tend towards infinity and all points in the search space will be evaluated. If a minimum size limit is posed for the DIRECT search boxes, eventually DIRECT turns into a method which is equivalent to an exhaustive search on the corresponding grid. However, the minimum will be found much earlier for any reasonably smooth objective function, even if an analytical Lipschitz K might be hard to identify. The process of genetic recombination and the family structures used in QTL experiments should ensure a high level of correlation in function values locally, supporting the argument for why DIRECT should quickly find the optimum for QTL search problems. However, a critical issue is to know when to stop DIRECT and accept the current minimum as the answer. Several termination criteria have been tested in the literature, including performing a fixed number of iterations, detecting a slowing rate of change in the minimum, or stopping after a certain number of iterations with no further change in the

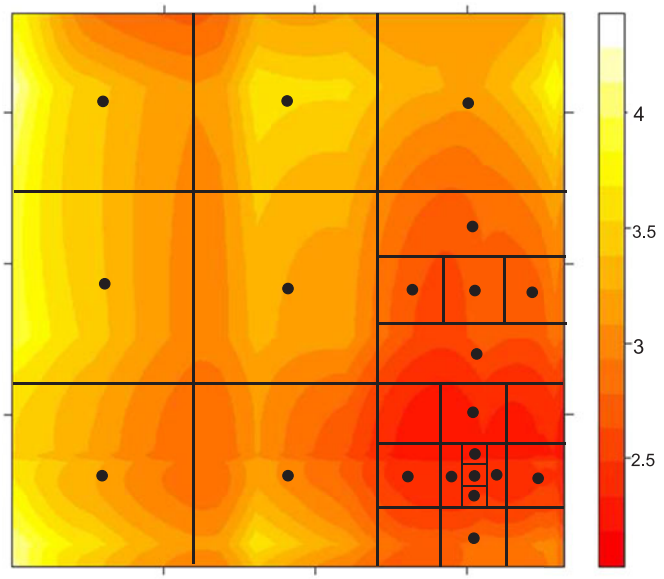

Fig. 2. An example of a search space for DIRECT after a few iterations (splittings). The lines represent search boxes and the black dots indicate where the objective function is evaluated. The background colors show the search landscape.

minimum. For the QTL search problem, a modification of the latter was proposed [8], with a scaling imposed in terms of the number of search dimensions.

4.2 PruneDIRECT

Recently, we presented the PruneDIRECT algorithm, which can be seen as a reconciliation between DIRECT and more general branch and bound strategies in the context of the QTL search problem [5]. PruneDIRECT is based on carefully choosing the objective function used. While many distinct objective function formulations can be topologically identical in the relative ordering of points (including the location of the global minimum), they are not all identical in terms of their local rate of variation. For DIRECT to be maximally efficient, a Lipschitz bound with a small constant K should exist. Second and higher order derivatives should be small, except for the immediate region around an extreme point (where the derivative switches sign).

An analysis of the objective function for the QTL linear regression shows that neither the residual sum of squares, nor the logarithm of the residual variance, are ideal objective functions for DIRECT, but the logarithm of the explained variance, called *LogVar*, has very attractive properties. Using this objective function, we in [5] derived a Lipschitz bound with $K = 0.04$ for an ideal infinite-size population. However, no population is truly ideal. Even a population arising from experimental conditions perfectly replicating the models will deviate from this result for K, due to sampling effects. Therefore, it turns out that directly applying a Lipschitz-bound based optimization scheme with $K = 0.04$ is not possible. In [5] we also derived a stochastic bound for actual finite-size populations which can be used in a practical algorithm. This is done by calculating the overall probability of recombination, as well as the sampling effects due to the specific phenotypes of those individuals that do recombine. Given an existing or desired value of f_{min}, the current global minimum to beat, a hypothesis test can then be posed for any box in the DIRECT algorithm: What is the probability of finding an objective value equivalent to that of the centroid of this box, given that there actually is another point within the box with an objective function value equivalent to f_{min}? If that probability value is exceedingly small (small enough to account for the multiple hypothesis tests, one for each box), the conclusion is instead that the other part of the hypothesis is wrong – there is no such point with a function value equivalent to f_{min} within the box. Due to monotonicity in the function, it is enough to test for f_{min} at the boundary to account for the presence of any value below f_{min} anywhere within the box.

The box pruning test can easily be integrated into DIRECT when boxes are added and when the convex hull is extracted. The main prioritization effect of DIRECT, giving preference to lower values of K, is still important in order to quickly locate interesting regions for the search and determining the proper f_{min}. The presence of a strict upper bound on the variation in each box allows *permanent* exclusion, or pruning, of some boxes in the continued search. Large boxes far away from any QTL signal can be pruned early, avoiding splitting. With permanent exclusion of boxes and a limit on the smallest box size allowed, termination comes natural when there are no boxes left to evaluate. This will happen before the search has degenerated into an exhaustive search. The only exception is when the global minimum is close to the overall noise level. In that case, it is actually necessary to test every single position, since the local variation is on the same order as the global optimum.

4.3 Permutation Testing in PruneDIRECT

According to the reasoning above, the most expensive PruneDIRECT search possible is trying to find the global minimum in complete noise. That is the exact setting of the permutation experiment, which seems to indicate that the PruneDIRECT scheme is not useful. However, by performing the search for the QTL signals of interest before the permutation testing, the searches for the permuted data can be performed with a pre-seeded f_{min}. Then, most of the permuted searches will terminate very early. In order to find the significance of our signals of interest, we only need to know in what proportion of tests the permuted datasets provide superior optima. Therefore, pruning can be done against the f_{min} of the true signal. The stronger the original QTL signal is, the earlier the permuted searches will terminate. Thus, the permuted searches are quicker for those signals where the strongest significance levels (the most permutations) would be relevant to test for. Note that neither the standard exhaustive search nor the original DIRECT algorithm allows for this type of integration of the search for the QTL and the permutation test.

