GIGSEA: Genotype Imputed Gene Set Enrichment Analysis

Shijia Zhu

Department of Genetics and Genomic Sciences and Icahn Institute for Genomics and Multiscale Biology, Icahn School of Medicine at Mount Sinai, New York, NY 10029, USA

August 28, 2017

Contents

Abstract

We presented the Genotype-imputed Gene Set Enrichment Analysis (GIGSEA), a novel method that uses GWAS-and-eQTL-imputed trait-associated differential gene expression to interrogate gene set enrichment for the trait-associated SNPs. By incorporating eQTL from large gene expression studies, e.g. GTEx, GIGSEA appropriately addresses such challenges for SNP enrichment as gene size, gene boundary, SNP distal regulation, and multiple-marker regulation. The weighted linear regression model, taking as weights both imputation accuracy and model completeness, was used to perform the enrichment test, properly adjusting the bias due to redundancy in different gene sets. The permutation test, furthermore, is used to evaluate the significance of enrichment, whose efficiency can be largely elevated by expressing the computational intensive part in terms of large matrix operation. We have shown the appropriate type I error rates for GIGSEA (<5%), and the preliminary results also demonstrate its good performance to uncover the real signal.

1. Import packages

GIGSEA depends on the R package Matrix, which has already been built in R. In GIGSEA, the gene sets are saved as matrices. Such matrices are largely sparse, so, in order to save space, we used the functions provided by the Matrix package to build the sparse matrices and pre-saved into the GIGSEA package.

```
library(GIGSEA)
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 3.1.3
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:base':
##
##     crossprod, tcrossprod
```

```
library(Matrix)
```

2. Quick start

GIGSEA first uses MetaXcan to impute trait-associated differential gene expression from both GWAS summary and eQTL database with LD structure adjusted, and next, builds a weighted regression model to perfrom gene set enrichment analysis. In user's convenience, we combine these procedures together into one function `runGIGSEA()`, which writes the enrichment test results at the local directory. Users only need to provide their GWAS summary data, and specify the paths to the MetaXcan.py file, the eQTL database (e.g. GTex and DGN) and the reference popultation (e.g. 1000 Genome).

```
#runGIGSEA( MetaXcan="software/MetaXcan.py" , model_db_path="eQTL/DGN-WB_0.5.db", covariance="reference
```

3. Load data

GIGSEA is built on the weighted regression model, so it permits both discrete-valued and continuous-valued gene sets. We already incorporated several gene sets into the GIGSEA package, including:

1) **discrete-valued gene sets**:

- `MSigDB.KEGG.Pathway`: Gene sets derived from the KEGG pathway database. It comprises 186 pathways (column) and 5267 genes (row). See c2.cp.kegg.v6.0.symbols.gmt.txt at http://software.broadinstitute.org/gsea/msigdb/collections.jsp#C2
- `MSigDB.TF`: Gene sets that share upstream cis-regulatory motifs which can function as potential transcription factor binding sites. It comprises 615 TFs (column) and 12774 genes (row). See c3.tft.v6.0.symbols.gmt.txt at http://software.broadinstitute.org/gsea/msigdb/collections.jsp#C3
- `MSigDB.miRNA`: Gene sets that contain genes sharing putative target sites (seed matches) of human mature miRNA in their 3'-UTRs. It comprises 221 miRNAs (column) and 7444 genes (row). See c3.mir.v6.0.symbols.gmt.txt at http://software.broadinstitute.org/gsea/msigdb/collections.jsp#C3
- `GO`: Gene sets that contain genes annotated by the same Gene Ontology (GO) term. For each GO term, we not only incorporate its own gene sets, but also incorporate the gene sets belonging to its offsprings. See the database GO.db in R.

