Genome-Wide Detection of Allele-Specific Gene Expression with BLRM

Jing Xie

2018-12-21

Introduction

The *BLRM* package performs genome-wide detection of allele-specific gene expression (ASE). Compared with most existing methods which do not test allele-specific expression of a gene as a whole and variation of allele-specific expression within a gene across exons separately and simultaneously, the BLRM method closes the gaps by a Bayesian hierarchical generalized linear mixed model which incorporates variation due to various sources and shares information across genes in the whole genome. Bayes factor was utilized to test the hypothesis of allele-specific gene expression for each gene and the ASE variations across SNPs within a gene. In this vignette, we first illustrate the use of this package by providing a quick start guide. Then a more detailed example will show additional functionality and features.

Quick start guide

Given a data set contains gene ID, gene number, SNPs information and the counts data from maternal allele and total counts from both maternal and paternal alleles, if there are R biological replicates at each SNP within a gene, then the allele-specific expression detection based on FDR=0.05 significance level can be performed by simply using the following code:

```
library("BLRM")
rawdata<-read.csv(file="YourRawdata.csv")
hyperparas<- para.est(data=rawdata,rep=R)
res<- detection(data=rawdata,clean_index=hyperparas$index,paras=hyperparas$para,rep=R,fdr=0.05)
list.ASEgene<-res$GeneEffect
list.SNPvariation<-res$SNPEffect
list.ASE.SNP<- res$GSEffect
```

Example

In this section, we will show how to apply *BLRM* package to perform ASE detection step by step. The starting point is to load the raw data set which should be in the exactly same format with the example sample data below. The first three columns of the data set should be gene ID, gene number and the index of SNPs within each gene. The total counts from two alleles and the counts from maternal allele for all the biological replicates are listed in the following columns. The number of columns depends on the number of biological replicates, e.g., there are 4 replicates in this example thus the total number of columns is $11(3 + 2 \times 4)$. Note the order of those columns matters in the raw data set and loading a data set in which the columns were organized differently would result in errors.

```
library("BLRM")
load("mysample.rda")
head(mysample,n=10)
#>         GeneID GeneNum SNP sum1 t1 sum2 t2 sum3 t3 sum4 t4
#> 1  XLOC_000320       1   1   39 24   29 22   NA NA   50 31
#> 2  XLOC_000326       2   1    1  0    1  0   NA NA    1  0
#> 3  XLOC_000326       2   2    1  0   NA NA   NA NA   NA NA
#> 4  XLOC_000326       2   3    2  0    5  2    NA NA    4  1
#> 5  XLOC_000326       2   4    2  1    2  0     1  1    4  1
#> 6  XLOC_000326       2   5    3  2    3  1     5  3    4  3
#> 7  XLOC_000326       2   6   NA NA    3  2    14 11   10  7
#> 8  XLOC_000326       2   7   NA NA   21 13    70 36   68 42
#> 9  XLOC_000327       3   1   NA NA   23 14    48 19   53 24
#> 10 XLOC_000327       3   2   NA NA   19 11    67 32   62 26
```

The first step of *BLRM* workflow is to convert the raw data into a structure which contains necessary information for analysis, i.e., SNPs, Replicates, counts from maternal allele (YI in below example) and total counts (NI in below example). This structure is friendly to GLMM fitting in *glmer*() function in *lme*4 package. This step can be easily completed by repeatedly calling *GDD*() function in *BLRM* for each gene, and for every gene it will return a data set in the structure that would be needed for downstream analysis by *BLRM* method. The below example shows the second gene in the example data set where the argument i means gene number. Fortunately, users don't have to perform the data re-structurization manually because the *GDD*() function was nested with functions in the next steps. But it's still useful when users need to check a specific gene.

```
GDD(i=2,data = mysample, rep = 4)
#>    SNP NI YI Rep
#> 1    1  1  0   1
#> 2    1  1  0   2
#> 3    1  1  0   4
#> 4    2  1  0   1
#> 5    3  2  0   1
#> 6    3  5  2   2
#> 7    3  4  1   4
#> 8    4  2  1   1
#> 9    4  2  0   2
#> 10   4  1  1   3
#> 11   4  4  1   4
#> 12   5  3  2   1
#> 13   5  3  1   2
#> 14   5  5  3   3
#> 15   5  4  3   4
#> 16   6  3  2   2
#> 17   6 14 11   3
#> 18   6 10  7   4
#> 19   7 21 13   2
#> 20   7 70 36   3
#> 21   7 68 42   4
```