4.4 Fixed Mesh

Computing the probabilities of inherited alleles for each individual is not trivial. The most efficient current methods rely on hidden Markov models [11], [12], [13]. These can easily compute the probabilities for any set of positions. However, they do so using data structures with a possibly quite significant memory consumption. Therefore, existing QTL analysis tools tend to loop over all individuals,

computing all probabilities for one individual before going to the next one. In this setting, it is not possible to "query" the method for the probabilities for all individuals at a single position. Due to the nature of the experimental data, it is also not necessary, or relevant, to refine the exact location of the optimum too far. For exact elucidation of the locus, other experimental methods are needed, like local gene sequencing.

Therefore, we adapted DIRECT to make sure that box centroids and side lengths coincide with the mesh probabilities. If this was not done, the Lipschitz assumption would not hold from the perspective of the algorithm, since there would be discontinuities at probability mesh borders. By aligning the probability mesh and the DIRECT box mesh, all such discontinuities are contained within the minimum box length.

5 The Framework for QTL Analysis

Several earlier efforts have been made to deal with the large computing demands of multidimensional QTL analysis, both with regards to efficient algorithms and different computational models. Classical parallel computing strategies like MPI, OpenMP and grid computing have been used to enable pilot experiments for high-dimensional QTL searches [9], [16], [17]. For these early efforts, a significant level of expertise and experience of parallel computing was required to develop, install, and run the QTL search application. This is often lacking in the genetics community.

We use cloud computing technologies and ideas in our new framework. We believe that the use and installation of our new QTL search framework would be within reach for many geneticists. Moreover, by building the framework by integrating a set of pre-existing components, we claim that all of the framework should be accessible and maintainable by a wider community. This facilitates further development and adoption both to new problem settings and computational infrastructures.

One of the main reasons that we implemented our framework in the statistical environment R is that R is a familiar environment for geneticists. Also, having the main framework in R using different functions and components gives the users the possibility to rearrange the functions and components and adopt the statistical model, the search algorithm, as well as the parallelization steps to their needs. While the main steps of the algorithm should be in R, the details of the functions can be implemented in C and be called from R, if they are computationally expensive. In other words, we are trying to strike a balance between easy adaptability and performance.

A large number of packages has been made available for R by other software developers, including packages for QTL analysis. These packages can easily be used for e.g., pre- and post-processing together with our new search tools. We explain later how one can benefit from these packages in our framework

The high level overview of our framework is illustrated in Fig. 3. The major contribution of this article is block 4, distributed computing of the significance level. We explain each block in details, and how one can modify them to adapt it to different QTL searches.

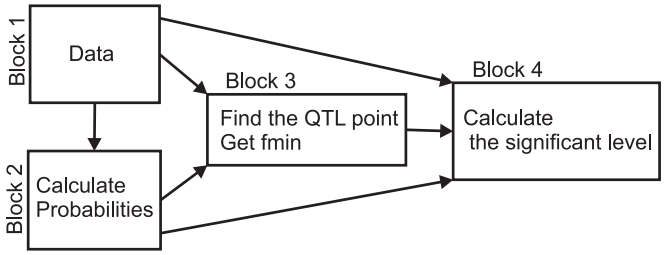

Fig. 3. The high level overview of our framework, from data to calculating the QTL, to calculating the significance level. The main contribution of this article is in block 4.

5.1 Initialization

Block 1 and 2 are the initialization steps for a QTL search. In block 1 we prepare the data and in block 2 we calculate the genotype probabilities needed in the statistical model for the QTL search. In our framework we use the R/qtl [11] package for generating the genotype probabilities. These could alternatively be provided from the more generic cnF2freq codebase [12] using the R bridge outlined in [18].

These two block are also needed for permutation testing in block 4.

5.2 Search Schemes

The PruneDIRECT optimization scheme was implemented on top of DIRECT by modifying the extraction of convex hulls. The cumulative density function for residual-sum-of-squares values was computed as outlined in [5], however the efficient implementation of the binomial probability density function suggested in [5] was not used. This omission simplifies the code, but it also makes the overhead from PruneDIRECT evaluations non-trivial. For each box radius, a binary search was performed to find the right bound level for a 10^{-6} level of significance. These bounds were recomputed each time f_{min} changed. Candidate boxes in convex hulls that satisfied the pruning conditions were permanently removed from the data structure.

The objective function was implemented as a standard linear regression based on a QTL effect design matrix allowing interaction. The exact objective function formulation was that of normalized LogVar as described in [5], i.e., the logarithm of the explainable variance divided by the total phenotype variance. The maximum evaluation depth was limited by using a minimal box size. The mesh for genotype probabilities was chosen to match with a power of three of the chromosome lengths, in order to make sure that DIRECT box split boundaries coincide with probability mesh boundaries. The LogVar objective function was used for all three scenarios of exhaustive search, DIRECT and PruneDIRECT.

Our previous work with DIRECT has been based on a C codebase with some fragments in C++. This code was extended into a pilot implementation of PruneDIRECT in [5]. Using C results in high performance and easy enabling of threading parallelism, but it also makes the software harder to understand and use for many geneticists. In many cases, users might be interested in modifying an algorithm to adapt it to their own specific needs. We therefore re-implemented the code needed for performing QTL searches using DIRECT and PruneDIRECT in R. Here, the implementation is based on the general DIRECT implementation in

the R package bootfs [35]. The DIRECT code in that package is in turn a re-implementation in R of a Matlab code developed in [20].

Block 3 in Fig. 3 contains a QTL search algorithm, and it has the following components:

1. Get the data from block 1 and the probabilities from block 2.
2. Define the objective function which is the statistical fitness used to find the QTL point.
3. Define the function to calculate the PruneDIRECT bound, RSS_x. For intercross and backcross we defined the bound in [5].
4. Use PruneDIRECT.