2) **continuous-valued gene sets**:

- `Fantom5.TF`: Gene sets of predicted human transcriptional factor (TF) target sites were downloaded from Fantom5. The dataset contains 500 WMs. For each WM, there is a list of associated human TFs, ordered by percent identity of TFs known to bind sites of the WM. The list of associations was checked manually. The entire set of WMs and mapping to associated TFs is available from the SwissRegulon website http://www.swissregulon.unibas.ch.
- `TargetScan.miRNA`: Gene sets of predicted human miRNA target sites were downloaded from TargetScan. TargetScan groups miRNAs that have identical subsequences at positions 2 through 8 of the miRNA, i.e. the 2-7 seed region plus the 8th nucleotide, and provides predictions for each such seed motif. TargetScan covers 87 human miRNA seed motifs in total. It provides a score for each seed motif and each RefSeq transcript, called preferential conservation scoring (aggregate Pct), which shows consistently high performance in various benchmark tests. To obtain a site count associated with each gene, we average the TargetScan Pct scores of all RefSeq transcripts associated with each gene. It comprises 87 miRNA seed motifs and 9861 genes. See http://www.targetscan.org.
- `LINCS.CMap.drug`: Large perturbational datasets of gene expression signature from small-molecule compounds in multiple cell types from LINCS/CMap database. We downloaded the data of LINCS phase 2 level 5 from GEO (GSE70138). The data is saved in the GCTx format (binary format based on HDF5 that enables fast i/o than text), and we parsed it using the R package cmapR. The LINCS level 5 data is a numeric matrix, comprising 118050 drugs/doses and 12328 genes. Each entry is a replicate-collapsed z-score of differential gene expression due to a drug perturbation, which is calculated by aggregating across individual replicates. In order to generate a single signature for each drug perturbation, we further average the differential gene expression across different drug doses, resulting in a condensed matrix of 1826 drugs (column) and 12328 genes (row). See https://clue.io.

3.1 Discrete-valued gene sets:

We first show an example of discrete-valued gene set: MSigDB.KEGG.Pathway, where the row represents the gene, and the column represents the pathway. Each entry takes discrete values of 0 or 1, where 1 represents the gene (row) belongs to the pathway (column), and otherwise, not.

```
# load data
data(MSigDB.KEGG.Pathway)
# MSigDB.KEGG.Pathway is a list comprising two components: net and annot
class(MSigDB.KEGG.Pathway)
```

```
## [1] "list"
```

```
names(MSigDB.KEGG.Pathway)
```

```
## [1] "net"   "annot"
```

```
dim(MSigDB.KEGG.Pathway$net)
```

```
## [1] 5267  186
```

```
# the column is the pathway and the row is the gene
head(colnames(MSigDB.KEGG.Pathway$net))
```

```
## [1] "KEGG_GLYCOLYSIS_GLUCONEOGENESIS"
## [2] "KEGG_CITRATE_CYCLE_TCA_CYCLE"
## [3] "KEGG_PENTOSE_PHOSPHATE_PATHWAY"
## [4] "KEGG_PENTOSE_AND_GLUCURONATE_INTERCONVERSIONS"
## [5] "KEGG_FRUCTOSE_AND_MANNOSE_METABOLISM"
## [6] "KEGG_GALACTOSE_METABOLISM"
```

```
head(rownames(MSigDB.KEGG.Pathway$net))
```

```
## [1] "A2M"    "A4GALT" "AACS"   "AADAT"  "AANAT"  "AARS"
```

```
# the annotation of the pathway
head(MSigDB.KEGG.Pathway$annot)
```

```
##                                         term
## 1               KEGG_GLYCOLYSIS_GLUCONEOGENESIS
## 2                  KEGG_CITRATE_CYCLE_TCA_CYCLE
## 3                KEGG_PENTOSE_PHOSPHATE_PATHWAY
## 4 KEGG_PENTOSE_AND_GLUCURONATE_INTERCONVERSIONS
## 5          KEGG_FRUCTOSE_AND_MANNOSE_METABOLISM
## 6                     KEGG_GALACTOSE_METABOLISM
##                                                                                    link
## 1               http://www.broadinstitute.org/gsea/msigdb/cards/KEGG_GLYCOLYSIS_GLUCONEOGENESIS
## 2                  http://www.broadinstitute.org/gsea/msigdb/cards/KEGG_CITRATE_CYCLE_TCA_CYCLE
## 3                http://www.broadinstitute.org/gsea/msigdb/cards/KEGG_PENTOSE_PHOSPHATE_PATHWAY
## 4 http://www.broadinstitute.org/gsea/msigdb/cards/KEGG_PENTOSE_AND_GLUCURONATE_INTERCONVERSIONS
## 5          http://www.broadinstitute.org/gsea/msigdb/cards/KEGG_FRUCTOSE_AND_MANNOSE_METABOLISM
## 6                     http://www.broadinstitute.org/gsea/msigdb/cards/KEGG_GALACTOSE_METABOLISM
##   totalGenes
## 1         62
## 2         32
## 3         27
## 4         28
## 5         34
## 6         26
```

```
# the net takes discrete values of 0 or 1
head(MSigDB.KEGG.Pathway$net[,1:30])
```

```
## 6 x 30 sparse Matrix of class "dgCMatrix"
```

```
##    [[ suppressing 30 column names 'KEGG_GLYCOLYSIS_GLUCONEOGENESIS', 'KEGG_CITRATE_CYCLE_TCA_CYCLE',
```

```
##
## A2M    . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## A4GALT . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## AACS   . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## AADAT  . . . . . . . . . . . . . . . . . . . . . . 1 . . . . 1 . . . . .
## AANAT  . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . . . .
## AARS   . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

3.2 Continuous-valued gene sets:

Followed is an example of continuous-valued gene set: TargetScan.miRNA, where the row represents the gene, and the column represents the miRNA. Each entry takes continuous values of pct, representing the binding affinity of miRNA on the gene 3' UTR.

