snpGeneSets: an *R* Package for Genome-wide Study Annotation

Hao Mei and Lianna Li

Version 2.0, 05-01-2018

Contents

1. Introduction

Genome-wide studies (GWS) of SNP association and differential gene expression have generated abundant results, and the next-generation sequencing technology has further boosted the increasing. Effective interpretation of these results and understanding of the genetic effects often require massive annotation and post-analysis over genome, which is however a computationally challenging task. To address this challenge, the *snpGeneSets* package is developed to simplify post-annotation and analysis of GWS results[1]. The package integrates local copies of parsed NCBI dbSNP [2] and Entrez Gene [3] databases based on two recent genome builds of GRCh37/hg19 and GRCh38/hg38 and MSigDB gene sets V6.1 [4], and provides three types of main annotations: 1) genomic mapping annotation for SNPs and genes, and function annotation for gene sets; 2) bidirectional mapping relation between SNPs and genes, and between genes and gene sets; and 3) gene effect measures from SNP associations and enrichment analysis-based annotations for identifying function pathways from genes. The auxiliary functions are also provided to facilitate the annotation and analysis for genome-wide study. The package structures and components are summarized at the Figure 1.

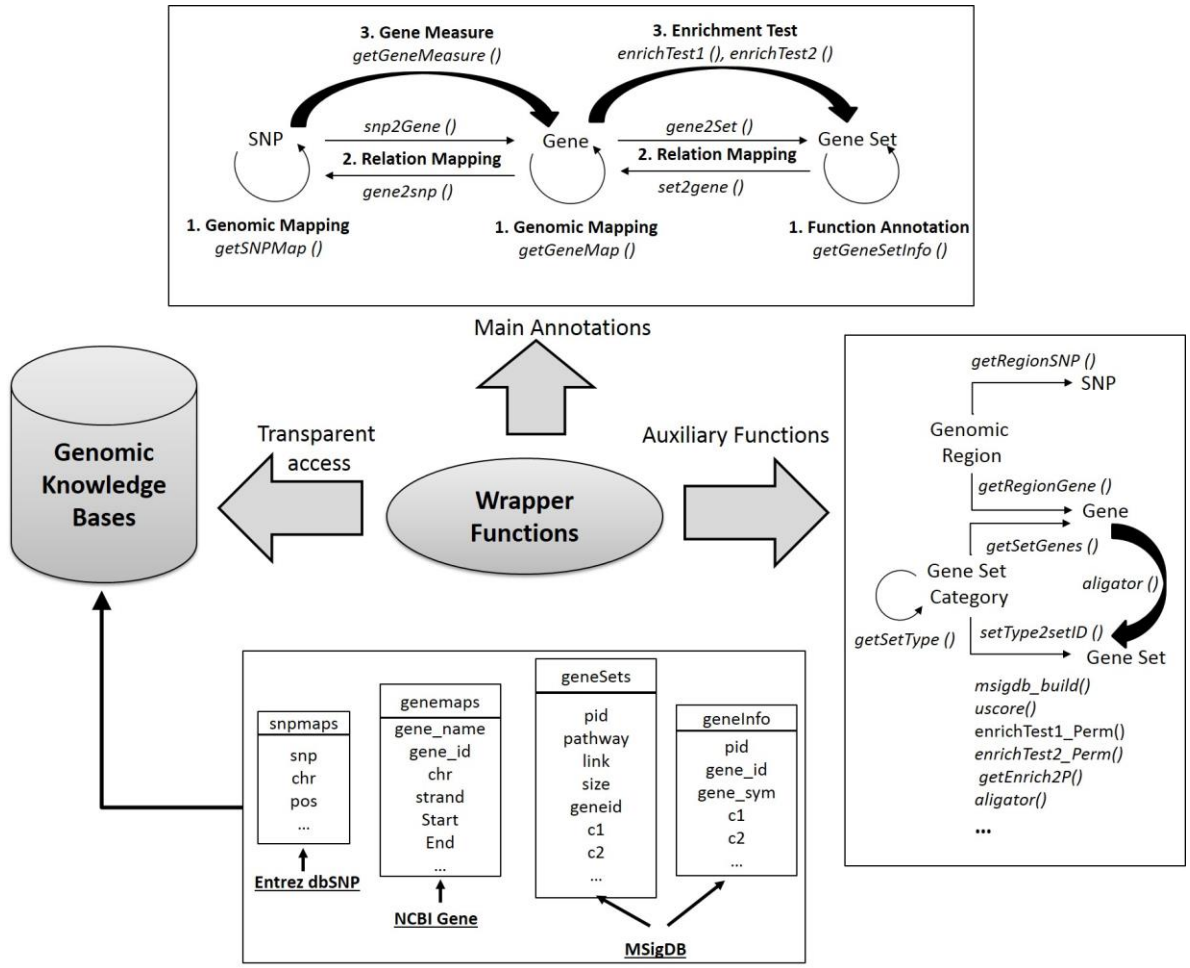

Figure 1. Systematic structures and annotation components of the *snpGeneSets* package

2. Installation

Before the installation of new version, an old version of *snpGeneSets* can be removed by system command:

```
R CMD REMOVE snpGeneSets
```

Or by the command below under R environment.

```
>remove.packages("snpGeneSets")
```

2.1 GitHub installation

The package can be directly installed from the GitHub. The R package of "devtools" need to be installed first if it is not. Under R, run the command below:

```
> install.packages("devtools")
```

Then run the following commands to install snpGeneSets:

```
> library(devtools)

> install_github("meiprojects/snpGeneSets")
```

2.2 Download and install locally

The package source file of *snpGeneSets_2.0.tar.gz* and windows binary file of *snpGeneSets_2.0.zip* can be downloaded from https://www.umc.edu/biostats_software/ .

Installation from the source file of *snpGeneSets_2.0.tar.gz* can be completed through the system command:

```
R CMD INSTALL snpGeneSets_2.0.tar.gz
```

Installation from the binary file of *snpGeneSets_2.0.zip* for Windows can be completed through the GUI interface: "Packages"→" Install package(s) from local zip files..." .

Notes: The package is integrated with parsed NCBI dbSNP 138 (GRCh37/hg19) and 142 (GRCh38/hg38) [2], Entrez gene 105 (GRCh37/hg19) and 106 (GRCh38/hg38) [3]. The installation will automatically download and install the integrated databases based on GRCh37 and GRCh38, which requires high-speed internet access. The SNP annotation data based on GRCh37/hg19 includes common variants with unique position from NCBI dbSNP and those low-frequency variants from *1000* Genome project. The SNP annotation data based on GRCh38/hg38 includes common variants with

unique position from NCBI dbSNP, but does not have low-frequency variants from 1000 Genome project.

3. Installation of MSigDB gene sets

If the installation does not find pathway data of MSigDB, then manual installation of MSigDB database will be required. The MSigDB data can be downloaded at http://www.broadinstitute.org/gsea/downloads.jsp .

To install the MSigDB 6.1, the zipped file of *msigdb_v6.1_files_to_download_locally.zip* ("MSigDB version 6.1 - zipped msigdb.xml, gmt and chip files") has to be downloaded and extracted locally. All required *gmt* files can be founded at the extracted directory of "msigdb_v6.1_GMTs". The installation can be completed by function *msigdb_build*:

```
> library(snpGeneSets)
> msigdb_build(gmt_dir="~/tmp/msigdb_v6.1_files_to_download_locally/msigdb_v6.1_GMTs")
```

The argument of *gmt_dir* shows where all the extracted *gmt* files can be found. The function will parse all *gmt* files and build the integrated database.

