iBMQ: An Integrated Hierarchical Bayesian Model for Multivariate eQTL Mapping

Greg Imholte, MariePier Scott-Boyer, Aurélie Labbe,
Christian F. Deschepper and Raphaël Gottardo

April 27, 2020

1 Download

`iBMQ` is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation. For citation purposes, please refer to the first reference in the bibliography.

To install this package, start R and enter:
if (!requireNamespace("BiocManager", quietly=TRUE)) install.packages("BiocManager")
BiocManager::install("iBMQ").

In order to properly compile iBMQ, the GNU Scientific Library (GSL) is required. GSL is free and can be downloaded at http://www.gnu.org/software/gsl/ . A tutorial on how to configure and use GSL and R can be found at http://wiki.rglab.org/ . For calculations with multiple processors (Mac or Linux only), OpenMP needs also to be installed (http://openmp.org/wp/).

2 Introduction

Recently, mapping studies of expression quantitative loci (eQTL), where expression levels of thousands of genes are viewed as quantitative traits, have been used to provide greater insight into the biology of gene regulation. Current data analyses and interpretation of eQTL studies involve the use of multiple methods and applications, the output of which is often fragmented. We present an integrated hierarchical Bayesian model that jointly models all genes and genomic markers (the most common ones being single nucleotide polymorphisms (SNPs)) to detect eQTLs. We propose a model (named iBMQ) that is specifically designed to handle a large number G of gene expressions, a large number S of regressors (genetic markers) and a small number n of individuals in what we call a large G, large S, small n paradigm [1]. This method incorporates genotypic and gene expression data into a single model while 1) specifically coping with the high dimensionality of eQTL data (large number of genes), 2) borrowing strength from all gene expression data for the mapping procedures, and 3) controlling the number of false positives to a desirable level. One distinct feature of our model is that we calculate one weight parameter for each Gene g and each SNP j.

3 Formatting the data

The next subsection describes how to format data. To run the eQTL analysis, two types of data are required: a genotype data frame and an expression data frame. To visualize the data, we also need a data frame with the genomic positions of each SNP and of each gene/probe.

3.1 The genotypes

Each genotype needs to be a SnpSet object from the Biobase package. When using data where each member of the population is homozygous at each position (such as for recombinant inbred strains (RIS): see example 1), the genotype must be coded with 0 and 1. When using data where each member may be homozygous or heterozygous (such as example 2) the genotypes must be coded 1, 2 or 3 and the parameter "RIS" of the eqtlMcmc function must be set as equal to FALSE. The column names must be the SNP names and the row names must be names of the individuals (or strains).

Example:

	SNP1	SNP2	SNP3	SNP4
Ind1	0	0	1	1
Ind2	0	1	0	1
Ind3	0	1	1	0

3.2 The gene expression values

Each gene expression value needs to be anExpressionSet object from the Biobase package. The assayData comprises the expression data for each member of the population. Each row and column corresponds to an individual and a gene, respectively. We have assumed that the gene expression values have been appropriately normalized and pre-processed using specialized statistical methods. The column names must be the gene or probe names and the row names must be the names of the individuals. The order of individuals needs to be the same as for the gene expression value needs to be the name as for the genotype data.

	Gene1	Gene2	Gene3
Ind1	10.2	11.4	12.2
Ind2	6.7	5.6	7.7
Ind3	13.1	14.5	12.3

3.3 The SNP position data frame

For the genome-wide eQTL mapping plot, we need a data frame specifying the genomic locations of each SNP with following columns: SNP name, chromosome number, SNP location (in base pair). Please note that the entries for the chromosome and the position need to be numerical values.

SNP	Chr	pos
Snp1	1	17098
Snp2	1	1029012

3.4 The gene position data frame

A data frame specifying the genomic locations of each gene/probe needs to be prepared with the following columns: gene name, chromosome number, start location (in base pairs) and the location (in base pairs). Please note that the entries for the chromosome and the position need to be numerical values.