```r
# load data
data(TargetScan.miRNA)
# TargetScan.miRNA is a list comprising two components: net and annot
class(TargetScan.miRNA)
```

```
## [1] "list"
```

```r
names(TargetScan.miRNA)
```

```
## [1] "net"   "annot"
```

```r
dim(TargetScan.miRNA$net)
```

```
## [1] 9861   87
```

```r
# the column is the miRNA and the row is the gene
head(colnames(TargetScan.miRNA$net))
```

```
## [1] "hsa-let-7a hsa-let-7b hsa-let-7c hsa-let-7d hsa-let-7e hsa-let-7f hsa-miR-98 hsa-let-7g hsa-let-
## [2] "hsa-miR-1 hsa-miR-206 hsa-miR-613"
## [3] "hsa-miR-101"
## [4] "hsa-miR-103a hsa-miR-107"
## [5] "hsa-miR-10a hsa-miR-10b"
## [6] "hsa-miR-122"
```

```r
head(rownames(TargetScan.miRNA$net))
```

```
## [1] "A1CF"  "A2LD1" "AAGAB" "AAK1"  "AAMP"  "AARS"
```

```r
# the annotation of the miRNA
head(TargetScan.miRNA$annot)
```

```
##
## 49 hsa-let-7a hsa-let-7b hsa-let-7c hsa-let-7d hsa-let-7e hsa-let-7f hsa-miR-98 hsa-let-7g hsa-let-7
## 57                                                                                          hsa-mi
## 14
## 53
## 18
## 59
##     family totalGenes
## 49 GAGGUAG       1046
## 57 GGAAUGU        769
## 14 ACAGUAC        788
## 53 GCAGCAU        638
## 18 ACCCUGU        216
## 59 GGAGUGU        112
```

```
# the net takes continuous values
head(TargetScan.miRNA$net[,1:20])
```

```
## 6 x 20 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 20 column names 'hsa-let-7a hsa-let-7b hsa-let-7c hsa-let-7d hsa-let-7e hsa-let-7f