Users can modify each of the above components to adapt the search algorithm to their problem. Changes in component 4 is for advanced users which they want to modify the core elements of PruneDIRECT to develop a new algorithm. As an example, one might try to change the way that PruneDIRECT is handling symmetric spaces.

5.2.1 Distributed Computation of the Significance Level

In this section we explain the details of block 4, which is the main contribution of this article. In this block, we do a permutation testing to calculate a significance level for the found QTL point. Our goal is to do N permutations using m mappers in parallel using Hadoop, meaning that each mapper should do N/m permutations. We use the map-reduce programming framework in Hadoop to distribute these task among the virtual CPUs. First we briefly introduce what is Hadoop and cloud computing, then we describe the details of block 4.

5.2.2 Hadoop and RHadoop

Apache Hadoop provides a scalable and fault-tolerant distributed computational platform for large-scale applications, used in both industrial and research-based settings. The main strengths lies in easy parallel execution of independent tasks based on a **map-reduce** approach as well as high availability of large datasets using the Hadoop Distributed File System (HDFS). Together with HDFS, Hadoop enables an elementary model where only the two functions Map and Reduce need to be implemented to establish a many-task computing environment. Even if the usage of Hadoop is often associated with analysis of massive data sets, this platform can be used also for parallel execution of a large number of independent tasks, even if the data sets involved are not extremely large. At this stage, our focus is to fulfill the computational demands of the QTL search algorithm, but at a later stage the data-handling capabilities of Hadoop will be of use for dealing with the larger data sets connected to high-dimensional searches.

We use RHadoop to implement the permutation testing, allowing us to employ Apache Hadoop [37] from within the R environment. The RHadoop packages provide a distributed map-reduce programming model, which is substantially easier to use for non-expert programmers than would be the case if Hadoop-based programs would be written in, e.g., Java, as is common practice. Currently, four separate RHadoop packages are available:

- rmr: Contains the main set of APIs to write map-reduce programs in R.
- rhdfs: Provides an interface to the Hadoop Distributed File System (HDFS).
- rhbase: Used to access structured data within the Hadoop environment, backed by HBase.
- plyrmr: Provides access to common data manipulation operations on very large datasets available in HDFS.

RHadoop may not result in the computationally most efficient use of map-reduce, but this approach for giving access to basic map-reduce functionality from R significantly improves the ease of implementation and the usability for our QTL search framework. The level of abstraction provided by RHadoop can encourage biologists and statisticians to adopt modern programming paradigms in all parts of their workflows, rather than continuing with the monolithic single node, single thread approach still typical of R package development and use.

5.2.3 Using Cloud Resources

The rapid growth of computational resources offered by cloud providers provide a radically different option to traditional practices of managing and gaining access to computational resources. Together with the concepts of on-demand resource availability, elasticity, and the variety of available platforms, cloud computing thus becomes appealing concept in many contexts. One can utilise the resources from private or public cloud providers like Amazon, HP, and Microsoft public clouds. This model of resource availability frees the end users from infrastructure management and thus can fully focus on the application's requirements.

5.3 Distributed Computation of Permutations

Block 4 is the actual implementation of permutation testing. For each permutation, one should first permute the phenotypes, i.e., y_i's. Then, with the new set of y_i's and the calculated fmin, one should run PruneDIRECT to check if it is possible to find a point that has lower minimum value than fmin. If PruneDIRECT finds such a point, then it should give 0 as output (otherwise it should give 1), meaning that the null hypothesis that we found the QTL point is rejected. The significance level is the sum of all 0's divided by total number of permutations.

As we have mentioned before, to parallelise a problem using Hadoop, one should identify what should be the Map function and what should be the Reduce function and also what are the Keys and Values. We have N permutations to parallelise over m mappers. Therefore, we need $n = N/m$ permutations for each mapper. Each virtual CPU runs an equal number of permutations with a set of pre-specified random number generator seeds. The controlled seeds ensured reproducibility of experiments. Each mapper returns a list of key-value pairs, where the key is 0 (reject the null hypothesis) or 1 (accept the null hypothesis), and its value is 1. Once all mappers have finished, the generated key-value pairs are passed to the reduce function for further processing. The reducer calculates the total number of 0's and 1's, which if we divide it by total number of permutations we get significance level. This map-reduce structure is illustrated in Fig. 4.

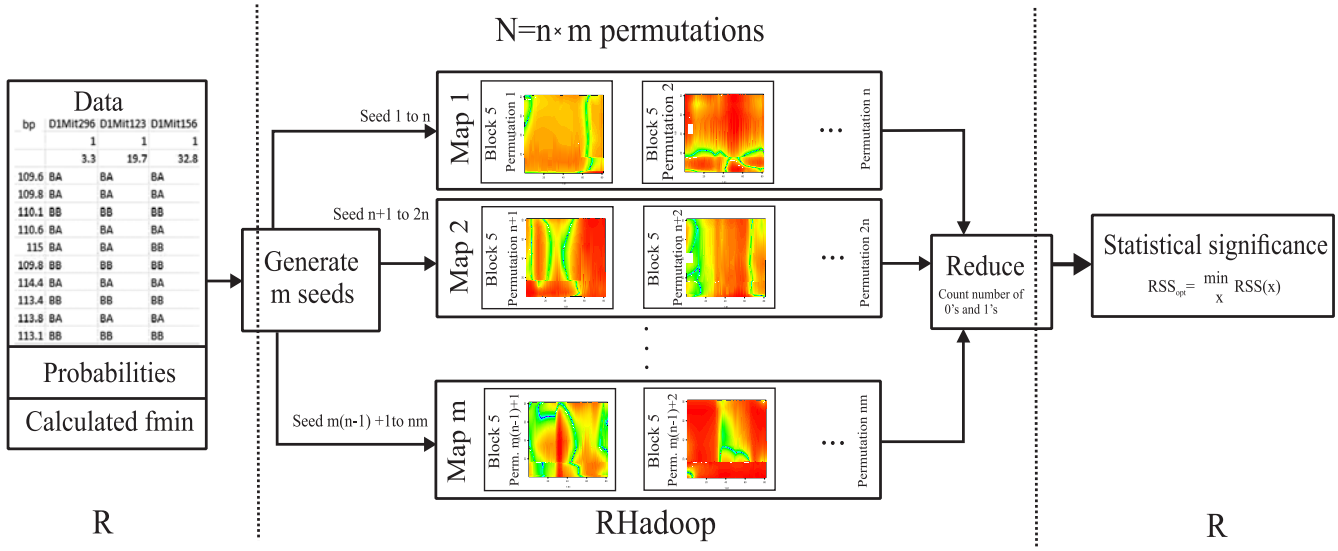

Fig. 4. Distributed computing of the significance level using map-reduce setting. Details of block 4.