4. Identification of SNP and gene map positions from an updated reference genome

Some GWAS of SNP associations may be based on an old reference genome build, e.g. NCBI36. The *snpGeneSets* can quickly convert old map positions for a large number of GWAS SNPs to updated positions based on a recent genome build, GRCh37 or GRCh38, simultaneously by function of *getSNPMap()*. Map positions of GRCh37 and GRCh38 for genes can be identified by function of *getGeneMap()*.

The *snpGeneSets* comes with two GWS (Genome-wide study) data, the type 2 diabetes-genome wide association study (T2D-GWAS) and the type 2 diabetes-genome wide expression study (T2D-GWES). The T2D-GWAS contains GWAS SNP association for T2D in Finnish population from the dbGaP (Analysis ID: pha002839) [5], and T2D-GWES presents differential expression p-values at pancreases of *10* control and *10* T2D human subjects [6], which we obtained by analysis of GEO expression data (GDS3782) using the linear models with empirical Bayes adjusting method [7].

4.1 *Example: Identification of T2D-GWAS SNP map positions*

```
> library(snpGeneSets)
```

```
> data("T2DGWAS")

> class(T2DGWAS)
[1] "data.frame"

> dim(T2DGWAS)
[1] 306368     2

> head(T2DGWAS)
         snp          p
1  rs4649592 0.95773144
2 rs41332249 0.98747972
3  rs1079109 0.42112743
4  rs3934834 0.38536813
5  rs3737728 0.64311534
6  rs6687776 0.08061468
```

The T2DGWAS results can be loaded into R by the function *data()*. There are total *306,368* SNPs with GWAS association p-values available. Identifiers of these SNPs and their map positions are obtained based on old genome build. Genomic map positions of these SNPs based on a recent map build can be obtained simultaneously by function of *getSNPMap()* and the reference genome build can be specified by parameter *GRCh=37* (in default) or *GRCh=38*.

```
> snpMapAnn<- getSNPMap(T2DGWAS$snp)

> snpMapAnn38<- getSNPMap(T2DGWAS$snp, GRCh=38)
```

Depending on the computer performance, the map annotation may take up to 1 minute for completing the process.

```
> names(snpMapAnn)
[1] "rsid_map" "other"
```

The returned result variables of *snpMapAnn* and *snpMapAnn38* are a data type of list and it contains two components, a data frame of *'rsid_map'* and a character vector of *'other'*. The *'rsid_map'* contains all SNP identifiers that can be found for their genomic positions. The *'other'* contains the SNP identifiers that cannot be found for map positions.

```
> class(snpMapAnn$rsid_map)
[1] "data.frame"

> dim(snpMapAnn$rsid_map)
[1] 306252     3

> dim(snpMapAnn38$rsid_map)
[1] 306045     3

> head(snpMapAnn$rsid_map)
```

```
  chr       pos        snp
1   4  21618674 rs10000010
2   4  95733906 rs10000023
3   4 103374154 rs10000030
4   2 237752054  rs1000007
5   4  21895517 rs10000092
6   4 157574035 rs10000121
```

```
> head(snpMapAnn38$rsid_map)
  chr       pos        snp
1   4  21617051 rs10000010
2   4  94812755 rs10000023
3   4 102452997 rs10000030
4   2 236843411  rs1000007
5   4  21893894 rs10000092
6   4 156652883 rs10000121
```

```
> class(snpMapAnn$other)
[1] "character"
```

```
> length(snpMapAnn$other)
[1] 116
```

```
> length(snpMapAnn38$other)
[1] 323
```

```
> head(snpMapAnn$other)
[1] "rs4649592"  "rs41332249" "rs1079109"  "rs7549320"  "rs7412106"
[6] "rs12619064"
```

```
> head(snpMapAnn38$other)
[1] "rs4649592"  "rs41332249" "rs1079109"  "rs41511844" "rs17559902"
[6] "rs4297265"
```

The mapping annotation based on GRCh37 showed *306,252* SNPs have been identified for genomic map positions and *116* SNPs cannot be identified, which may be due to alteration or obsolete of these rs ids. For reference genome GRCh38, total *306,045* SNPs have been identified, but *323 SNPs* are not.

4.2 *Example: Identification of gene map positions for T2D-GWES*

```
> library("snpGeneSets")
```

```
> data("T2DGWES")
```

```
> class(T2DExpression)
[1] "data.frame"
```

```
> dim(T2DExpression)
[1] 20185     3
```

```
> head(T2DExpression)
```

```
       symbol gene_id            p
9199    MDFIC   29969 6.399265e-07
3613   PPP2CB    5516 1.209549e-06
3503    FXYD3    5349 1.955109e-06
2292   IGFBP3    3486 2.853953e-06
6984   UNC13B   10497 2.876275e-06
11673   RRAGD   58528 5.047322e-06
```

The T2D-GWES data can be loaded by the *data("T2DGWES")* command, and the results of *T2DExpression* variable are stored as a data frame that contains differential expression p-values for *20,185* genes. The *T2DExpression* contains gene symbol ('symbol'), its Entrez gene ID ('gene_id') and the differential expression p-value ('p'). Map positions of T2D-GWES genes can be identified by *getGeneMap()* function with the reference genome specified at parameter of *GRCh* that is 37 in default.

```
> geneMapAnn<-getGeneMap(T2DExpression$gene_id)

> names(geneMapAnn)
[1] "gene_map" "other"

> class(geneMapAnn$gene_map)
[1] "data.frame"

> dim(geneMapAnn$gene_map)
[1] 19299     6

> head(geneMapAnn$gene_map)
  chr     start       end strand gene_name gene_id
1  19  58858172  58864865      -      A1BG       1
2  12   9220304   9268558      -       A2M       2
3   8  18027971  18081198      +      NAT1       9
4   8  18248755  18258723      +      NAT2      10
5  14  95078639  95090395      +  SERPINA3      12
6   3 151531769 151546276      +     AADAC      13

> class(geneMapAnn$other)
[1] "numeric"

> length(geneMapAnn$other)
[1] 920

> head(geneMapAnn$other)
[1] 100130051 100129513      5558    727770 100132999 100134017

> geneMapAnn38<-getGeneMap(T2DExpression$gene_id, GRCh=38)

> dim(geneMapAnn38$gene_map)
[1] 19283     6

> head(geneMapAnn38$gene_map)
  chr     start       end strand gene_name gene_id
1  19  58346806  58353499      -      A1BG       1
2  12   9067708   9115962      -       A2M       2
3   8  18170462  18223689      +      NAT1       9
4   8  18391245  18401213      +      NAT2      10
5  14  94612377  94624053      +  SERPINA3      12
6   3 151813974 151828488      +     AADAC      13

> length(geneMapAnn38$other)
```

```
[1] 927

> head(geneMapAnn38$other)
[1] 100130051 100129513    727770 100132999 100134017    730184
```

The returned annotation variables of *geneMapAnn* and *geneMapAnn38* are a list with two components: *"gene_map"* and *"other"*. The *"gene_map"* is a data frame with *19,299* genes from GRCh37 and *19,283* genes from GRCh38, and the map position of a gene is defined by chromosome ('chr'), transcription start position ('start') and transcription termination position ('end'). The *"other"* component is a numeric vector and it contains *920* Entrez gene IDs for GRCh37 and *927* genes for GRCh38 that are not identified for their map positions.

The *getGeneMap()* function has a second argument of logical variable, *isGeneID,* that determines if the searched genes are character vector of gene symbol or numerical vector of Entrez Gene ID.