##
## A1CF  0.000 . . . .     . .     . . . 0.331 .     .     . . . .     . . .
## A2LD1 .     . . . .     . .     . . . .     .     .     . . . .     . . .
## AAGAB .     . . . .     . .     . . . .     .     .     . . . .     . . .
## AAK1  0.922 . . . 0.581 0 0.836 . . . .     0.951 0.879 . . . 0.376 . . .
## AAMP  .     . . . .     . .     . . . .     .     .     . . . .     . . .
## AARS  .     . . . .     . .     . . . .     .     .     . . . .     . . .
```

3.3 User self-defined gene set

In adition to the above predefined gene sets, users can also specify their own gene set. Here, we take as an example the CREEDS http://amp.pharm.mssm.edu/CREEDS/. It is a manually curated database of gene signatures of single drug perturbations. For each drug perturbation, it lists both up-regulated and down-regulated gene sets. In the following example, we transform the gmt format file into a sparse matrix, where for each drug perturbation, the up-regulated genes take the value of 1, the down-regulated genes take the value of -1, and the others take the value of 0.

```
# downlaod the gmt file
gmt <- readLines('http://amp.pharm.mssm.edu/CREEDS/download/single_drug_perturbations-v1.0.gmt')
# obtain the index of up-regulated and down-regulated gene sets
index_up <- grep('-up',gmt)
index_down <- grep('-dn',gmt)
# transform the gmt file into gene sets. The gene set is a data frame, comprising three vectors: term (
gff_up <- gmt2geneSet( gmt[index_up] , termCol=c(1,2) , singleValue = 1 )
gff_down <- gmt2geneSet( gmt[index_down] , termCol=c(1,2) , singleValue = -1 )

# as following, combine the up-regulated and down-regulated gene sets together, and use value of 1 and
# extract the drug names
term_up <- sapply( gff_up$term , function(x) gsub('-up','',x) )
term_down <- sapply( gff_down$term , function(x) gsub('-dn','',x) )
all(term_up==term_down)
```

```
## [1] TRUE
```

```
# combine the up-regulated and down-regulated gene names for each drug perturbation
geneset <- sapply( 1:nrow(gff_up) , function(i) paste(gff_up$geneset[i],gff_down$geneset[i],sep=',') )
# use 1 and -1 to indicate the direction of the up-regulated and down-regulated genes, respectively
value <- sapply( 1:nrow(gff_up) , function(i) paste(gff_up$value[i],gff_down$value[i],sep=',') )
# transform the gene set into matrix, where the row represents the gene, the column represents the drug
net1 <- geneSet2Net( term=term_up , geneset=geneset , value=value )
# transform the gene set into sparse matrix, where the row represents the gene, the column represents t
net2 <- geneSet2sparseMatrix( term=term_up , geneset=geneset , value=value )
tail(net1[,1:30])
```

```
##        Fluorouracil,drug:3639 Resveratrol,drug:3499 Citalopram,drug:3292
## ZYG11B                      0                     0                    0
## Zyx                         0                     0                    0
## ZYX                         0                     0                    0
## ZZEF1                       0                     0                    0
## Zzz3                        0                     0                    0
## ZZZ3                        0                     0                    0
##        Fluorouracil,drug:3638 Ethanol,drug:3475 Resveratrol,drug:3498
## ZYG11B                      0                0                     0
## Zyx                         0                0                     0
## ZYX                         0                0                     0
## ZZEF1                       0                0                     0
## Zzz3                        0                0                     0
## ZZZ3                        0                0                     0
##        Fluorouracil,drug:3637 Resveratrol,drug:3497
## ZYG11B                      0                    0
## Zyx                         0                    0
## ZYX                         0                    0
## ZZEF1                       0                    0
## Zzz3                        0                    0
## ZZZ3                        0                    0
##        Sodium arsenite,drug:3357 Vitamin c,drug:3510 Vitamin c,drug:3511
## ZYG11B                         0                  0                   0
## Zyx                            0                 -1                   0
## ZYX                            0                  0                   0
## ZZEF1                          0                  0                   0
## Zzz3                           0                  0                   0
## ZZZ3                           0                  0                   0
##        Perfluorooctanoic acid,drug:3516 Perfluorooctanoic acid,drug:3517
## ZYG11B                                0                                0
## Zyx                                   0                                0
## ZYX                                   0                                0
## ZZEF1                                 0                                0
## Zzz3                                  0                                0
## ZZZ3                                  0                                0
##        Formaldehyde,drug:3550 Tretinoin,drug:3634 Vitamin e,drug:3515
## ZYG11B                      0                   0                   0
## Zyx                         0                   0                   0
## ZYX                         0                   0                   0
## ZZEF1                       0                   0                   0
## Zzz3                        0                   0                   0
## ZZZ3                        0                   0                   0
##        Gefitinib,drug:3474 Methylprednisolone,drug:3668
## ZYG11B                   0                           0
## Zyx                      0                           0
## ZYX                      0                           0
## ZZEF1                    0                           0
## Zzz3                     0                           0
## ZZZ3                     0                           0
##        Methylprednisolone,drug:3669 Haloperidol,drug:2471
## ZYG11B                            0                    0
## Zyx                               0                    0
## ZYX                               0                    0
## ZZEF1                             0                    0
```