6 Computational Experiments

We did two set of experiments to check our framework. The first experiment was done to compare Exhaustive search versus DIRECT versus PruneDIRECT. We performed the first experiment for a 2D QTL search since it is possible to illustrate the search graphically and analyze how PruneDIRECT is working. The second experiment was done for a 3D QTL search to verify how our framework scales out, and also illustrates the practical application of PruneDIRECT for 3D searches. We should emphasis that most of other softwares are based on exhaustive search and not able to perform a 3D search for high number of permutations. We show in the results section that it is impractical to use exhaustive search for multidimensional QTL scans because of the high computational cost.

The first experiment in this paper were run on public cloud resources provided by HP Sweden, via the service provider IT-Total. In practice, access to a base VM image was provided by IT-Total, and we augmented this with the additional components needed for our software stack. Snapshots of the same VM were used to spawn multiple instances for distributed computing. It would be rather straightforward to go from our proof-of-concept experiments to an online service for QTL analysis of complex traits.

The setup provided by HP Sweden consists of 25 Ubuntu 64-bit 12.04TLS instances, each mapped to a quad-core Westmere generation Intel Xeons node with 4 GB of RAM and 40 GB disk storage. The Ubuntu instances are equipped with R, RHadoop and the R/qtl package. We only directly used the rmr and rhdfs components of RHadoop. The specific hadoop version used was 1.2.1, the latest stable version when the environment was first configured. In total, the setup has the capacity of running 100 mappers in parallel (one per core). For HDFS, we deployed one Name-Node and three Secondary-Nodes in the setup.

For the PruneDIRECT permutation tests, experiments were done both providing the original unpermitted QTL f_{min} as a starting point, and without doing that, in order to demonstrate the possible gains from aggressive pruning of searches for irrelevant permutations when exploring significance thresholds.

The data used consisted of the "hyper" example dataset from R/qtl. This dataset contains backcross data on salt-induced hypertension of 250 mice. Specifically, a 2D search was performed using the chromosomes 6 and 15, modelling a situation where one QTL resides on each chromosome. The literature shows that a strong QTL signal is expected there [11]. We consider a uniform fixed mesh with 729 points which results in a 2D search space containing 531,441 points.

In our map-reduce configuration, 100 mappers were used, each running an equal number of permutations with pre-specified random number generator seeds. The controlled seeds ensured reproducibility of experiments. The map-reduce structure is illustrated in Fig 4.

For our second experiment, we have employed cloud resources from Swedish National Infrastructure for Computing (SNIC) [38]. The SNIC Cloud initiative provides IaaS over distributed resources. Its a national level cloud facility offering compute, network and storage as a service to Swedish research community. We used the same data from experiment one, but this time preforming a 3D QTL search with two-way interactions (Epistasis) on chromosome 4, 5 and 6.

The scaling of the permutation testing process in terms of number of mappers is thus dependent on the expected finish time for the single mapper that takes the longest time to execute its task. This highlights the overall drawbacks of large monolithic tasks with varying execution times. Some mappers completed all their work early, while others kept running. In order to resolve this problem, we create more mappers than the number of virtual CPUs. In this way, hadoop's built-in scheduler allocates new tasks to a virtual CPU when it is finished with its task. We performed 640 permutations over 1, 4, 8, 16, and 32 virtual CPUs to analyze the scalability of our framework.

We should mention that we did not use this method for experiment one to avoid extra complexity in explaining our framework. In the first experiment, i.e., 2D QTL

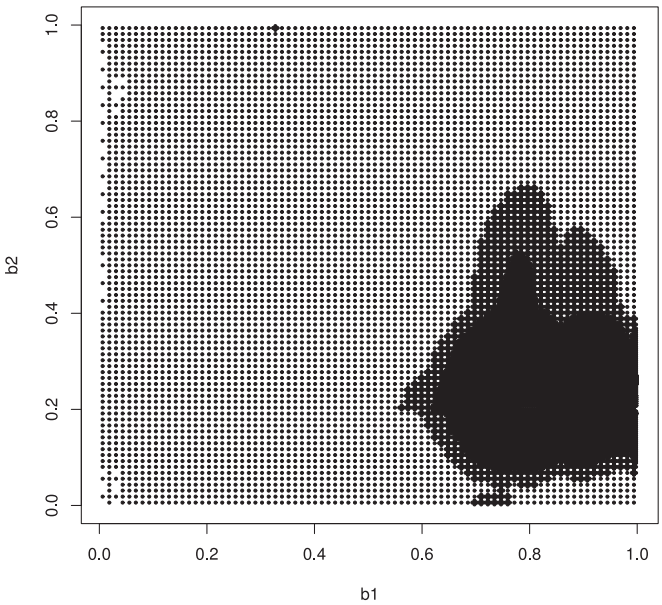

Fig. 5. The distribution of locations for function evaluations when DIRECT is terminated. The global minimum is located at $(0.99, 0.18)$.

