GESE package vignette

Dandi Qiao

2017-05-18

Contents

1 Introduction

This tutorial briefly introduces the functions provided by the GESE package, using the example data included in the package. The paper describing this method is "Qiao, D. Lange, C., Laird, N.M., Won, S., Hersh, C.P., et al. 2017. Gene-based segregation method for identifying rare variants for family-based sequencing studies. Genet Epidemiol 41(4):309-319". We have also implemented a simple pipeline to compute the GESE p-values using this R library, which is described in detail at http://scholar.harvard.edu/dqiao/gese.

We can load the library using:

```r
library(GESE)
```

```
## Loading required package: kinship2
## Loading required package: Matrix
## Loading required package: quadprog
```

2 Example data

A randomly simulated example data is included in the package. This data includes 198 sequenced subjects in 50 families. Only 2 genes with 10 variants each are included for these subjects. With real sequencing data, the first step is to filter down to the rare and functionally important variants since we hypothesize that there are rare causal variants of large effects for Mendelian diseases, or Mendelian-subtypes of complex diseases. One way is to filter the study data and the reference data using MAF $< 0.1\%$ and CADD score[1] > 15 on

---

[1] Kircher, M, Witten, DM, Jain, P, O'Roak, BJ, Cooper, GM, and Shendure, J. A general framework for estimating the relative pathogenicity of human genetic variants. Nature genetics 2014; 46(3):310–315.

LoF (and missense) variants. Other annotation tools such as SIFT and Polyphen2 could also be helpful in filtering down to the functional variants.

After the filtering step, we can load the variant data of the sequenced subjects. This data frame is in the PLINK raw format (with the default header generated by PLINK). We need to make sure that the genotype is the number of the minor alleles in the corresponding population.

```
data(dataRaw)
dim(dataRaw)
```

```
## [1] 198  16
```

```
dataRaw[1:10,]
```

```
##    FID   IID  PAT   MAT SEX PHENOTYPE X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
## 1  267 13456  534   533   2         1  0  0  0  0  1  0  0  0  0   0
## 2  267 13466 3067 13463   2         0  0  0  0  0  0  0  0  0  0   0
## 3  267 13468 3067 13463   2         0  0  0  0  0  0  0  0  0  0   0
## 4  267 13470  534   533   2         1  0  0  0  0  0  0  0  0  0   0
## 5  267 13475 3068 13470   2         1  0  0  0  0  0  0  0  0  0   0
## 6  145  2578    0     0   1         1  0  0  0  0  0  0  0  0  0   0
## 7  145 10036 2577 10033   2         0  0  0  0  0  0  0  0  0  0   0
## 8  145 10045 2578 10040   2         1  0  0  0  0  0  0  0  0  0   0
## 9  763 27340 5049 27337   2         1  0  0  0  0  0  0  0  0  0   0
## 10 763 27343 5049 27337   2         1  0  0  0  0  0  0  0  0  0   0
```

```
length(unique(dataRaw$FID))
```

```
## [1] 50
```

The corresponding map file with variant information can be loaded next. The order of the variants needs to match the order in the dataRaw file above.

```
data(mapInfo)
mapInfo[1:3, ]
```

```
##       GENE SNP
## 1 GENE_1_1   1
## 2 GENE_1_1   2
## 3 GENE_1_1   3
```

```
dim(mapInfo)
```

```
## [1] 10  2
```

The complete pedigree information for the 50 families can be loaded next:

```
data(pednew)
dim(pednew)
```

```
## [1] 1700    5
```

```
pednew[1:5,]
```

```
##   FID IID faID moID sex
## 1  16  31   NA   NA   2
## 2  16  32   NA   NA   1
## 3  33  65   NA   NA   2
## 4  33  66   NA   NA   1
## 5  34  67   NA   NA   2
```

The complete variant information for the corresponding population can be loaded next. This reference database is used to compute the background variation in the corresponding population.

```
data(database)
dim(database)
```

```
## [1] 10  3
```

```
database[1:5,]
```

```
##   SNP     GENE          MAF
## 1   1 GENE_1_1 2.386785e-05
## 2   2 GENE_1_1 7.634928e-05
## 3   3 GENE_1_1 7.546418e-05
## 4   4 GENE_1_1 7.826542e-05
## 5   5 GENE_1_1 4.406916e-05
```

3 Functions

The main function for obtaining gene-based segregation information and (unweighted and weighted) segregation tests is `GESE`. Other functions that may be helpful in analyzing family-based sequencing data will be discussed too.