```
## Zzz3                              0                    0
## ZZZ3                              0                    0
##        Levetiracetam,drug:2671 Rosiglitazone,drug:2672
## ZYG11B                       0                       0
## Zyx                          0                       0
## ZYX                          0                       0
## ZZEF1                        0                       0
## Zzz3                         0                       0
## ZZZ3                         0                       0
##        Alitretinoin,drug:2673 Tretinoin,drug:3233 Tretinoin,drug:3232
## ZYG11B                       0                   0                   0
## Zyx                          0                   0                   0
## ZYX                          0                   0                   0
## ZZEF1                        0                   0                   0
## Zzz3                         0                   0                   0
## ZZZ3                         0                   0                   0
##        Tretinoin,drug:3231 Pristane,drug:3230 Tretinoin,drug:3237
## ZYG11B                   0                  0                   0
## Zyx                      0                  0                   0
## ZYX                      0                  0                   0
## ZZEF1                    0                  0                   0
## Zzz3                     0                  0                   0
## ZZZ3                     0                  0                   0
##        Hypochlorous acid,drug:3198 Methoxychlor,drug:3235
## ZYG11B                           0                      0
## Zyx                              0                      0
## ZYX                              0                      0
## ZZEF1                            0                      0
## Zzz3                             0                      0
## ZZZ3                             0                      0
```

```
tail(net2[,1:30])
```

```
## 6 x 30 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 30 column names 'Fluorouracil,drug:3639', 'Resveratrol,drug:3499', 'Citalopram,drug

##
## ZYG11B . . . . . .  . . . . . . . . . . . . . . . . . . . . . . .
## Zyx    . . . . . . . -1 . . . . . . . . . . . . . . . . . . . . . .
## ZYX    . . . . . .  . . . . . . . . . . . . . . . . . . . . . . . .
## ZZEF1  . . . . . .  . . . . . . . . . . . . . . . . . . . . . . . .
## Zzz3   . . . . . .  . . . . . . . . . . . . . . . . . . . . . . . .
## ZZZ3   . . . . . .  . . . . . . . . . . . . . . . . . . . . . . . .
```

```
# the size of sparse matrix is much smaller than the matrix
format( object.size(net1), units = "auto")
```