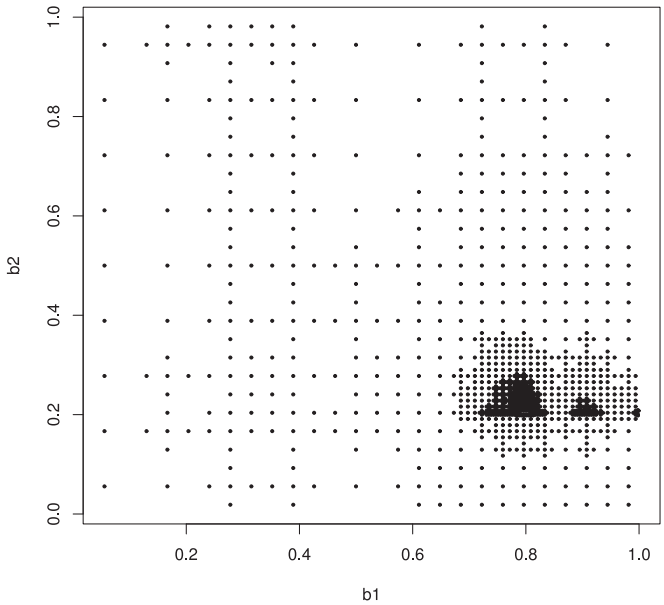

Fig. 6. The distribution of locations for function evaluations when PruneDIRECT is terminated. The global minimum is located at $(0.99, 0.18)$.

search, we used the same number of mappers as the numbers of virtual CPUs.

7 Results

7.1 Finding the True QTL

The basis for a QTL analysis experiment is to identify the QTL in non-permuted data. This in itself is a single QTL search, that might not be very computationally demanding. We compared exhaustive search, DIRECT and PruneDIRECT in this setting, with using the LogVar objective function. Exhaustive search traverses the complete space, thus doing 531,441 function evaluations. The performance for the DIRECT search depends on the stopping criterion. In our experiment DIRECT stops when the value of the found minimum is not changing, and there is no medium-sized box left to split. DIRECT used 19,136 function evaluations with this criterion, while PruneDIRECT stopped after only 4,454 function evaluations. Even in this case, it should be reiterated that the guarantees for correctness from using PruneDIRECT are much stronger compared to DIRECT, despite the lower number of evaluated positions. When PruneDIRECT stops, we can be sure with probability almost one to have found the exact solution (given the significance limit in our pruning bounds).

As illustrated by Figs. 5 and 6, the behavior of the two algorithms at termination is quite different. As DIRECT proceeds, it approaches a pattern similar to an exhaustive search, almost blindly evaluating additional points, while still not giving full guarantees for to what extent the most likely minimum has been explored. In contrast, PruneDIRECT stops after finding the global minimum, and checking the possibility of having another minimum in other parts of the space. The density of function evaluations close to the minimum is very similar in the two cases.

It is of interest to investigate how weak versus strong effect QTL affect the speed of PruneDIRECT. The QTL that we found above has an strong effect. In order to simulate a weak effect QTL, we added a normal distributed random noise to the phenotypes (with mean and variance larger than the mean and variance of the phenotypes). PruneDIRECT took $4,454$ function evaluations to find the strong effect QTL point (as we mentioned before), while it took $13,839$ function evaluations to find the weak effect QTL. It took longer time to find a QTL point for the weak effect, since the speed of PruneDIRECT depends on the structure of the search space. Generally speaking, if there is less structure in a search space, then it takes more time for PruneDIRECT to find the global minimum.

7.2 Permutation Test

As we mentioned before, we performed two experiments to analyze our framework. First experiment was to analyze PruneDIRECT for a high number of permutations for a 2D QTL search. Second experiment was to illustrates the practical application of PruneDIRECT for 3D searches, and also check the scalability of our framework.

7.2.1 First Experiment

Permutation testing was performed using RHadoop. In order to find a significance level of an interacting QTL point in settings such as this, one would need between 10^4 to 10^8 permutations, depending on the problem. Note that the search space for the permutation testing is generally significantly more unstructured, since there are many more almost equivalent (weak) local optima in most of these spaces. Using exhaustive search for permutation testing is very expensive as evidenced in Table 1. Already DIRECT provides significant savings in terms of function evaluations when performing 10^4 permutations.

As indicated in Section 4, there are two ways to use PruneDIRECT for permutation testing. If we use the calculated global minimum from finding the true QTL, the number of function evaluations required by PruneDIRECT decreases dramatically. A majority of runs terminate at the very beginning since the bound shows that the space given cannot

TABLE 1
10^4 Permutations, Prunedirect Is Using the Global Minimum from True QTL

	Exhaustive	DIRECT	PruneDIRECT
Total time (hours)	70	15	8
Average of FE.*	531,441	16,867	63
Standard division of FE.*	0	2,239	1,555

*FE. is the number of function evaluations.

TABLE 2
2×10^5 Permutations, Prunedirect Is Using the Global Minimum from True QTL

	PruneDIRECT
Total time (hours)	15
Average of FE.*	62
Standard division of FE.*	1,650

*FE. is the number of function evaluations.

contain a point smaller than the global minimum for the true QTL. If this information is not provided, PruneDIRECT needs more function evaluations, becoming more similar to unmodified DIRECT. In a smaller experiment, we checked this specifically, by running PruneDIRECT for 100 permutations. Using the global minimum for true QTL, PruneDIRECT used on average 20 function evaluation to find the minimum for this subset. Without the global minimum, 1,120 function evaluations were required, on average.

Table 1 shows results from runs where PruneDIRECT was instead initialized with the global minimum from the true QTL. Only 63 function evaluations were needed, on average, with a DIRECT average of 16,867. A very important result in this table is the standard deviation. The standard deviation for the function evaluation count is rather high. This means that while some searches finish very early, some take much longer. This is a main reason behind unbalanced workloads between different permutation subsets. The high standard deviation also explains how we could get a mean of only 20 evaluations for our smaller subset of 100 permutations, purely due to sampling effects. The median and mode of the distribution of the number of function evaluations per permutation are closer to this value.