5. Two-way mapping between SNP, gene and pathway

5.1 Mapping between SNP and Gene

Fast mapping of GWAS SNPs to genes is important for interpreting and understanding GWAS results. *snp2Gene* identifies genes spanning the target SNPs, based on user-defined gene boundary and SNP positions.

```
>  library(snpGeneSets)
>  data(T2DGWAS)
> T2DGWAS[T2DGWAS$p==min(T2DGWAS$p),]
          snp             p
70765  rs886374      2.37573e-06
```

The top SNP hit of the T2D-GWAS is the *rs886374* with association p-value of *2.4E-06.* We can apply the *snp2Gene()* function to obtain the genes that cover this SNP based on either GRCh37 (in default) or GRCh38.

```
> rs886374_map<-getSNPMap("rs886374")$rsid_map

> rs886374_map
       chr      pos            snp
1      4        7738369        rs886374

> rs886374_gene<-snp2Gene("rs886374")

> rs886374_gene
$map
       snp            gene_id
1      rs886374       57537
```

```
$other
character(0)

> getGeneMap(57537)$gene_map
      chr    start      end        strand  gene_name   gene_id
1     4      7194374    7744564    +       SORCS2      57537

> rs886374_map38<-getSNPMap("rs886374", GRCh=38)$rsid_map

> rs886374_map38
       chr    pos        snp

1      4      7736642    rs886374

> rs886374_gene38<-snp2Gene("rs886374", GRCh=38)

> rs886374_gene38
$map
    snp gene_id
1 rs886374   57537

$other
character(0)

> getGeneMap(57537,GRCh=38)$gene_map
      chr    start      end        strand  gene_name   gene_id
1     4      7192647    7742837    +       SORCS2      57537
```

The *snp2Gene()* function requires a data frame of SNP map including *'chr'*, *'pos'* and *'snp'* as the input to perform the SNP-Gene mapping. We first obtained the data frame of *rs886374_map* (GRCh37) and *rs886374_map38* (GRCh38) by *getSNPMap()* function. The *rs886374_gene* and *rs886374_gene38* returned by *snp2Gene()* function showed that SNP *rs886374* mapped to Entrez gene ID *57537*. The *getGeneMap()* function showed that the gene ID of *57537* is *SORCS2*, which is at Chromosome 4 from *7,194,374* to *7,744,564* bp for GRCh37 and from *7,192,647* to *7,742,837* bp for GRCh38.

The *snp2Gene()* function can be applied to map all T2D-GWAS SNPs to genes simultaneously. The mapping may take >1 hour depending on the number of GWAS SNPs. To speed the process, GWAS SNPs can be split to map *100,000* SNPs every time (about a few minutes for mapping).

```
> snpMapAnn_100k<- getSNPMap(T2DGWAS$snp[1:100000])

> snpGeneMapAnn_100k<-snp2Gene(snpMapAnn_100k$rsid_map$snp)

> names(snpGeneMapAnn_100k)
[1] "map"   "other"

> class(snpGeneMapAnn_100k$map)
```

```
[1] "data.frame"

> dim(snpGeneMapAnn_100k$map)
[1] 53823     2

> head(snpGeneMapAnn_100k$map)
          snp gene_id
1 rs10000010   80333
2 rs10000023     658
3 rs10000092   80333
4 rs10000169   57619
5 rs10000300   54502
6 rs10000432   57205

> length(unique((snpGeneMapAnn_100k$map$gene_id)))
[1] 7159

> class(snpGeneMapAnn_100k$other)
[1] "character"

> length(snpGeneMapAnn_100k$other)
[1] 49611

> head(snpGeneMapAnn_100k$other)
[1] "rs10000030" "rs1000007"  "rs10000121" "rs10000141" "rs1000016"
[6] "rs10000272"

> snpGeneMapAnn38_100k<-snp2Gene(snpMapAnn_100k$rsid_map$snp, GRCh=38)

> head(snpGeneMapAnn38_100k$map)

          snp gene_id
1 rs10000010   80333
2 rs10000023     658
3 rs10000092   80333
4 rs10000169   57619
5 rs10000300   54502
6 rs10000432   57205

> length(unique((snpGeneMapAnn38_100k$map$gene_id)))
[1] 7453

> length(snpGeneMapAnn38_100k$other)
[1] 48896

> head(snpGeneMapAnn38_100k$other)
[1] "rs10178538" "rs10496708" "rs10497199" "rs10515088" "rs10929316"
[6] "rs11100863"
```

The *snp2Gene()* function returned the SNP-gene mapping annotation results of *snpGeneMapAnn_100k* (GRCh37) and *snpGeneMapAnn38_100k* (GRCh38) for *100,000* GWAS SNPs. The *snpGeneMapAnn_100k* and *snpGeneMapAnn38_100k* are a list with two components: a data frame of *"map"* and a character vector of *"other"*. The *snpGeneMapAnn_100k$map* showed that *53,823* SNPs were successfully mapped to *7,159* genes and *snpGeneMapAnn_100k$other* indicated that *49,611* SNPs are out of gene boundary.

The gene boundary is defined by two arguments, *'up'* for the upstream region and *'down'* for the downstream region with default value of *2,000* bp for both. The mapping results of all SNPs can be directly found at *'snpGeneMap'* variable from *"T2DGWAS"* data.

In contrast to the *snp2Gene()* function, the *getRegionSNP()* function performs the reverse mapping and it shows annotated common SNPs spanned by the target gene or genomic region. The *getRegionSNP()* function takes a data frame including *'chr'*, *'start'* and *'end'* as the input.

```
> chr=c("14","1","18","16","16")
> start=c(78786077, 213910494, 57850422, 53813450, 53820527)
> end=start+1000
> regionDF=data.frame(chr=chr, start=start, end=end, stringsAsFactors=FALSE)
> regionSNPs<-getRegionSNP(regionDF)
> class(regionSNPs)
[1] "data.frame"
> dim(regionSNPs)
[1] 73  3
> head(regionSNPs)
  chr       pos         snp
1   1 213910494   rs1704198
2   1 213910566  rs10864067
3   1 213910585 rs141152028
4   1 213910675  rs79688837
5   1 213910826 rs182273155
6   1 213910983 rs186584814
>regionSNPs38<-getRegionSNP(regionDF, GRCh=38)
> dim(regionSNPs38)
[1] 24  3
> head(regionSNPs38)
  chr       pos         snp
1   1 213910556 rs75780458
2   1 213910610   rs853744
3   1 213910621 rs59335652
4   1 213910799   rs701894
5   1 213911041 rs12087028
6   1 213911081   rs919894
```

For the example above, the *getRegionSNP()* function returned the results to a data frame variable of *regionSNPs* for mapping annotations of *73* SNPs (GRCh37) and *regionSNPs38* for mapping annotations of *24* SNPs (GRCh38)

5.2 Mapping between Gene and Pathway

For a significant gene from GWAS or GWES, identification of its implicated pathways may shed light on novel gene function for disease genetics, and the mapping of gene to pathway is implemented by the *gene2Set()* function.

```
> library(snpGeneSets)
> data(T2DGWES)
> T2DExpression[T2DExpression$p==min(T2DExpression$p),]
      symbol      gene_id     p
9199  MDFIC       29969       6.399265e-07
```

The top gene of the T2D-GWES is MDFIC (Entrez gene ID: gene_id=*29969*) with p-value of *6.4E-07*, which acts as a transcriptional activator or repressor. The *gene2Set()* function can be applied to identify the MSigDB gene sets that include the MDFIC gene.

```
> gid29969_C2<-gene2Set(29969, setType=2)

> length(gid29969_C2)
[1] 45

> head(gid29969_C2)
[1] 2619 3471 3473 3518 3574 3619

> gid29969_C5<-gene2Set(29969, setType=14)

> length(gid29969_C5)
[1] 73

> head(gid29969_C5)
[1] 11857 11867 12098 12138 12192 12262
```

Application of *gene2Set()* function shows that *MDFIC* gene is the component gene of *45* MSigDB gene sets at the category of "C2: curated gene sets" and the component gene of *73* MSigDB gene sets at the category of "C5: GO gene sets". The category of gene sets can be specified by the argument of *'setType'*, which takes the value of category ID from *0* to *20*. The *21* gene-set categories and their description can be shown by *getSetType()* function. The Table 1 below summarizes all categories and their IDs, and *setType=2* and *setType=14* correspond to category of "C2: curated gene sets" and "C5: GO gene sets" respectively.