Gene	Chr	start	end
Gene1	1	10290	10460
Gene2	1	18989	19069

4 First example dataset: data from mouse Recombinant Inbred Strains (RIS)

This example uses data generated by Williams and Lu, as available from the Gene Network website (genenetwork.com). This dataset comprises the profiles of mRNA abundance in whole eye tissue from n = 68 BXD RIS mice, as measured using Affymetrix M430 2.0 microarrays [2]. To ease calculation and facilitate comparisons, we will use a set of G = 1000 probes and 1700 single nucleotide polymorphic markers (SNPs).

5 Preparing the workspace

We will first load the library iBMQ, the SNP data and gene expression data.
To load the `iBMQ` package:

```
> library(iBMQ)
```

To load the SNP data:

```
> data(snp)
```

To load gene expression data:

```
> data(gene)
```

6 Running the eQTL model:

This function computes the MCMC algorithm to produce Posterior Probabilities of Association (PPA) for eQTL mapping. This function takes time: please be patient! On a Mac 2* 3.2 Ghz Quad-Core Intel Xeon computer using 6 cores, it takes 2.44 minutes.
If you have an appropriately powered computer, you can calculate the PPAs by running the function below. If not, you can skip the function and load pre-computed PPA table instead.

```
> #  PPA <-  eqtlMcmc(snp, gene, n.iter=100,burn.in=100,n.sweep=20,mc.cores=6, RIS=TRUE)
```

In this example, we will perform only 100 iterations in the interest of time. Please note that an optimal analysis requires a greater number of iterations (preferably in the order of 100,000, with a burn-in of 50,000). The result is a matrix with Posterior Probabilities of Association for each gene (row) and SNP (column).

7 Computing the FDR Threshold for eQTL identification:

Our ultimate goal is to identify gene/SNP associations, which can be done using parameter estimates from our model. An eQTL for gene g at SNP j is declared significant if it's corresponding marginal posterior probability of association (PPA) is greater than a given threshold. In the context of multiple testing and discoveries, a popular approach is to use a common threshold leading to a desired false discovery rate (FDR)[3]. In this example we will calculate a FDR of 0.1 (corresponding to 10%). The FDR value needs to be a numerical value between 0 and 1.
To calculate the threshold:

```
> cutoff <- calculateThreshold(PPA, 0.1)
```

In this example, the PPA optimal cutoff is 0.74. This means that all the eQTLs with a PPA above 0.74 are significant eQTLs. We can calculate how many eQTLs have PPA above the cutoff with the eqtlFinder function:

```
> eqtl <- eqtlFinder(PPA, cutoff)
```

With this example, a total of 759 significant eQTLs are identified. The output of the eqtlFinder is a data frame where the first column contains the names of each gene, the second column contains the names of corresponding markers and the third column contains the PPA value for each significant eQTL.

8 Classifying the eQTLs:

It is customary to distinguish two kinds of eQTLs: 1) cis-eQTLs (where the eQTL is on the same locus as the expressed gene); and 2) trans-eQTLs (where the eQTL is on a locus other than that of the expressed gene). The eqtlClassifier allows us to classify the eQTLs as either cis-eQTL or trans-eQTL according to their position in the genome. For this function, we need a data frame specifying the genomic locations of the SNPs and a data frame specifying the genomic locations of the gene/probe (see section on data format).
Load the data containing the position for each SNP:

```
> data(snppos)
```

Load the data containing the position for each gene/probe:

```
> data(genepos)
```

We need to specify a cutoff value(in base pair) corresponding to the threshold where a eQTL is considered a cis-eQTL. In this case, we will use a cutoff of 1 MB (1000000 pb).

```
> eqtltype <- eqtlClassifier(eqtl, snppos, genepos,1000000)
```

The output of the eqtlClassifier is a data frame where the first column contains the names of each gene, the second column contains the names of markers and the third column contains the PPA value for each significant eQTL. The fourth column contains the number of the chromosome to which the gene belongs, the fifth column contains the start position of the gene and the sixth column contains the end position of the gene. The seventh column contains the number of the chromosome to which the marker belongs, the eighth column contains the position of the marker and the ninth column contains a descriptor of the type of eQTL (either cis or trans). Please note that in order to ascertain that an eQTL is either cis or trans, the positions of the markers and the gene need to the given to the function. If one of the values is missing the type of eQTL will be "NA".