3.1 Compute variant-based and gene-based segregation information for different mode of inheritance.

One major function in this package for computing segregation information for different mode of inheritance is `getSegInfo`. This function returns the variant-based and gene-based segregation information for each family and the total number of segregating families for three different mode of inheritance: dominant, recessive, and compound heterozygous.

For example, segregation in the family with dominant mode of inheritance means the variant is present in all the cases in the family, and absent in all the controls in the family. Therefore variants that are segregating in multiple families are more likely to be causal.

Since it is likely that different rare variants are influencing the disease susceptibility in different families, collapsing the variants into genes may give us more power to detect the causal genes. Therefore we also need to compute the total number of families that are segregating in each gene.

For recessive mode of inheritance, segregation means two copies of the alternative alleles are present in all the cases in the family, and less than two copies in all the controls of the family (`varSeg`). This information can also be collapsed for each gene (`geneSeg`).

For compound heterozygous (CH) mode of inheritance, segregation at two variants means the alternative alleles are present at both loci for all the cases in the family, and absent in at least one locus in all the controls in the family (`varSeg`). This information can be collapsed for each gene, where only pairs of variants in the same gene are considred (`geneSeg`). This can also be collapsed for each pair of genes, where pairs of variants from each of the two genes are considered (`genePairSeg`).

The data frame `varSeg` is a matrix containing logical value (TRUE or FALSE) for each variant (each row) and each family (each column). The first column is the variant ID. The last column `numSegFam` is the total number of families the variant is segregating in. The data frame `geneSeg` is also a matrix containing logical values, for each gene and each family. TRUE value means that at least one variant in this gene is segregating in the family.

We demonstrate the use of this function below.

To compute segregation information for domiant mode of inheritance:

```
results <- getSegInfo(pednew,  dataRaw, mapInfo, mode="dominant")
results
```

```
## $geneSeg
##       GENE   267   145   763   686   828   612   93    34   606   252
## 1 GENE_1_1 FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE
##      96   960   414  566   228   986   554   660    16   849   101   150
## 1 FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##     336   889   398   230   711   401   210   664   684   393   867   332
## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      99   658   137   571    87   477   570   412   557   266   845   534
## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##     746   236   216    33 numSegFam
## 1 FALSE FALSE FALSE FALSE         2
## 
## $varSeg
##        GENE SNP   267   145   763   686   828   612    93    34   606
## 2  GENE_1_1   2 FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## 8  GENE_1_1   8 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1  GENE_1_1   1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  GENE_1_1   3 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  GENE_1_1   4 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  GENE_1_1   5 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  GENE_1_1   6 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  GENE_1_1   7 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  GENE_1_1   9 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 GENE_1_1  10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      252    96   960   414   566   228   986   554   660    16   849   101
## 2  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      150   336   889   398   230   711   401   210   664   684   393   867
```

```
## 2  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      332    99   658   137   571    87   477   570   412   557   266   845
## 2  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      534   746   236   216    33 numSegFam
## 2  FALSE FALSE FALSE FALSE FALSE         1
## 8  FALSE FALSE FALSE FALSE FALSE         1
## 1  FALSE FALSE FALSE FALSE FALSE         0
## 3  FALSE FALSE FALSE FALSE FALSE         0
## 4  FALSE FALSE FALSE FALSE FALSE         0
## 5  FALSE FALSE FALSE FALSE FALSE         0
## 6  FALSE FALSE FALSE FALSE FALSE         0
## 7  FALSE FALSE FALSE FALSE FALSE         0
## 9  FALSE FALSE FALSE FALSE FALSE         0
## 10 FALSE FALSE FALSE FALSE FALSE         0
```