Table 1. Summary of 21 MSigDB gene-set categories

ID	symbol	name
0	c0	C0: all gene sets
1	c1	C1: positional gene sets
2	c2	C2: curated gene sets
3	c2_cgp	C2_CGP: chemical and genetic perturbations
4	c2_cp	C2_CP: Canonical pathways
5	c2_biocarta	C2_CP:BIOCARTA: BioCarta gene sets
6	c2_kegg	C2_CP:KEGG: KEGG gene sets

7	c2_reactome	C2_CP:REACTOME: Reactome gene sets
8	c3	C3: motif gene sets
9	c3_mir	C3_MIR: microRNA targets
10	c3_tft	C3_TFT: transcription factor targets
11	c4	C4: computational gene sets
12	c4_cgn	C4_CGN: cancer gene neighborhoods
13	c4_cm	C4_CM: cancer modules
14	c5	C5: GO gene sets
15	c5_bp	C5_BP: GO biological process
16	c5_cc	C5_CC: GO cellular component
17	c5_mf	C5_MF: GO molecular function
18	c6	C6: oncogenic signatures
19	c7	C7: immunologic signatures
20	hallmark	hallmark: hallmark gene sets

In contrast to *gene2Set()* function, the *getGeneSetInfo()* function identifies all member genes of a pathway and provides the mapping of pathway to genes. The *gid29969_C2* showed that the *MDFIC* gene is a member gene of gene-set *ID*=*2619 3471 3473 3518 3574 3619 …*. Description of the pathway can be shown by the *getGeneSetInfo()*.

The *getGeneSetInfo()* function below returns the results to *pid5029Ann* which contains 5 components: the *'setID'* of gene set identifier, the *'set_name'* of the gene set name, the *'set_link'* of the MSigDB web link describing the gene set, the *'set_type'* of the gene-set category including the gene set and the *'set_geneid'* of Entrez gene IDs belonging to the gene set (or pathway).

```
> pid2619Ann<-getGeneSetInfo(2619)

> names(pid2619Ann)
[1] "setID"     "set_name"   "set_link"   "set_type"   "set_geneid"

> pid2619Ann
```

```
$setID
[1] 2619

$set_name
[1] "PID_BETA_CATENIN_NUC_PATHWAY"

$set_link
[1] "http://www.broadinstitute.org/gsea/msigdb/cards/PID_BETA_CATENIN_NUC_PATHWAY"

$set_type
         c1          c2      c2_cgp       c2_cp c2_biocarta     c2_kegg
      FALSE        TRUE       FALSE        TRUE       FALSE       FALSE
c2_reactome          c3      c3_mir      c3_tft          c4      c4_cgn
      FALSE       FALSE       FALSE       FALSE       FALSE       FALSE
      c4_cm          c5       c5_bp       c5_cc       c5_mf          c6
      FALSE       FALSE       FALSE       FALSE       FALSE       FALSE
         c7    hallmark
      FALSE       FALSE

$set_geneid
 [1]  27121   1499   2033   3576    367   6591   7534   3848   8295   6925
[11]  57167   6597   2249   7091  10971  79718  83439   4617   6934   3725
[21]    113   7532   3619  22943  57680   7529   5308   1044  10856   8313
[31]  23145  80333   4286   3398   7531   9968   4318   6907  56998   2810
[41]  29969   1046   4313   3491   7015  10499   4513 170261  25776   1487
[51]  26959   4762   8454   6500   1029  23043   6932    166   7088   7089
[61]   3066    607    595   4656   1462  51176   7533   7514   4609   9314
[71]    999   1857   6862 389058    814   3065   8913    324  10642    894
```

6. Gene measures by SNP associations and U-score calculation for gene effects

A gene typically contains associations of multiple SNPs from a GWAS, and the *getGeneMeasure()* function provides four measures (*minP*, *2ndP*, *simP* and *fishP*) of the gene effect by summarizing SNP association *p*-values. *U*-score of a gene measure represents percentage of genome-wide genes with effects stronger than the given gene and it can be calculated by *uscore()* function.

For *K* SNPs mapped to a gene with GWAS p-values ($p_1, p_2, \ldots p_k$), the ordered p-value is defined as $p_{(1)} \leq p_{(2)} \leq \ldots \leq p_{(k)}$, where $p_{(1)}=\min\{p_1, p_2, \ldots p_k\}$ and $p_{(k)}=\max\{p_1, p_2, \ldots p_k\}$. Four gene measures are calculated respectively as $minP=p_{(1)}$, $2ndP=p_{(2)}$, $simP=\min_i\{Kp_{(i)}/i\}$ and $fishP=Pr\left(X \geq x = -2\sum_{i=1}^{K} log(p_i)\right) = \Psi(x)$, where Ψ is the chi-square distribution function with $df=2K$. Uniform score (*U*-score) is calculated as $U_i = \left(\sum_j I\left(M_j < M_i\right) + 0.5 \cdot \sum_j I\left(M_j = M_i\right)\right)/L$, where M_i is gene measure of the *i*-th gene and *L* is the total number of genes.

The *getGeneMeasure()* takes an arguments of *'snpGeneP'*. The *'snpGeneP'* is a data frame containing column of *'snp'* for rs id, column of *'gene_id'* for Entrez gene IDs spanning the *'snp'*, and column of *'p'* for SNP association p-value. At the following example, we derived *'snpGeneP'*data from the mapped genes

of T2DGWAS data. The mapping results of all T2DGWAS SNPs to genes can be directly found at *'snpGeneMap'* variable by loading *"T2DGWAS"* data.

```
> library(snpGeneSets)

> snpGeneP<-merge(snpGeneMap, T2DGWAS, all=FALSE)

> head(snpGeneP)
         snp gene_id         p
1 rs10000010   80333 0.2489708
2 rs10000023     658 0.2059405
3 rs10000092   80333 0.7070708
4 rs10000169   57619 0.5055075
5  rs1000022  171425 0.8532224
6 rs10000300   54502 0.5191723

> T2DGWASGene0<-getGeneMeasure(snpGeneP)

> head(T2DGWASGene0)
  gene_id       minp       sndp       simp      fishp
1       1 0.14992377 0.61819639 0.29984753 0.31313443
2       2 0.63210108 0.65196227 0.79051801 0.95773440
3       3 0.33866379 0.33866379 0.33866379 0.33866379
4       9 0.28229107 0.43147721 0.80126634 0.88787668
5      10 0.04538995 0.05860277 0.08790415 0.03175270
6      12 0.10190141 0.13136668 0.19705002 0.06615973

> minp_uscore<-uscore(T2DGWASGene$minp)

> head(minp_uscore)

[1] 0.3713382 0.8397428 0.6154937 0.5545215 0.1592300 0.2841735

> T2DGWASGene <- T2DGWASGene0

> for (ms in c("minp", "sndp", "simp", "fishp")) T2DGWASGene[[ms]]<-uscore(T2DGWASGene[[ms]])

> head(T2DGWASGene)
  gene_id      minp      sndp       simp      fishp
1       1 0.3713382 0.7145733 0.28760426 0.33140228
2       2 0.8397428 0.7440528 0.74729857 0.91304080
3       3 0.6154937 0.4576400 0.32357533 0.35102100
4       9 0.5545215 0.5503307 0.75900818 0.82988208
5      10 0.1592300 0.1039279 0.08642508 0.06933317
6      12 0.2841735 0.2105469 0.19181150 0.10803648
```

The *'snpGeneMap'*, *'snpGeneP'* and *'T2DGWASGene'* can be manually created as above. These variables are also pre-generated and automatically loaded with 'T2DGWAS' data. The *T2DGWASGene0* contains measures of *minP, 2ndP, simP* and *fishP* for every T2DGWAS gene. The *minp_uscore* is the uniform score for *minp* measure and U-score can also be similarly generated for other three measures. The *T2DGWASGene* contains U-scores for every gene measure.