9 Visualizing the result:

To visualize the data we propose using the package ggplot2 to provides a genome-wide plot visualizing the characteristics of each eQTL in terms gene/snp position. Using the data frame from the output of the eqtlClassifier function and the provided code we can generate the plot .

```
> library(ggplot2)
> ggplot(eqtltype, aes(y=GeneStart, x=MarkerPosition)) +
+ geom_point(aes(y=GeneStart, x=MarkerPosition,color = PPA), size = 1.5)+
+ facet_grid(GeneChrm~MarkerChrm)+theme_bw(base_size = 12, base_family = "")+
+ theme(axis.ticks = element_blank(), axis.text.x = element_blank(), axis.text.y = element
```

Genome-wide distribution of eQTLs found by iBMQ for the test using 1000 probes for whole eye tissue from 68 BXD mouse recombinant inbred strains. The x-axis gives the position of each eQTL along the genome; the y-axis gives the position of the probe set target itself. The grey lines mark chromosome boundaries. The cis-QTLs are distributed along a diagonal line. Vertical bands represent groups of transcripts linked to one same trans-eQTL.

10 Finding eQTL hotspots:

One main advantage of our method is its increased sensitivity for finding trans-eQTL hotspots (corresponding to situations where a single SNP is linked to the expression of several genes across the genome). Two types of data frame can be used as input: 1) the data frame (3 columns) corresponding to the output of the eqtlFinder function; 2) the data frame (9 columns) corresponding to the output of the eqtlClassifier function.

```
> hotspot <- hotspotFinder(eqtltype, 10)
```

Using the dataset from example 1, we find 9 eQTL hotspots each containing more than 10 genes. The output of this function is a list, where each element is a marker. For each marker there is a data frame with all the eQTLs linked to this marker.

11 Example dataset 2: data from a mouse F2 cross

The next example uses data from a mouse F2 cross. In this case each genotype takes a value of either 1,2 or 3. This F2 cross data set containing genotypic and phenotypic information for 60 mice was obtained from the lab of Alan Attie at the University of Wisconsin-Madison. These data are also available at GEO (accession number GSE3330). Only the 5000 most variable expression traits out of 45,265 transcripts from the liver were used for the current example.
We will first load the genotype and gene expression data for the 60 mice.

```
> data(genotype.liver)

> data(phenotype.liver)
```

Since these functions take a lot of time, we have already calculated the results for 100,000 iterations and 50,000 burn-in. These numbers were suggested by simulation[1]. Please note that for the F2 data the "RIS" parameter needs to be set as equal to FALSE. If you have an appropriately powered computer, you can calculate the PPAs by running the function below. If not, you can skip the function and load pre-computed PPA table instead.

```
> #PPA.liver <-  eqtlMcmc(genotype.liver, phenotype.liver, n.iter=100,burn.in=100,n.sweep=

> data(PPA.liver)
```

This function will calculate the threshold to identify significant eQTLs.

```
> cutoff.liver <- calculateThreshold(PPA.liver, 0.2)
```

This function will find the all significant eQTLs.

```
> eqtl.liver <- eqtlFinder(PPA.liver, cutoff.liver)
```

We will load the position of the markers and probes to be able the classify each eQTL as either cis-eQTL or trans-eQTL. Since in this case, the genetics map is not very dense, we will use a 5 MB cutoff for detecting cis-eQTLs.