```
## [1] "197.4 Mb"
```

```
format( object.size(net2), units = "auto")
```

```
## [1] "7.7 Mb"
```

4. Structure of MetaXcan output

MetaXcan integrates GWAS summary result with eQTL information to map trait-associated genes. See https://github.com/hakyimlab/MetaXcan. It provides a novel way to aggregate the multiple markers within each gene, address the long-range regulation, and adjust bias from gene boundaries and gene size. We use MetaXcan to impute the complex-trait-associated differential gene expression from eQTL summary and GWAS summary datasets. MetaXcan imputes ~10,000 genes with high quality prediction in most tissues. The training dataset for the expression prediction or 1000 Genomes was used as reference population to address the LD structure (covariance) of markers. Users can also specify their own genotype data to address the LD structure. The eQTL summary data was pre-calculated from large gene expression studies, such as the Genotype-Tissue Expression Project (GTEx; a comprehensive set of tissues from of ~20,000 samples) (Lonsdale, et al., 2013) and Depression Genes and Networks (DGN; 922 whole-blood samples) (Battle, et al., 2014). So, users only need to provide the GWAS summary data to estimate the genetically regulated gene expression.

We take the heart disease GWAS data as an example. We run the MetaXcan on it based on DGN eQTL database and 1000 genome as covariance.

```
data(heart.metaXcan)
head(heart.metaXcan)
```

```
##              gene gene_name    zscore effect_size       pvalue
## 1 ENSG00000082258     CCNT2  5.293351   14.135406 1.200952e-07
## 2 ENSG00000152128   TMEM163  4.196239    7.447722 2.713842e-05
## 3 ENSG00000047579    DTNBP1 -4.017977   -2.120019 5.869990e-05
## 4 ENSG00000147457     CHMP7 -3.868460   -2.671272 1.095251e-04
## 5 ENSG00000163947   ARHGEF3  3.755494   32.994931 1.730000e-04
## 6 ENSG00000163563      MNDA -3.697886  -22.400910 2.174021e-04
##          var_g pred_perf_r2 pred_perf_pval pred_perf_qval n_snps_used
## 1 8.663595e-04  0.005082875   3.041603e-02   4.118107e-02           3
## 2 1.904250e-03  0.107549597   1.475641e-24   4.854061e-24           7
## 3 1.859056e-02  0.339299671   7.174712e-85   7.493904e-84          16
## 4 1.060915e-02  0.040070269   8.670184e-10   1.772664e-09           9
## 5 6.451472e-05  0.013210923   4.706739e-04   7.270990e-04           1
## 6 1.411039e-04  0.091274313   6.520400e-21   1.943870e-20           5
##   n_snps_in_cov n_snps_in_model
## 1            10              10
## 2            55              55
## 3            96              97
## 4            36              36
## 5             7               7
## 6            28              28
```

Each row is a gene's association result:

- `gene`: a gene's id: as listed in the Tissue Transcriptome model
- `gene_name`: gene name as listed by the Transcriptome Model, generally extracted from Genquant
- `zscore`: MetaXcan's association result for the gene

- `effect_size`: MetaXcan's association effect size for the gene
- `pvalue`: P-value of the aforementioned statistic.
- `pred_perf_r2`: R2 of tissue model's correlation to gene's measured transcriptome
- `pred_perf_pval`: pval of tissue model's correlation to gene's measured transcriptome
- `pred_perf_qval`: qval of tissue model's correlation to gene's measured transcriptome
- `n_snps_used`: number of snps from GWAS that got used in MetaXcan analysis
- `n_snps_in_cov`: number of snps in the covariance matrix
- `n_snps_in_model`: number of snps in the model
- `var_g`: variance of the gene expression

We use the linear regression model to build a threshold-free gene set enrichment test, checking whether genes are significantly differentially expressed in a given gene set, compared to the background. In the regression model, we regress the imputed Z-score of gene differential expression on the gene sets. However, the gene expression cannot be perfectly predicted using genotype, indicated by the prediction R^2, and moreover, not all SNPs in the prediction model are also in user's dataset. In order to take into account such two factors, we use both the multiplication of prediction R^2 and fraction of imputation-used SNPs as weights, building a weighted linear regression model.