We examined *9* genes that were previously reported to have associations with T2D, and their measures (*'gmeasure'*) and U-scores (*'gscore'*) were shown in the Figure 1.

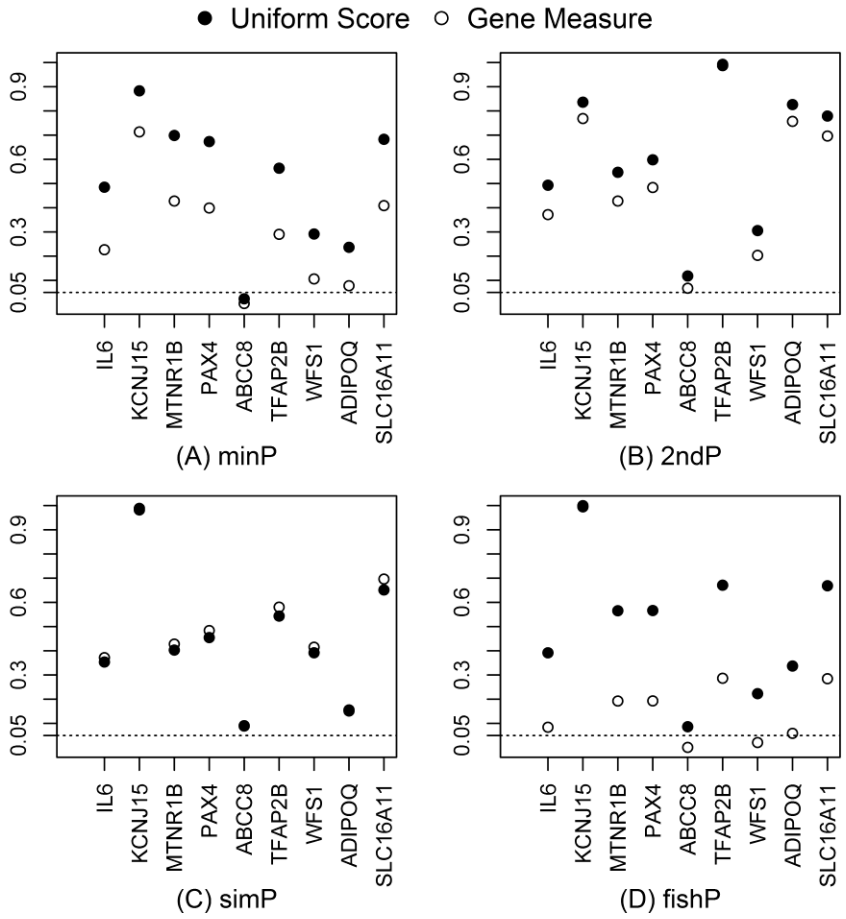

Figure 1. Gene measures and uniform scores of 9 T2D-GWAS genes

> genes<-c("IL6", "KCNJ15", "MTNR1B", "PAX4", "ABCC8", "TFAP2B", "WFS1", "ADIPOQ", "SLC16A11")

> genes<-getGeneMap(genes, FALSE)$gene_map[,c("gene_name","gene_id")]

> gmeasure<-merge(genes, T2DGWASGene0,all=FALSE)

> gmeasure

	gene_id	gene_name	minp	sndp	simp	fishp
1	3569	IL6	0.226553618	0.37160095	0.37160095	0.2925271
2	3772	KCNJ15	0.713388908	0.76853023	0.98839315	0.9999999
3	4544	MTNR1B	0.427681378	0.42768138	0.42768138	0.4276814
4	5078	PAX4	0.398935200	0.48357556	0.48357556	0.5103582
5	6833	ABCC8	0.004314517	0.06752818	0.09060486	0.2639783
6	7021	TFAP2B	0.290199840	0.98718080	0.58039968	0.6446044
7	7466	WFS1	0.105901871	0.20330600	0.41484572	0.2088555
8	9370	ADIPOQ	0.077799163	0.75677862	0.15559833	0.2256339
9	162515	SLC16A11	0.408816921	0.69633535	0.69633535	0.6423410

> gscore<-merge(genes, T2DGWASGene,all=FALSE)

> gscore

	gene_id	gene_name	minp	sndp	simp	fishp
1	3569	IL6	0.48488023	0.4930564	0.35365052	0.3135503
2	3772	KCNJ15	0.88304778	0.8361272	0.98206582	0.9996097
3	4544	MTNR1B	0.69865237	0.5464275	0.40295411	0.4220182
4	5078	PAX4	0.67338428	0.5981141	0.45464070	0.4887423
5	6833	ABCC8	0.02368626	0.1180205	0.08909569	0.2874605
6	7021	TFAP2B	0.56331402	0.9924607	0.54346933	0.6002095
7	7466	WFS1	0.29148691	0.3056617	0.39186080	0.2407864
8	9370	ADIPOQ	0.23651341	0.8262459	0.15142364	0.2544681
9	162515	SLC16A11	0.68275196	0.7788734	0.65144418	0.5979498

The calculation showed that only ABCC8 has all *4* gene measures and U-scores around or smaller than 0.05. The results presented that a stronger gene measure (i.e. smaller p-values) tends to have a smaller *U*-score. However, different gene measures for the same gene have varied *U*-scores, showing inconsistent measures of gene effects over genome. The calculation of *U*-score will unify these gene measures for comparability with the same interpretability. For example, the *minP*, *2ndP*, *simP* and *fishP* presented summary SNP association p-values of *0.004*, *0.068*, *0.091* and *0.0008* for ABCC8 gene respectively, and the corresponding *U*-scores indicated that *2.4%*, *11.8%*, *8.9%* and *28.7%* GWAS genes have stronger gene effects than ABCC8.