7.2.2 Second Experiment

640 permutation was performed using RHadoop over 1, 4, 8, 16 and 32 virtual CPUs to analyze the scalability of our framework. The result is presented in Table 3. As the result shows, our framework scales out almost linearly as the number of virtual CPUs increases.

7.3 An Evaluation of the Presented Framework

The presented framework consists of two major sections, the first one enabling a high-level parallel execution mechanism, allowing users to utilize their familiar R statistical environment. Hadoop provides a restricted set of APIs that only allow execution in a map-reduce manner. However, this approach in practice allows serial applications that fit the model to be transformed into parallel ones with limited efforts. Map-reduce provides horizontal scaling of mappers to high counts of attached nodes. No change is required at the application level, if the overall problem can be readily separated into mapped tasks. Other alternatives would be to use, e.g., OpenMP or MPI. These require a higher degree of expertise and more care with regards to the system architecture, while also allowing a much higher degree of implementation flexibility in terms of shared resources during processing. The second part of the framework demonstrates the ease of utilizing cloud resources. The pay-as-you-go model together with pre-configured operating system images empower the scientists to launch the framework with minimal configuration and maintenance effort.

The efficient PruneDIRECT strategy together with the scalability features of Hadoop makes it possible to use PruneDIRECT for 2×10^5 2D QTL scan permutations within a reasonable timespan. As we can see in Table 2, PruneDIRECT can do 2×10^5 permutations in 18 hours while DIRECT does 10^4 permutation in 15 hours. A comparison of DIRECT and PruneDIRECT has been made previously, but this was largely a proof of concept requiring manual code modifications of a C codebase, and scripting of a cluster batch system to allow large-scale testing. Both of these aspects made that proof of concept ill suitable for a general scientific workflow.

We show that our framework scales our almost linearly as the number of virtual CPUs increases. We should highlight that virtual CPUs are used in cloud infrastructures. The consequence of using virtual CPUs is that one might run more than one job on the same CPU. This can affect execution time depending on how many jobs are running on a single CPU. At the time of this experiment, the full potential of the cloud was all the time in use, therefore, we believe that the error in the execution time for checking the scalability cannot be very large. We repeated the scalability experiment and got almost the same results.

8 Related Work

Permutation is a common method for establishing significance levels in genomic studies. However, the computational cost for each single model fitting in these studies is usually high, which restricts the ability to perform many permutations. Some authors have employed other sorts of permutation strategies, such as adaptive permutation [19], to decrease the raw count needed.

There are also many parallelization packages in R, such as "parallel" [22] and R-SPRINT [23]. Using these packages for massive permutation would demand larger administration efforts and pre-requisites regarding parallelization scheme knowledge on the user. R-SPRINT does not require the user to have a deep knowledge of parallelization. R-SPRINT is developed for high performance computing in R. One can use R-SPRINT for permutation-adjusted p-value calculations. However, R-SPRINT uses the MPI library in C for

TABLE 3
640 Permutations to Analyze the Scalability of the Proposed Framework

Number of virtual CPUs	1	4	8	16	32
Total time in hours	39.9	11.02	5.83	2.88	1.50

parallelization, while we use Hadoop. In our case, the permutation scheme is very amenable to the high-level, low-communication approach of Hadoop. The parallelization work itself is thus very straightforward. In fact, separate middleware and respond allocation systems such as SLURM have been developed to managed computational resources in MPI settings. Hadoop can also easily scale to very wide configurations on more general-purpose resources, with less effort compared to MPI-based schemes.

The viability of using the Hadoop framework has been explored in number of different research projects, including life science applications. The Biohoop [24] project focuses on the parallelization of three important tools using Hadoop framework. They have worked on the sequence alignment tool BLAST; the statistical analysis method for DNA microarray datasets GSEA; and GRAMMER, a fast method from QTL association analysis. The results presented in that publication demonstrate that the Hadoop computational framework is viable both for canonical (with large datasets) and non-standard (large computations with very little data) approaches.

Sudha et al., [25] presents a fast and accurate multiple alignment of protein sequences. The described approach is based on dynamic programming algorithm Needleman-Wnsch and its implementation in Hadoop framework. The execution environment requires different permutations of sequences to be generated and stored in HDFS. Each permutation than executes with the map-reduce phases in parallel. This project signifies the importance of Hadoop framework for running large number of permutations.

BlueSNP [26] presents an R package powered by the Hadoop framework for genome-wide association study analysis workloads. The approach of this project is related to ours in the sense that R is used to provide an environment which is familiar to many users. BlueSNP relies on RHIPE whereas we have used RHadoop.

Efforts have also been done to tackle parallel QTL searches and permutation testing specifically. Two widely used packages and services we would like to point out here are GridQTL [28], and recent releases of R/qtl [29]. GridQTL uses DIRECT with a somewhat problematic termination condition, executed on an academic grid HPC infrastructure. GridQTL has also recently been extended to cloud environment under the name CloudQTL [31]. In our framework, we are using PruneDIRECT algorithm which is, as we analyzed before, superior to DIRECT algorithm.

R/qtl uses the snow package [30], yet another approach for parallel processing in R. This package does not easily scale to very high node counts with the reliability and ease which are characteristic to Hadoop. Furthermore, much of the core logic in R/qtl is written in C, although the public interface is in R. R/qtl is therefore more of a fixed-function package, relative to the stated goal of our framework.

One of the major differences between our work and other works is PruneDIRECT can perform 3D QTL searches while neither CloudQTL nor R/qtl can perform a 3D QTL search.