To compute segregation information for recessive mode of inheritance:

```
results <- getSegInfo(pednew,  dataRaw, mapInfo)
results
```

```
## $geneSeg
##       GENE   267   145   763   686   828   612    93    34   606   252
## 1 GENE_1_1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      96   960   414   566   228   986   554   660    16   849   101   150
## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##     336   889   398   230   711   401   210   664   684   393   867   332
## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      99   658   137   571    87   477   570   412   557   266   845   534
## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##     746   236   216    33 numSegFam
## 1 FALSE FALSE FALSE FALSE         0
##
## $varSeg
##        GENE SNP   267   145   763   686   828   612    93    34   606
## 1  GENE_1_1   1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 2  GENE_1_1   2 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## 3  GENE_1_1   3 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  GENE_1_1   4 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  GENE_1_1   5 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  GENE_1_1   6 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  GENE_1_1   7 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  GENE_1_1   8 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  GENE_1_1   9 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 GENE_1_1  10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      252    96   960   414   566   228   986   554   660    16   849   101
## 1  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 2  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      150   336   889   398   230   711   401   210   664   684   393   867
## 1  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 2  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      332    99   658   137   571    87   477   570   412   557   266   845
## 1  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 2  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      534   746   236   216    33 numSegFam
## 1  FALSE FALSE FALSE FALSE FALSE         0
## 2  FALSE FALSE FALSE FALSE FALSE         0
## 3  FALSE FALSE FALSE FALSE FALSE         0
## 4  FALSE FALSE FALSE FALSE FALSE         0
## 5  FALSE FALSE FALSE FALSE FALSE         0
## 6  FALSE FALSE FALSE FALSE FALSE         0
## 7  FALSE FALSE FALSE FALSE FALSE         0
## 8  FALSE FALSE FALSE FALSE FALSE         0
## 9  FALSE FALSE FALSE FALSE FALSE         0
## 10 FALSE FALSE FALSE FALSE FALSE         0
```

To compute segregation information for compound heterozygous mode of inheritance:

```
results <- getSegInfo(pednew,  dataRaw, mapInfo, mode="CH")
results
```

```
## $geneSeg
## [1] NA
## 
## $genePairSeg
## [1] NA
## 
## $varSeg
##  [1] ID        GENE.x    GENE.y    267       145       763       686      
##  [8] 828       612       93        34        606       252       96       
## [15] 960       414       566       228       986       554       660      
## [22] 16        849       101       150       336       889       398      
## [29] 230       711       401       210       664       684       393      
## [36] 867       332       99        658       137       571       87       
## [43] 477       570       412       557       266       845       534      
## [50] 746       236       216       33        numSegFam
## <0 rows> (or 0-length row.names)
```

The `geneSeg` or `genePairSeg` return NA values, because there is no pair of variant in any gene, or gene pair that is segregating in any family, with compound heterozygous mode of inheritance.

Alternatively, we can compute segregation with dominant mode of inheritance wthout computing any probabilities using the `GESE` function and specify `onlySeg` to be TRUE:

```
results <- GESE(pednew, database, 1000000, dataRaw, mapInfo, threshold=1e-5,
                onlySeg=TRUE)
results
```

```
## $segregation
##       GENE   267   145   763   686   828   612   93    34   606   252
## 1 GENE_1_1 FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE
##      96   960   414  566   228   986   554   660    16   849   101   150
## 1 FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##     336   889   398   230   711   401   210   664   684   393   867   332
## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      99   658   137   571    87   477   570   412   557   266   845   534
## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##     746   236   216    33 numSegFam
## 1 FALSE FALSE FALSE FALSE         2
## 
## $varSeg
##        GENE SNP   267   145   763   686   828   612    93    34   606
## 2  GENE_1_1   2 FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## 8  GENE_1_1   8 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1  GENE_1_1   1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  GENE_1_1   3 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  GENE_1_1   4 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  GENE_1_1   5 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  GENE_1_1   6 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  GENE_1_1   7 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  GENE_1_1   9 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## 10 GENE_1_1  10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      252    96   960   414   566   228   986   554   660    16   849   101
## 2  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      150   336   889   398   230   711   401   210   664   684   393   867
## 2  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      332    99   658   137   571    87   477   570   412   557   266   845
## 2  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      534   746   236   216    33 numSegFam
## 2  FALSE FALSE FALSE FALSE FALSE         1
## 8  FALSE FALSE FALSE FALSE FALSE         1
## 1  FALSE FALSE FALSE FALSE FALSE         0
## 3  FALSE FALSE FALSE FALSE FALSE         0
## 4  FALSE FALSE FALSE FALSE FALSE         0
## 5  FALSE FALSE FALSE FALSE FALSE         0
## 6  FALSE FALSE FALSE FALSE FALSE         0
## 7  FALSE FALSE FALSE FALSE FALSE         0
## 9  FALSE FALSE FALSE FALSE FALSE         0
## 10 FALSE FALSE FALSE FALSE FALSE         0
```

Similarly, the `segregation` value returned is a data matrix, where a logical value (TRUE or FALSE) is returned for each gene(each row) and each family (column names). The last column `numSegFam` sums each row, which gives the total number of pedigrees each gene segregates in. The families were first trimmed to satisfy the assumption of GESE, so that only one founder case is present for each pedigree.