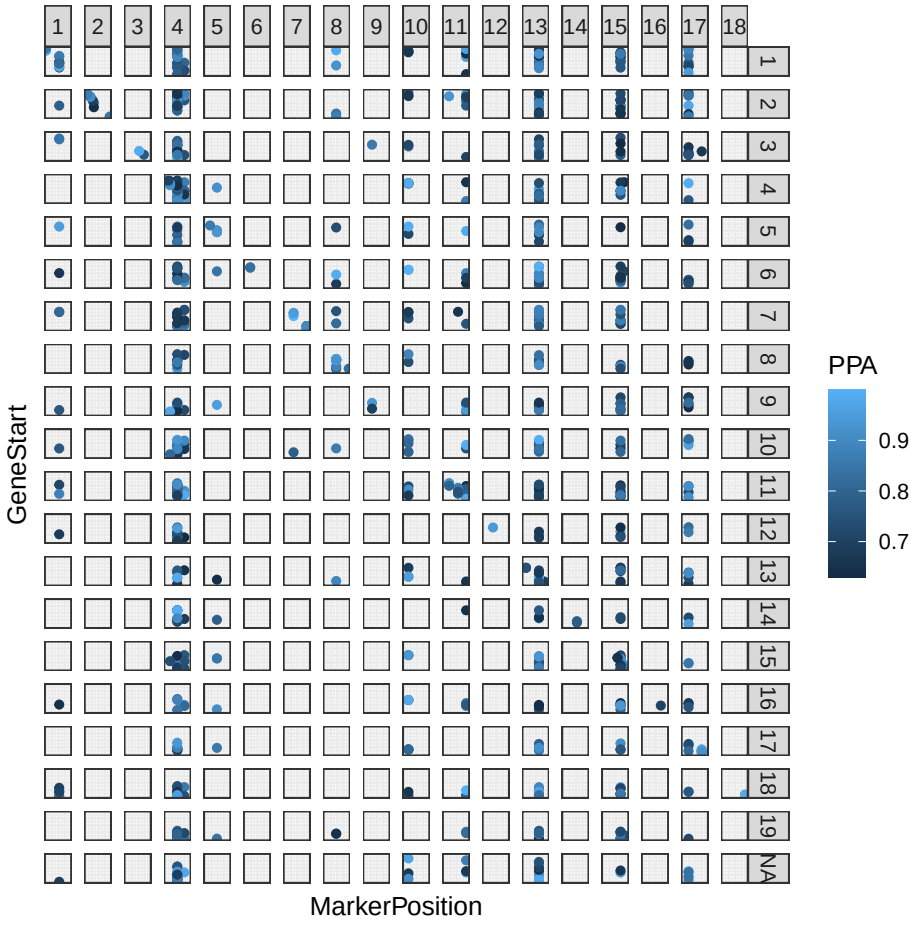
```
> data(probe.liver)

> data(map.liver)

> eqtl.type.liver <- eqtlClassifier(eqtl.liver, map.liver, probe.liver,5000000)
```

We will now plot the results and calculate how many hotspots have been identified.

```
> library(ggplot2)
> ggplot(eqtl.type.liver, aes(y=GeneStart, x=MarkerPosition)) +
+ geom_point(aes(y=GeneStart, x=MarkerPosition,color = PPA), size = 1.5)+
+ facet_grid(GeneChrm~MarkerChrm)+theme_bw(base_size = 12, base_family = "")+
+ theme(axis.ticks = element_blank(), axis.text.x = element_blank(), axis.text.y = element
```



Genome-wide distribution of eQTLs found by iBMQ for the test using 5000 probes for liver tissue from 60 F2 mice. The x-axis gives the position of each eQTL along the genome; the y-axis gives the position of the probe set target itself. The grey lines mark chromosome boundaries. cis-QTLs form a diagonal line. Vertical bands represent groups of transcripts linked to one same trans-eQTL

```
> hotspot.liver <- hotspotFinder(eqtl.type.liver,20)
```

There is a total of 10 hotspots each containing more than 20 genes.

References

[1] Scott-Boyer, MP., Tayeb, G., Imholte, Labbe, A., Deschepper C., and Gottardo R. An integrated Bayesian hierarchical model for multivariate eQTL mapping (iBMQ). Statistical Applications in Genetics and Molecular Biology Vol. 11, 2012.

[2] Geisert, EE., Lu, L., Freeman-Anderson, NE., Templeton, JP., Nassr, M., Wang, X., Gu, W., Jiao, Y., Williams, RW. (2009): Gene expression in the mouse eye: an online resource for genetics using 103 strains of mice. *Mol Vis.*,15, 1730-63

[3] Newton, MA., Noueiry, A., Sarkar, D. and Ahlquist, P. (2004): Detecting differential gene expression with a semiparametric hierarchical mixture method. *Biometrics*, 5(2), 155-176