9 Conclusion

RHadoop provides the user a high-level programming environment, allowing us to parallelize PruneDIRECT for permutation testing. Using this framework, we show that we can run 2×10^5 2D QTL scan permutations in less than a day. Based on our observations, a natural step would be to make the assignment of permutation tasks more dynamic, which is allowed by RHadoop. If one mapper engages in an expensive search, remaining permutations could be (re-) assigned to other mappers. The most straightforward way to do this would be to create more map tasks than there are mappers. However, there is a clear trade-off between balancing and overall performance in well-balanced cases, since the reduce coordination effort would be slightly increased.

If PruneDIRECT is to be used for only finding the significance level for one specific QTL solution, it is also possible to add another termination condition, allowing the search to terminate early in those cases where the currently recorded f_{min} of the current permutation search is superior to the global f_{min} for the true QTL. Some long-running searches may be due to an unproportional amount of time being spent on finding the specific value and location of the best f_{min} in such cases. The significance level found for our first experiment was 2×10^{-5}, meaning that in 2×10^5 hypothesis tests, the null hypothesis was rejected only two times.

In the example presented above, we are dealing with high computational costs rather than truly large or distributed data sets. A valuable extension of our framework is that one can easily extend this framework to work with big data sets, since Hadoop is designed to handle analysis of large data sets. This might be useful while working with, e.g., expression QTL data, where there might be tens of thousands of unique phenotype vectors, rather than a single one.

One of the benefits of using our framework is that one can use exactly the same framework in any cloud infrastructure such as amazon clouds. Also, this framework can be easily re-implemented in other Map-Reduce programs such as Spark or Snowdoop package in R.

In summary, we have presented a new framework for analysis of multiple QTL. Here, all experiments, settings and code modifications are done within the R environment, relying to a high extent on the standard library and common packages. This choice allows users with a limited programming knowledge to use the code and adapt it further to their own requirements. The framework provides transparent use of cloud resources for performing the demanding part of the computations, and it would be rather straightforward to develop our first implementation of the framework further into an online service for QTL analysis of complex traits. This framework can also be useful for general massive permutation testing, or model fitting problems.

As we mentioned before, the HRadoop configurations used in this article and the R code for PruneDIRECT can be obtained from our github repository. In near future we will release PruneDIRECT as an R package.

Acknowledgments

The computations presented in this paper have been performed using a public cloud provided by HP Sweden and hosted by IT-Total. Special appreciation is directed towards Max Samuelson at IT-Total for valuable help regarding the practical use of these resources. Carl Nettelblad was affiliated with the Department of Cell and Molecular Biology, Uppsala University, during most of this work.

REFERENCES

[1] E. S. Lander and D. Botstein, "Mapping mendelian factors underlying quantitative traits using RFLP linkage maps," *Genetics*, vol. 121, pp. 185–199, Jan. 1989.

[2] R. W. Doerge, "Mapping and analysis of quantitative trait loci in experimental populations, "*Nat. Rev. Genetics*, vol. 3, pp. 43–52, 2002.

[3] Ö. Carlborg, L. Andersson, and B. Kinghorn, "The use of a genetic algorithm for simultaneous mapping of multiple interacting quantitative trait loci,"*Genetics*, vol. 155, pp. 2003–2010, 2000.

[4] K. Ljungberg, S. Holmgren, and Ö. Carlborg, "Simultaneous search for multiple QTL using the global optimization algorithm DIRECT,"*Bioinf.*, vol. 20, pp. 1887–1895, 2004.

[5] C. Nettelblad, B. Mahjani, and S. Holmgren, "Fast and accurate detection of multiple quantitative trait loci,"*J. Comput. Biol.*, vol. 20, pp. 687–702, 2013.

[6] Z. Chen, *Statistical Methods for QTL Mapping*. London, U.K.: Chapman & Hall, 2013.

[7] C. S. Haley and S. A. Knott, "A simple regression method for mapping quantitative trait loci in line crosses using flanking markers," *Heredity*, vol. 69, pp. 315–324, 1992.

[8] K. Ljungberg, K. Mishchenko, and S. Holmgren, "Efficient algorithms for multi-dimensional global optimization in genetic mapping of complex traits,"*Adv. Appl. Bioinf. Chem.*, vol. 3, pp. 75–88, 2010.

[9] M. Jayawardena, C. Nettelblad, S. Toor, P.-O. Östberg, E. Elmroth, and S. Holmgren, "A grid-enabled problem solving environment for QTL analysis in R.," in *Proc. 2nd Int. Conf. Bioinf. Comput. Biol.*, 2010, pp. 202–209.

[10] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the Lipcshitz constant,"*J. Optimization Theory Appl.*, vol. 79, pp 157–181, 1993.

[11] K. W. Broman, H. Wu, S. Sen, and G. A. Churchill, "R/qtl: QTL mapping in experimental crosses,"*Bioinf.*, vol. 19, pp. 889–890, 2003.

[12] C. Nettelblad, S. Holmgren, L. Crooks, and Ö. Carlborg, "cnF2freq: Efficient determination of genotype and haplotype probabilities in outbred populations using Markov models," in *Proc. Int. Conf. Bioinf. Comput. Biol.*, 2009, pp. 307–319.

[13] C. Nettelblad, "Inferring haplotypes and parental genotypes in larger full sib-ships and other pedigrees with missing or erroneous genotype data,"*BMC Genetics*, vol. 13, no. 85, pp. 1–13, 2012.

[14] G. A. Churchill and R. W. Doerge, "Naive application of permutation testing leads to inflated type I error rates,"*Genetics*, vol. 178, no. 1, pp. 609–610, 2008.

[15] Ö. Carlborg and C. S. Haley, "Epistasis: Too often neglected in complex trait studies?" *Nat. Rev. Genetics*, vol. 5, no. 8, pp. 618–625, 2004.

[16] M. Jayawardena and S. Holmgren, "Grid-Enabling an efficient algorithm for demanding global optimization problems in genetic analysis," in *Proc. 3rd IEEE Int. Conf. e-Sci. Grid Comput.*, 2007, pp. 205–212.