The `varSeg` value returned is a similar data matrix, but returns a logical value for each variant (each row) and each family (column names).

3.2 Compute gene-based segregation test.

The gene-based segregation information may be helpful, however, it does not take into account the different family structure among the families, nor the different genetic background of different genes. The gene-based segregation test combines this information by approximating the marginal probability of segregation events among the families.

To compute the p-value for this test, there are a few steps in the process.

- First, as we mentioned, we need to make sure that only variants that are rare and functionally important are included in the data, to satisfy the assumptions of the method. For example, we used MAF $< 0.1\%$, CADD score > 15, LoF and missense variants as filtering criterion for the Boston Early-Onset COPD data, described in the paper. Since we are looking for extremely rare variant of large effect for the disease, such filtering is reasonable.

To ensure that only one founder case is present for each family, we will trim the pedigrees to keep only the founder case that is related to most other non-founder cases if necessary.

- Second, we need to compute the conditional probabilities that a variant segregates in the family conditional on that variant in present in one of the founders.
- Third, we need to compute the marginal probability that at least one variant in the gene segregates in the family.
- Fourth, we need to compute the final p-value using the marginal segregation probabilities computed above.

These steps can be done using the GESE function in one call:

```
results2 <- GESE(pednew, database, 1000000, dataRaw, mapInfo, threshold=1e-5)
results2
```

```
## $results
##       GENE     obs_prob pvalue numSim N_seg
## 1 GENE_1_1 2.558034e-09      0  1e+05     2
##
## $condSegProb
##    fam      condP
## 1  267 0.15625000
## 2  145 0.50000000
## 3  763 0.37500000
## 4  686 0.21875000
## 5  828 0.04687500
## 6  612 0.06250000
## 7   93 0.09375000
## 8   34 0.18750000
## 9  606 0.28125000
## 10 252 0.50000000
## 11  96 0.31250000
## 12 960 0.18750000
## 13 414 0.25000000
## 14 566 0.04687500
## 15 228 0.43750000
## 16 986 0.06250000
```

```
## 17 554 0.09375000
## 18 660 0.09375000
## 19  16 0.09375000
## 20 849 0.43750000
## 21 101 0.11718750
## 22 150 0.50000000
## 23 336 0.50000000
## 24 889 0.12500000
## 25 398 0.09375000
## 26 230 0.21875000
## 27 711 0.37500000
## 28 401 0.04687500
## 29 210 0.06250000
## 30 664 0.25000000
## 31 684 0.03515625
## 32 393 0.06250000
## 33 867 0.18750000
## 34 332 0.25000000
## 35  99 0.12500000
## 36 658 0.03125000
## 37 137 0.12500000
## 38 571 0.04687500
## 39  87 0.12500000
## 40 477 0.25000000
## 41 570 0.43750000
## 42 412 0.06250000
## 43 557 0.09375000
## 44 266 0.37500000
## 45 845 0.43750000
## 46 534 0.06250000
## 47 746 0.18750000
## 48 236 0.02343750
## 49 216 0.25000000
## 50  33 0.15625000
##
## $segProbGene
##       GENE          267          145          763          686         828
## 1 GENE_1_1 0.0001196611 0.0003828729 0.0002871663 0.0001675222 3.58996e-05
##           612           93           34          606          252
## 1 4.78659e-05 7.179812e-05 0.0001435919 0.0002153813 0.0003828729
##             96          960         414         566          228
## 1 0.0002393101 0.0001435919 0.000191452 3.58996e-05 0.0003350206
##           986          554          660           16          849
## 1 4.78659e-05 7.179812e-05 7.179812e-05 7.179812e-05 0.0003350206
##            101          150          336          889          398
## 1 8.974696e-05 0.0003828729 0.0003828729 9.572985e-05 7.179812e-05
##            230          711         401         210         664
## 1 0.0001675222 0.0002871663 3.58996e-05 4.78659e-05 0.000191452
##           684         393          867         332           99
## 1 2.69248e-05 4.78659e-05 0.0001435919 0.000191452 9.572985e-05
##            658          137         571           87         477
## 1 2.393319e-05 9.572985e-05 3.58996e-05 9.572985e-05 0.000191452
##            570         412          557          266          845
## 1 0.0003350206 4.78659e-05 7.179812e-05 0.0002871663 0.0003350206
```

```
##           534          746          236         216           33
## 1 4.78659e-05 0.0001435919 1.794994e-05 0.000191452 0.0001196611
## 
## $segregation
##       GENE   267   145   763   686   828   612   93    34   606   252
## 1 GENE_1_1 FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE
##      96   960   414  566   228   986   554   660    16   849   101   150
## 1 FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##     336   889   398   230   711   401   210   664   684   393   867   332
## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      99   658   137   571    87   477   570   412   557   266   845   534
## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##     746   236   216    33 numSegFam
## 1 FALSE FALSE FALSE FALSE         2
## 
## $varSeg
##        GENE SNP   267   145   763   686   828   612    93    34   606
## 2  GENE_1_1   2 FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## 8  GENE_1_1   8 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1  GENE_1_1   1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  GENE_1_1   3 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  GENE_1_1   4 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  GENE_1_1   5 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  GENE_1_1   6 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  GENE_1_1   7 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  GENE_1_1   9 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 GENE_1_1  10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##       252    96   960   414   566   228   986   554   660    16   849   101
## 2  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##       150   336   889   398   230   711   401   210   664   684   393   867
## 2  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##       332    99   658   137   571    87   477   570   412   557   266   845
## 2  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 8  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## 4  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 6  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##      534   746   236   216    33 numSegFam
## 2  FALSE FALSE FALSE FALSE FALSE         1
## 8  FALSE FALSE FALSE FALSE FALSE         1
## 1  FALSE FALSE FALSE FALSE FALSE         0
## 3  FALSE FALSE FALSE FALSE FALSE         0
## 4  FALSE FALSE FALSE FALSE FALSE         0
## 5  FALSE FALSE FALSE FALSE FALSE         0
## 6  FALSE FALSE FALSE FALSE FALSE         0
## 7  FALSE FALSE FALSE FALSE FALSE         0
## 9  FALSE FALSE FALSE FALSE FALSE         0
## 10 FALSE FALSE FALSE FALSE FALSE         0
```