```
gene = heart.metaXcan$gene_name
# extract the imputed Z-score of gene differential expression, which follows the normal distribution
fc <- heart.metaXcan$zscore
# use the prediction R^2 and fraction of imputation-used SNPs as weights
usedFrac <- heart.metaXcan$n_snps_used / heart.metaXcan$n_snps_in_cov
r2 <- heart.metaXcan$pred_perf_r2
weights <- usedFrac*r2
# build a new data frame for the following weighted linear regression-based enrichment analysis
data <- data.frame(gene,fc,weights)
head(data)
```

```
##      gene        fc     weights
## 1   CCNT2  5.293351 0.001524862
## 2 TMEM163  4.196239 0.013688131
## 3  DTNBP1 -4.017977 0.056549945
## 4   CHMP7 -3.868460 0.010017567
## 5 ARHGEF3  3.755494 0.001887275
## 6    MNDA -3.697886 0.016298985
```

5. Gene set enrichment analysis

5.1 Gene set enrichment analysis using weighted single regression model

After obtaining the imputed gene differential expression and the weights, we build the weighted linear regression model to investigate the gene set enrichment. Permutation test was used to adjust the p values of the regression coefficients. We repeatedly shuffle the gene differential expression to obtain a global null distribution of no associated gene sets and calculate the empirical p value for each gene set. For the Single Gene Set Erichment Analysis (SGSEA), especially with many gene sets tested, a large number of weighted simple linear regression model would be interrogated. To improve the efficiency, we use weighted Pearson correlation to rank the significance, which uses the same hypothesis statistic with the single weighted regression model. Furthermore, we expressed it in terms of large matrix inner product, substantially improving the time efficiency.

```r
# take MSigDB.KEGG.Pathway as an example
net <- MSigDB.KEGG.Pathway$net
# do intersection of genes between the user-provided imputed gene expression dataset and the gene sets
data2 <- orderedIntersect( x = data , by.x = data$gene , by.y = rownames(net)  )
net2 <- orderedIntersect( x = net , by.x = rownames(net) , by.y = data$gene  )
all( rownames(net2) == as.character(data2$gene) )
```

```
## [1] TRUE
```

```r
# the SGSEA.res1 uses the single weighted linear regression model, while SGSEA.res2 used the weighted P
system.time( SGSEA.res1 <- permutationSingleLm( fc=data2$fc , net=net2 , weights=data2$weights , num=10
```

```
## 0%.....10%.....20%.....30%.....40%.....50%.....60%.....70%.....80%.....90%.....100%.
```

```
##    user  system elapsed
##   87.08    0.11   87.31
```

```r
system.time( SGSEA.res2 <- permutationSingleLmMatrix( fc=data2$fc , net=net2 , weights=data2$weights , r
```

```
## 0%.....10%.....20%.....30%.....40%.....50%.....60%.....70%.....80%.....90%.....100%.
```

```
##    user  system elapsed
##    0.14    0.02    0.16
```

```r
head(SGSEA.res2)
```

```
##                                          term usedGenes observedCorr
## 146       KEGG_ADIPOCYTOKINE_SIGNALING_PATHWAY        45   0.09372666
## 163                     KEGG_COLORECTAL_CANCER        46   0.07538714
## 172                        KEGG_BLADDER_CANCER        30   0.04321495
## 173               KEGG_CHRONIC_MYELOID_LEUKEMIA        53   0.05079492
## 91  KEGG_PHOSPHATIDYLINOSITOL_SIGNALING_SYSTEM        49  -0.06870315
## 93                              KEGG_CELL_CYCLE        84   0.05551670
##     observedPval empiricalPval
## 146 4.065158e-07          0.00
## 163 4.671905e-05          0.00
## 172 1.971682e-02          0.00
## 173 6.122168e-03          0.00
## 91  2.076036e-04          0.01
## 93  2.732284e-03          0.01
```

5.2 Gene set enrichment analysis using weighted multiple regression model

A gene may function in multiple ways and thus appear multiple times in functional gene sets. In spite of reflecting the crosstalk between gene sets, such overlap may make the results of gene set analysis more difficult to interpret. To address the redundancy existing among gene sets, we build a weighted multiple linear regression model, taking into account all gene sets in one model. The redundancy of one gene set can be adjusted by considering all other gene sets as covariates.