The second step is to estimate the hyperparameters which can be conducted by calling *par.est*() function. The *par.est*() tries to fit Generalized Linear Mixed Model to the data of each gene by *glmer*() function in *lme*4 and then filter out the genes with computational problems. The error messages in below chunk show computational problems occurred when calling *glmer*(). The function *par.est*() returns three elements, i.e., the *para* is the hyperparameter estimation and *index* is a vector of the gene number of genes without computational problems, *all* contains detailed intermediate results such as the p_values and estimated FDRs of likelihood ratio tests, the estimates of variance components etc.

```
hyperparas<-para.est(data = mysample, rep = 4)
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
#> error:  grouping factors must have > 1 sampled level
names(hyperparas)
#> [1] "para"  "index" "all"
hyperparas$para
#>        aRs        bRs      aSs       bSs    mlemus  mlesigmas
#> 1 2.025701 0.05465829 2.324607 0.8501158 0.6688857 0.04308877
```

Once the hyperparameter estimation and the index of genes without computation problems were achieved, the final step is to apply *detection*() function to conduct hypothesis testing. The *detection*() function returns three data frames corresponding to three situations of gene expression, as well as an additional data frame contains intermediate results. The $GeneEffect$ shows the testing results of genes exhibiting significant ASE gene effect, where PP denotes the posterior probability, FPP is $1 - PP$, and FDR is the estimated false discovery rate. Similary, the $SNPEffect$ shows the results of genes with significant ASE variation across SNPs; the $GSEffect$ corresponds to genes exhibiting both ASE gene effect and ASE variation across SNPs.

```
res<- detection(data=mysample,clean_index=hyperparas$index,paras=hyperparas$para,rep=4,fdr=0.05)
list.ASEgene<-res$GeneEffect
list.SNPvariation<-res$SNPEffect
list.ASE.SNP<- res$GSEffect
head(list.ASEgene,n=10)
#>     geneNum      geneID        PP          FPP          FDR
#> 252     273 XLOC_001005 0.9999015 9.849872e-05 9.849872e-05
#> 228     249 XLOC_000935 0.9998153 1.846972e-04 1.415979e-04
#> 233     254 XLOC_000942 0.9983848 1.615220e-03 6.328053e-04
#> 207     226 XLOC_000892 0.9966239 3.376120e-03 1.318634e-03
#> 119     131 XLOC_000663 0.9961231 3.876887e-03 1.830285e-03
#> 83       92 XLOC_000572 0.9960092 3.990832e-03 2.190376e-03
#> 101     111 XLOC_000628 0.9958003 4.199742e-03 2.477428e-03
#> 235     256 XLOC_000955 0.9929566 7.043388e-03 3.048173e-03
#> 271     292 XLOC_001059 0.9833092 1.669082e-02 4.564023e-03
#> 41       45 XLOC_000460 0.9673848 3.261524e-02 7.369144e-03
head(list.SNPvariation,n=10)
#>     geneNum      geneID PP FPP FDR
#> 3         4 XLOC_000329  1   0   0
#> 18       19 XLOC_000376  1   0   0
#> 40       44 XLOC_000459  1   0   0
#> 56       60 XLOC_000503  1   0   0
#> 64       68 XLOC_000517  1   0   0
#> 85       94 XLOC_000578  1   0   0
#> 137     152 XLOC_000728  1   0   0
#> 147     164 XLOC_000754  1   0   0
#> 186     204 XLOC_000845  1   0   0
#> 188     206 XLOC_000847  1   0   0
head(list.ASE.SNP,n=10)
#>     geneNum      geneID        PP          FPP          FDR
#> 252     273 XLOC_001005 0.9999015 9.849872e-05 9.849872e-05
#> 228     249 XLOC_000935 0.9998153 1.846972e-04 1.415979e-04
#> 233     254 XLOC_000942 0.9983847 1.615319e-03 6.328384e-04
#> 119     131 XLOC_000663 0.9961231 3.876887e-03 1.443851e-03
#> 101     111 XLOC_000628 0.9957974 4.202612e-03 1.995603e-03
#> 240     261 XLOC_000967 0.9365702 6.342980e-02 1.223463e-02
#> 186     204 XLOC_000845 0.9048908 9.510916e-02 2.407385e-02
#> 264     285 XLOC_001043 0.9019184 9.808156e-02 3.332482e-02
#> 23       25 XLOC_000398 0.8969296 1.030704e-01 4.107433e-02
```