For T2D-GWES, differential expression p-value is used to directly measure gene effect and calculate *U*-scores of the selected 9 genes. The p-value of ABCC8 is *3.4E-04,* showing only 0.4% of genes over genome with stronger measured effect than the ABCC8.

```
>data(T2DGWES)

> escore<-uscore(T2DExpression$p)

> T2DExpression$us<-escore

> T2DExpression[T2DExpression$symbol %in% genes$gene_name,]
```

	symbol	gene_id	p	us
4560	ABCC8	6833	0.0003363277	0.004235819
2490	KCNJ15	3772	0.0268452946	0.091825613
3293	PAX4	5078	0.1437856302	0.270027248
16017	SLC16A11	162515	0.1471156303	0.273792420
2934	MTNR1B	4544	0.1978929899	0.331904880
4994	WFS1	7466	0.2134392425	0.346569235
2330	IL6	3569	0.3036799699	0.440351746
4685	TFAP2B	7021	0.4280112100	0.552068368
6062	ADIPOQ	9370	0.8169270613	0.865320783

7. Pathway Enrichment Analysis I of candidate genes

The type I analysis is a generalized pathway enrichment analysis that aims to identify gene sets enriched for a candidate list of genes. The list can be previously identified susceptibility genes or top genes from a GWAS or GWES. The analysis is performed by function *enrichTest1().*

7.1 ***Example: Enrichment analysis I of T2D-GWAS***

For *T2DGWAS* data, the top *5%* genes were selected as candidate genes by measures of *minP*, *2ndP*, *simP* and *fishP* respectively, and they were tested for pathway enrichment at *186* KEGG gene sets.

```
> library(snpGeneSets)
> data(T2DGWAS)
> topMinpGenes<-
T2DGWASGene[order(T2DGWASGene$minp),][1:trunc(nrow(T2DGWASGene)*0.05),"gene_id"]
> topsndpGenes<-
T2DGWASGene[order(T2DGWASGene$sndp),][1:trunc(nrow(T2DGWASGene)*0.05),"gene_id"]
> topsimpGenes<-
T2DGWASGene[order(T2DGWASGene$simp),][1:trunc(nrow(T2DGWASGene)*0.05),"gene_id"]
> topfishpGenes<-
T2DGWASGene[order(T2DGWASGene$fishp),][1:trunc(nrow(T2DGWASGene)*0.05),"gene_id"]
```

The *enrichTest1()* function takes an argument of *'genes'* for the candidate genes tested for pathway enrichment, and the argument *'setType'* takes the category ID that defines which category of gene sets will be tested for enrichment. For example, *setType=6* defines the KEGG gene-sets for enrichment test. Description of the category ID can be found at Table 1.

```
> minpGeneSets_KEGG<-enrichTest1(topMinpGenes,setType=6)
> names(minpGeneSets_KEGG)
[1] "enrich_test" "useGenes"    "nGenes"      "nTopGenes"   "setTypeInfo"
> head(minpGeneSets_KEGG$enrich_test)
   pid size genesSize       effect         sd       pval
1 1264   62         4  0.009635764 0.02892385 0.44513479
2 1265   32         0 -0.054880365 0.04026029 1.00000000
3 1266   27         2  0.019193709 0.04382985 0.44100234
4 1267   28         0 -0.054880365 0.04304006 1.00000000
5 1268   34         3  0.033354930 0.03905822 0.28581041
6 1269   26         5  0.137427328 0.04466478 0.01220623
> length(minpGeneSets_KEGG$useGenes)
[1] 289
> minpGeneSets_KEGG$nGenes
[1] 5266
> minpGeneSets_KEGG$nTopGenes
[1] 289
> minpGeneSets_KEGG$setTypeInfo
```

```
$id
[1] 6

$symbol
[1] "c2_kegg"

$name
[1] "C2_CP:KEGG: KEGG gene sets"

$description
[1] "Gene sets derived from the KEGG pathway database, http://www.genome.jp/kegg/pathway.html"
```

For the candidate genes selected by *minP* measure, the *enrichTest1()* function returned the results to the *minpGeneSets_KEGG* variable, which consists of a data frame of *"enrich_test"*, an integer vector of *"useGenes"*, a number of *"nGenes"*, a number of *"nTopGenes"* and a list of *"setTypeInfo"*. The *"enrich_test"* shows the enrichment test results for every gene set in the specified category defined by *setType*. The *"useGenes"* lists the effective candidate genes used for enrichment test. The *"nGenes"* is the total number of genes in the specified category and the *"nTopGenes"* is the number of effective candidate genes for enrichment test. The analysis above indicated that the KEGG category contains *5,266* genes, of which *289* genes are candidates, and the test aims to identify which gene set in the KEGG category is significantly enriched for the *289* candidate genes. The *"setTypeInfo"* presents description of the specified category, *KEGG*.

```
> minpGeneSets_KEGG$enrich_test[order(minpGeneSets_KEGG$enrich_test$pval),][1:10,]
     pid size genesSize     effect          sd         pval
184 1447   76         17 0.16880385 0.02612433 4.671353e-07
183 1446   85         14 0.10982552 0.02470259 1.820448e-04
185 1448   92         14 0.09729355 0.02374422 4.264065e-04
116 1379   75         11 0.09178630 0.02629791 2.364568e-03
117 1380  134         16 0.06452262 0.01967431 2.519538e-03
86  1349  267         26 0.04249791 0.01393787 2.801824e-03
138 1401   70         10 0.08797678 0.02722092 4.484553e-03
6   1269   26          5 0.13742733 0.04466478 1.220623e-02
147 1410   47          7 0.09405581 0.03322025 1.322871e-02
136 1399   70          9 0.07369106 0.02722092 1.355566e-02

> sndpGeneSets_KEGG<-enrichTest1(topsndpGenes,setType=6)
> sndpGeneSets_KEGG$enrich_test[order(sndpGeneSets_KEGG$enrich_test$pval),][1:10,]
     pid size genesSize     effect          sd         pval
184 1447   76         19 0.19796810 0.02547563 5.711156e-09
86  1349  267         33 0.07156360 0.01359177 2.120621e-06
185 1448   92         17 0.13275071 0.02315463 3.752066e-06
183 1446   85         15 0.12443869 0.02408919 2.516989e-05
113 1376  201         23 0.06239596 0.01566512 2.634448e-04
117 1380  134         17 0.07483377 0.01918577 5.161342e-04
166 1429   52          9 0.12104502 0.03079853 1.250869e-03
176 1439   54          9 0.11463476 0.03022281 1.650456e-03
167 1430   65         10 0.10181425 0.02754705 1.735843e-03
143 1406  101         13 0.07668097 0.02209892 2.036780e-03

> simpGeneSets_KEGG<-enrichTest1(topsimpGenes,setType=6)
> simpGeneSets_KEGG$enrich_test[order(simpGeneSets_KEGG$enrich_test$pval),][1:10,]
```

```
     pid size genesSize     effect         sd        pval
124 1387   71         8 0.07203800 0.02343303 0.007655692
82  1345   44         6 0.09572558 0.02976675 0.008139327
41  1304   25         4 0.11936194 0.03949005 0.017140010
149 1412   25         4 0.11936194 0.03949005 0.017140010
180 1443   38         5 0.09094089 0.03203066 0.017793365
22  1285   29         4 0.09729298 0.03666559 0.028376308
169 1432   29         4 0.09729298 0.03666559 0.028376308
148 1411   44         5 0.07299831 0.02976675 0.031676160
16  1279   31         4 0.08839420 0.03546311 0.035310622
86  1349  267        17 0.02303236 0.01208376 0.042651730
```

> fishpGeneSets_KEGG<-enrichTest1(topfishpGenes,setType=6)

> fishpGeneSets_KEGG$enrich_test[order(fishpGeneSets_KEGG$enrich_test$pval),][1:10,]

```
     pid size genesSize     effect         sd         pval
184 1447   76        21 0.21763748 0.02695883 1.068782e-09
183 1446   85        19 0.16485110 0.02549168 2.828406e-07
185 1448   92        17 0.12610429 0.02450270 1.876074e-05
86  1349  267        33 0.06491719 0.01438309 2.848419e-05
113 1376  201        27 0.07565004 0.01657715 3.661893e-05
136 1399   70        14 0.14132169 0.02809046 4.137306e-05
114 1377   84        15 0.11989311 0.02564296 8.810209e-05
176 1439   54        11 0.14502539 0.03198239 2.339888e-04
88  1351  178        23 0.07053517 0.01761562 2.527549e-04
117 1380  134        19 0.08311273 0.02030278 2.650890e-04
```