```
MGSEA.res <- permutationMultiLm( fc=data2$fc , net=net2 , weights=data2$weights , num=100 )
```

```
## 0%.....10%.....20%.....30%.....40%.....50%.....60%.....70%.....80%.....90%.....100%.
```

```
head(MGSEA.res)
```

```
##                          term usedGenes   Estimate Std..Error   t.value
## 163   KEGG_COLORECTAL_CANCER        46  1.3031389  0.3202505  4.069123
## 167              KEGG_GLIOMA        45  1.0899196  0.4622590  2.357812
## 77          KEGG_SPLICEOSOME        71 -0.4032699  0.2013535 -2.002795
## 93           KEGG_CELL_CYCLE        84  0.4829422  0.2121533  2.276384
## 154  KEGG_PARKINSONS_DISEASE        72 -0.8815939  0.2437547 -3.616725
## 23  KEGG_TYROSINE_METABOLISM        22 -1.0226552  0.3075496 -3.325172
##     observedPval empiricalPval
## 163 4.853175e-05          0.00
## 167 1.845335e-02          0.03
## 77  4.529822e-02          0.04
## 93  2.290034e-02          0.04
## 154 3.037703e-04          0.04
## 23  8.953483e-04          0.05
```

5.3 One-step weightedGSEA

In user's convenience, we combine the above procedures together into one function `weightedGSEA()`. Based on the imputed differential gene expression, `weightedGSEA()` can check multiple classes of gene sets simultaneously and write out the enrichment analysis results. Users need to provide the same data with above, and specify the columns of gene names (geneCol), imputed differential gene expression (fcCol), weights (weightCol). Additionally, users specify the classes of gene sets of interest, the times of permutation (permutationNum) and the directory for saving the results (outputDir). By default, we only do SGSEA. Users can define `MGSEAthres` to perform MGMEA for those with less than `MGSEAthres` gene sets.

```
# import packages and prepare data as above
library(GIGSEA)
library(Matrix)
data(heart.metaXcan)
gene = heart.metaXcan$gene_name
fc <- heart.metaXcan$zscore
usedFrac <- heart.metaXcan$n_snps_used / heart.metaXcan$n_snps_in_cov
r2 <- heart.metaXcan$pred_perf_r2
weights <- usedFrac*r2
data <- data.frame(gene,fc,weights)

# run one-step GIGSEA
weightedGSEA(data, geneCol='gene', fcCol='fc', weightCol= 'weights', geneSet=c("MSigDB.KEGG.Pathway","Fa
```

```
## creating  ./GIGSEA
##
## Checking MSigDB.KEGG.Pathway ...
## --> performing SGSEA ...
## 0%.....10%.....20%.....30%.....40%.....50%.....60%.....70%.....80%.....90%.....100%.
## Checking Fantom5.TF ...
```

```
## --> performing SGSEA ...
## 0%.....10%.....20%.....30%.....40%.....50%.....60%.....70%.....80%.....90%.....100%.
## Checking TargetScan.miRNA ...
## --> performing SGSEA ...
## 0%.....10%.....20%.....30%.....40%.....50%.....60%.....70%.....80%.....90%.....100%.
## Checking GO ...
## --> performing SGSEA ...
## 0%.....10%.....20%.....30%.....40%.....50%.....60%.....70%.....80%.....90%.....100%.
## Checking LINCS.CMap.drug ...
## --> performing SGSEA ...
## 0%.....10%.....20%.....30%.....40%.....50%.....60%.....70%.....80%.....90%.....100%.
```

```
dir("./GIGSEA")
```

```
## [1] "Fantom5.TF.SGSEA.txt"          "GO.SGSEA.txt"
## [3] "LINCS.CMap.drug.SGSEA.txt"     "MSigDB.KEGG.Pathway.SGSEA.txt"
## [5] "TargetScan.miRNA.SGSEA.txt"
```