This call computes the GESE p-value using resampling with 1e5 times of simulations (so the smallest p-value is 1e-5). There are several other matrices returned by this call, such as the conditional segregation probability (`condSegProb`) - the conditional probability of segregation event in the family conditioning on that one of the (most recent common) founders introduced the variant to the family; the gene-based segregation probability (`segProbGene`) - the marginal probability that any variant in the gene is segregating in the family; and the variant-based and gene-based segregation matrices (`varSeg` and `segregation`).

The most useful results containing the GESE p-value here is:

```
results2$results
```

```
##       GENE     obs_prob pvalue numSim N_seg
## 1 GENE_1_1 2.558034e-09      0  1e+05     2
```

3.3 Compute the weighted gene-based segregation test.

We can also incorporate additional information, such as disease severity of the cases in the families, in the weighting of the families. We can also compute the p-value for such weighted test, using resampling.

Suppose we have a disease severity measure of the cases in the families, and we are using such weighting of families for all the genes.

```
### creating weights for the families (same weights for all genes)
fams <- unique(dataRaw$FID)
nfam = length(fams)
temp = runif(nfam)
famWeight <- temp/sum(temp)
weightFam = data.frame(FID=fams, weight=famWeight)
results3 <- GESE(pednew, database, 1000000, dataRaw, mapInfo, threshold=1e-5,
             familyWeight=weightFam)
results3$results
```

```
##       GENE     obs_prob pvalue obs_weight_stat pvalue_weighted numSim
## 1 GENE_1_1 2.558034e-09      0       0.8265824          0.0032  1e+05
##   N_seg
## 1     2
```

3.4. Other useful methods

There are a few public methods that were used in the GESE test and may also be useful in other contexts.

3.4.1. Trimming the pedigree file

This method `trim_oneLineage` accepts the complete pedigree information and selects only one lineage per pedigree. The lineage is selected such that the maximum possible number of sequenced subects (cases) are included in the selected lineage. Other family-based methods, such as PVAAST, also requires one lineage in each pedigree only.

The method `trim_unrelated` deals with families with multiple founder cases, which violates our assumption that only one founder introduced the causal variant. It removes the minimal number of founder cases so that the pedigree does not violate the assumption of the method.

3.4.2. Compute the conditional segregating probability

The method `condSegProbF` computes the probability that the variant is segregating in the family given that it is introducted by the most recent common ancestors of the cases in the family.