> getGeneSetInfo(1447)

```
$setID
[1] 1447

$set_name
[1] "KEGG_ARRHYTHMOGENIC_RIGHT_VENTRICULAR_CARDIOMYOPATHY_ARVC"

$set_link
[1] "http://www.broadinstitute.org/gsea/msigdb/cards/KEGG_ARRHYTHMOGENIC_RIGHT_VENTRICULAR_CARDIOMYOPATHY_ARVC"

$set_type
       c1          c2      c2_cgp       c2_cp c2_biocarta     c2_kegg
    FALSE        TRUE       FALSE        TRUE        FALSE        TRUE
c2_reactome          c3      c3_mir      c3_tft          c4      c4_cgn
    FALSE       FALSE       FALSE       FALSE       FALSE       FALSE
     c4_cm          c5       c5_bp       c5_cc       c5_mf          c6
    FALSE       FALSE       FALSE       FALSE       FALSE       FALSE
        c7    hallmark
    FALSE       FALSE

$set_geneid
 [1]       781       782 100418883       783       784       785       786
 [8]      3680     59283      6262      3728      4000     59285       488
[15]     59284      3676      3675      3674      3679     51176      3678
[22]      2010        81     22801      1824      1829        87      1756
[29]     55799       776      6546    646821       775       778      3690
[36]      9254       779      3691      3693      3694      3695      3685
[43]      3688      6932        71     93589      5318      1495      1000
[50]      6934      8515      6444      6445      6442      8516      6443
[57]     10369      1496      1605     10368      1499      3908      2697
[64]      3696        60      3655      3673      3672     27091     27092
[71]     83439      1832      1674     29119        88        89
```

> getGeneSetInfo(1387)

```
$setID
[1] 1387

$set_name
[1] "KEGG_RIG_I_LIKE_RECEPTOR_SIGNALING_PATHWAY"

$set_link
[1] "http://www.broadinstitute.org/gsea/msigdb/cards/KEGG_RIG_I_LIKE_RECEPTOR_SIGNALING_PATHWAY"

$set_type
         c1          c2      c2_cgp       c2_cp c2_biocarta     c2_kegg
      FALSE        TRUE       FALSE        TRUE       FALSE        TRUE
c2_reactome          c3      c3_mir      c3_tft          c4      c4_cgn
      FALSE       FALSE       FALSE       FALSE       FALSE       FALSE
      c4_cm          c5       c5_bp       c5_cc       c5_mf          c6
      FALSE       FALSE       FALSE       FALSE       FALSE       FALSE
         c7    hallmark
      FALSE       FALSE

$set_geneid
 [1]    843    841   3627   1147   7706   7124   3442   3441   3440   9140
[11]   9755   8772  80143   3439   8737   3576   5601   3593   5602   9636
[21]   5600   4214   3592   5603   3551   6885   3451  26007   3452   3443
[31]   3444   3445   3446  55593   3447   3448   3449   1654   9641   4790
[41]   4793   4792   3661  64135   8517   5599   1432   6300   5970   8653
[51] 338376   7189   7187   7186   3456  10010   1540   8717   9474  23586
[61]   5300  57506  56832  79132  29110 340061   3467  64343  79671   3665
[71]  54941
```

The four gene measures selected different candidate genes for enrichment test, which caused the pathway test results varied over the measures. The most enriched gene set was pathway of 'arrhythmogenic right ventricular cardiomyopathy' (*PID=1447*) for *minP*, *sndP* and *fishP*, and it was 'RIG-I-like receptor signaling pathway' (*PID=1387*) for *simP*. The pathway of *'1447'*, containing 76 genes, involves *17* candidate genes from *minP* (*effect=16.9%, p_e=4.67E-07*), *19* candidate genes from *2ndP* (*effect=19.8%, p_e=5.71E-9*) and *21* candidate genes from *fishP* (*effect=21.8%, p_e=1.07e-9*); and the pathway of *'1387'*, containing *71* genes, involves *8* candidate genes from *simP* (*effect=7.2%, p_e=7.66E-03*). All component genes for a particular gene set can be identified by the function *getGeneSetInfo()* function, e.g. *getGeneSetInfo(1477)* where 1477 is the pathway ID.

Different pathways may share common genes and these pathways will be dependent, potentially leading to an inflated type I error. To adjust for this issue and multiple testing, the *enrichTest1_Perm()* function applies a permutation-based test to obtain the adjusted p-value (*p_perm*) for pathway enrichment.

The most enriched gene set by every gene measure is prepared for permutation test and the R codes are as below:

```
> KEGG_rst<-
rbind(minpGeneSets_KEGG$enrich_test[minpGeneSets_KEGG$enrich_test$pval==min(minpGeneSets_KEGG$enrich_test$pval),],
sndpGeneSets_KEGG$enrich_test[sndpGeneSets_KEGG$enrich_test$pval==min(sndpGeneSets_KEGG$enrich_test$pval),],
```

```
simpGeneSets_KEGG$enrich_test[simpGeneSets_KEGG$enrich_test$pval==min(simpGeneSets_KEGG$enrich_test$pval),],
fishpGeneSets_KEGG$enrich_test[fishpGeneSets_KEGG$enrich_test$pval==min(fishpGeneSets_KEGG$enrich_test$pval),])
> KEGG_rst<-cbind(measure=c("minp","2ndp","simp","fishp"),
                  topGenes=c(minpGeneSets_KEGG$nTopGenes,sndpGeneSets_KEGG$nTopGenes,
                  simpGeneSets_KEGG$nTopGenes,fishpGeneSets_KEGG$nTopGenes), KEGG_rst)
> colnames(KEGG_rst)<-c("measure", "topGenes","pid","size","setTopGenes","effect","sd","p")
```

Results of the most enriched pathway for every gene measure were saved to *KEGG_rst* variable and the results were shown below:

```
> KEGG_rst
     measure topGenes  pid size setTopGenes    effect         sd           p
184     minp      289 1447   76          17 0.1688038 0.02612433 4.671353e-07
1841    2ndp      274 1447   76          19 0.1979681 0.02547563 5.711156e-09
124     simp      214 1387   71           8 0.0720380 0.02343303 7.655692e-03
1842   fishp      309 1447   76          21 0.2176375 0.02695883 1.068782e-09
```

The *enrichTest1_Perm()* function was applied to get permutation distribution table for calculating permutation p-value of the most enriched pathway. The argument of *geneSize* defines the number of effective candidate genes for enrichment test. The argument of *setType* defines the category of gene sets for permutation adjusting. The argument of *times* specifies the number of permutations for generating distribution table and the argument of *seed* assigns a random seed for permutation.

```
> minp_dist=enrichTest1_Perm(genesSize=KEGG_rst[1,"topGenes"], setType=6,times=1000, seed=1)
> sndp_dist=enrichTest1_Perm(genesSize=KEGG_rst[2,"topGenes"], setType=6,times=1000, seed=1)
> simp_dist=enrichTest1_Perm(genesSize=KEGG_rst[3,"topGenes"], setType=6,times=1000, seed=1)
> fishp_dist=enrichTest1_Perm(genesSize=KEGG_rst[4,"topGenes"], setType=6,times=1000, seed=1)
```