[17] M. Jayawardena, H. Löf, and S. Holmgren, "Efficient optimization algorithms and implementations for genetic analysis of complex traits on a grid system with multicore nodes,"*in Proc. State Art Scientific Parallel Comput.*, 2008, pp. 1–10.

[18] R. M. Nelson, C. Nettelblad, M. E. Pettersson, X. Shen, L. Crooks, F. Besnier, J. Álvarez-Castro, L. Rönnegård, W. Ek, Z. Sheng, M. Kierczak, S. Holmgren, and Ö. Carlborg, "MAPfastR: QTL mapping in outbred line crosses,"*G3*, vol. 3, pp. 2147–2149, Oct. 11, 2013.

[19] R. Che, J. R. Jack, A. A. Motsinger-Reif, and C. C. Brown, "An adaptive permutation approach for genome-wide association study; evaluation and recommendations for use,"*BioData Mining*, vol. 7, p. 9, 2014.

[20] D. E. Finkel, "Global optimization with the DIRECT algorithm," Department of Mathematics, Ph.D. dissertation, North Carolina State Univ., Raleigh, NC, USA, Feb. 2005.

[21] R. W. Doerge, G. A. Churchill "Permutation tests for multiple loci affecting a quantitative character," *Genetics,* vol. 142, no. 1, pp. 285–294, Jan. 1996.

[22] G. Vera, R. C. Jansen and R. L. Suppi, "R/parallel speeding up bioinformatics analysis with R," *BMC Bioinf.*, vol. 9, p. 390, 2008.

[23] J. Hill, M. Hambley, T. Forster, M. Mewissen, T. M. Sloan, F. Scharinger, A. Trew, and P. Ghazal, "SPRINT: A new parallel framework for R," *BMC Bioinf.*, vol. 9, p. 558, 2008.

[24] S. Leo, F. Santoni and G. Zanetti, "Biodoop: Gioinformatics on hadoop," in *Proc. Int. Conf. Parallel Process. Workshops*, 2009, pp. 415–422.

[25] G. Sudha Sadasivam and G. Baktavatchalam, "A novel approach to multiple sequence alignment using hadoop data grids," *Int. J. Bioinf. Res. Appl.* vol. 6, no. 5, pp. 472–483, 2010

[26] H. Huang, S. Tata, and R. J. Prill, "BlueSNP: R package for highly scalable genome-wide association studies using Hadoop clusters," *Bioinformatics*, vol. 29, no. 1, pp. 135–136, Jan. 2013

[27] B. Efron, R. J. Tibshirani, *An Introduction to the Bootstrap*. Boca Raton, FL, USA: CRC Press, 1993.

[28] G. Seaton, J. Hernandez, J. A. Grunchec, I. White, J. Allen, D. J. D. Koning, W. Wei, D. Berry, C. Haley, S. Knott, "GridQTL: A grid portal for QTL mapping of compute intensive datasets," in *Proc. 8th World Congr. Genetics Appl. Livestock Production*, Belo Horizonte, MG, Brasil, Aug. 2006, pp. 213–218.

[29] D. Arends, P. Prins, R. C. Jansen, K.W. Broman, "R/qtl: High-throughput multiple QTL mapping," *Bioinf. Appl. Note*, vol. 26, no. 23, pp. 2990–2992, 2010.

[30] L. Tierney, A. J. Rossini, N. Li, "Snow: A parallel computing framework for the R system," *Int. J. Parallel Program.*, vol. 37, no. 1, pp 78–90, Feb. 2009.

[31] J. Allen, D. Scott, M. Illingworth, B. Dobrzelecki, D. Virdee, S. Thorn, S. Knott, "CloudQTL: Evolving a bioinformatics application to the cloud," Oxford, U.K., Digital Res., pp. 1–4, Sep. 2012.

[32] D. Eddelbuettel, R. Francois, "Rcpp: Seamless R and C++ Integration," *J. Stat. Softw.*, vol. 40, no. 8, pp. 1–18, 2011.

[33] D. Eddelbuettel, *Seamless R and C++ Integration with Rcpp*. New York, NY, USA: Springer, 2013.

[34] Ś. Sen, G. A. Churchill, "A statistical framework for quantitative trait mapping," *Genetics,* vol. 159, no. 1, pp. 371–387, Sep. 2001.

[35] C. Bender. (2013). Bootfs: Use multiple feature selection algorithms to derive robust feature sets for two or multiclass classification problems. R package version 1.4.2/r25 [Online]. Available: http://R-Forge.R-project.org/projects/bootfs/

[36] (2016). RHadoop. [Online]. Available: https://github.com/Revolution Analytics/RHadoop/wiki

[37] (2016). Apache Hadoop. [Online]. Available: http://hadoop.apache.org

[38] (2016). SNIC. [Online]. Available: http://snic.se

Behrang Mahjani is currently working toward the PhD degree in scientific computing with specialization in statistical computing at the Department of Information Technology, Division of Scientific Computing, Uppsala University.

Salman Toor received the master's and PhD degrees in scientific computing from the Uppsala University, Sweden. In 2014, he finished his postdoctoral research from CMS program, Helsinki Institute of Physics, Finland. His area of expertise includes management, scalability, and performance of distributed infrastructures for scientific applications. He is currently employed as a researcher in the Department of Information Technology, Uppsala University, and as a cloud application expert at UPPMAX HPC Centre. During 2015, he joined Swedish Infrastructure for Computing (SNIC) as a senior cloud architect.

Carl Nettelblad received the PhD degree in scientific computing. He is currently a junior lecturer with the Swedish National Science for Life Laboratory, focusing on computational methods for analysis of noisy data in biological applications ranging from diffraction imaging to genomics.

Sverker Holmgren is a professor of scientific computing at the Uppsala University, Sweden. He is currently head in the cross-disciplinary research program in computational science and the dean in mathematics and computer science. He is also the director in the Nordic eScience Globalisation Initiative at NordForsk.