The minimum p-value of the KEGG category (*setType=6*) was extract to construct the distribution table and calculate permutation p-value (*p_perm*)

```
> minp_min=apply(minp_dist,2,min)
> sndp_min=apply(sndp_dist,2,min)
> simp_min=apply(simp_dist,2,min)
> fishp_min=apply(fishp_dist,2,min)
> KEGG_rst$p_perm<-c(sum(minp_min<=KEGG_rst[1,"p"]),sum(sndp_min<=KEGG_rst[2,"p"]),
sum(simp_min<=KEGG_rst[3,"p"]),sum(fishp_min<=KEGG_rst[4,"p"]))/1000
```

```
> KEGG_rst
```

The results were summarized at Table 2. The gene set of *'2901'* has *p_perm<1e-03* for enrichment of candidate genes from *minP*, *2ndP* and *fishP*, and the gene set of '*2799*' has *p_perm=0.463* for enrichment of candidate genes from *simP*

Table 2. The most enriched KEGG pathway of T2D-GWAS by enrichment analysis I

Measure	Genes	PID	size	setGenes	effect(%)	sd(%)	p_e	p_perm
minP	289	1447	76	17	16.9	2.6	4.67E-07	<1e-03
2ndP	274	1447	76	19	19.8	2.5	5.71E-09	<1e-03
simP	214	1387	71	8	7.20	2.3	7.65E-03	0.463
fishP	309	1447	76	21	21.8	2.7	1.07E-09	<1e-03

'Genes': the number of candidate that is taken for enrichment analysis; 'PID': the pathway ID used by *snpGeneSets*. 'size': the number of member genes of a pathway; 'setGenes': the number of candidate genes contained by the pathway.

7.2 *Example: Enrichment analysis I of T2D-GWES*

For T2D-GWES, the top *5%* genes with the smallest p-values of differential expression were selected as candidate genes and the pathway enrichment test were performed for KEGG gene sets by *enrichTest1()* function.

```
> library(snpGeneSets)

> data(T2DGWES)

> topExpGenes<-
T2DExpression[order(T2DExpression$p),][1:trunc(nrow(T2DExpression)*0.05),"gene_id"]

> length(topExpGenes)
[1] 1009
```

There are *1,009* candidate genes selected for the enrichment test of KEGG gene sets. However, only *262* genes belongs to the KEGG gene sets and are effectively used for pathway analysis (see codes below). The *10* most enriched gene sets were saved to *exp_rst* variable.

```
> expGeneSets_KEGG<-enrichTest1(topExpGenes,setType=6)

> expGeneSets_KEGG$nTopGenes
[1] 262

> exp_rst<-expGeneSets_KEGG$enrich_test[order(expGeneSets_KEGG$enrich_test$pval),][1:10,]

> exp_rst
```

```
     pid size genesSize     effect         sd       pval
86  1349  267        22 0.03264387 0.01330677 0.01277931
155 1418   53         7 0.08232234 0.02986692 0.01509198
152 1415   23         4 0.12415991 0.04533823 0.02510450
108 1371   47         6 0.07790644 0.03171608 0.02770251
149 1412   25         4 0.11024687 0.04348690 0.03319480
34  1297   44         5 0.06388323 0.03277948 0.06554085
133 1396  118        10 0.03499263 0.02001647 0.06795302
2   1265   32         4 0.07524687 0.03843735 0.07216414
99  1362  121        10 0.03289149 0.01976677 0.07764877
147 1410   47         5 0.05662985 0.03171608 0.08225960
```

The permutation test was applied to obtain permutation adjusted p-value (*p_perm*) by *enrichTest1_Perm()* function.

> exp_dist<-enrichTest1_Perm(genesSize =expGeneSets_KEGG$nTopGenes, setType=6,times=1000, seed=1)

> exp_min=apply(exp_dist,2,min)

> exp_rst$p_perm<-unlist(lapply(exp_rst$pval, function(x) sum(exp_min<=x)/1000))

> colnames(exp_rst)=c("pid","size","setTopGenes","effect","sd","p","p_perm")

> exp_rst

```
     pid size setTopGenes     effect         sd          p p_perm
86  1349  267          22 0.03264387 0.01330677 0.01277931  0.670
155 1418   53           7 0.08232234 0.02986692 0.01509198  0.734
152 1415   23           4 0.12415991 0.04533823 0.02510450  0.895
108 1371   47           6 0.07790644 0.03171608 0.02770251  0.909
149 1412   25           4 0.11024687 0.04348690 0.03319480  0.945
34  1297   44           5 0.06388323 0.03277948 0.06554085  1.000
133 1396  118          10 0.03499263 0.02001647 0.06795302  1.000
2   1265   32           4 0.07524687 0.03843735 0.07216414  1.000
99  1362  121          10 0.03289149 0.01976677 0.07764877  1.000
147 1410   47           5 0.05662985 0.03171608 0.08225960  1.000
```

> getGeneSetInfo(1418)

```
$setID
[1] 1418

$set_name
[1] "KEGG_AMYOTROPHIC_LATERAL_SCLEROSIS_ALS"

$set_link
[1] "http://www.broadinstitute.org/gsea/msigdb/cards/KEGG_AMYOTROPHIC_LATERAL_SCLEROSIS_ALS"

$set_type
        c1         c2     c2_cgp      c2_cp c2_biocarta     c2_kegg
     FALSE       TRUE      FALSE       TRUE       FALSE       TRUE
c2_reactome         c3     c3_mir     c3_tft         c4     c4_cgn
     FALSE      FALSE      FALSE      FALSE      FALSE      FALSE
     c4_cm         c5      c5_bp      c5_cc      c5_mf         c6
     FALSE      FALSE      FALSE      FALSE      FALSE      FALSE
        c7  hallmark
     FALSE      FALSE

$set_geneid
 [1]   842  7124   847  5535  5533  5534 57679   572  9973  4217 84134  2902
[13]  2903  5600  2904  6647  2905   637  2906  5606  5603  4842   581  4741
[25]  2891  6506  4744  2890  5608  7133   834   836  4747  7132  1616 63928
[37]  5879  1432  6300   317  2876  5868   596   598  5961  5532  7157  5530
[49]  5630 79139 54205 10452 11261
```

The gene set with *pid=1418* is the pathway of 'Amyotrophic lateral sclerosis' that contains *53* member genes. The pathway presented enrichment effect of *8.2%* with empirical p_e=*0.015*, but the test based on *1,000* permutations showed that the adjusted p-value was *0.734*.

8. Pathway Enrichment Analysis *II* of GWS genes

The type II analysis is a specialized pathway enrichment analysis that aims to identify enriched gene sets based on genome-wide association and expression study results. The analysis is performed by *enrichTest2()* function, which test for pathway enrichment by the *USGSA* method. The test depends the threshold of *U*-score that defines genome-wide significant genes. The default value of threshold is *0.05* for *enrichTest2()*, which assumes that *5%* of genome-wide genes are involved in pathway of studied phenotype.

8.1 *Example: Enrichment analysis II of T2D-GWAS*

Measures of *minP*, *2ndP, simP* and *fishP* or their *U*-scores can all be applied for pathway enrichment test. The required parameter of *geneDF* for *enrichTest2()* function is a data frame which contains at least a column of *'gene_id'* for Entrez gene IDs and a column of *'score'* for a gene measure or *U*-score. The argument of *'setType'* defines the pathway category for enrichment test. For the T2D-GWAS, the example below used *U*-score of *minp* measure for the analysis and *'setType=6'* limited enrichment analysis to pathways of the KEGG category.

```
> library(snpGeneSets)

> data(T2DGWAS)

> e2_minp<-enrichTest2(geneDF =
data.frame(gene_id=T2DGWASGene$gene_id,score=T2DGWASGene$minp), setType=6)

> names(e2_minp)
[1] "enrich_test" "useGenes"    "nGenes"     "nSigGenes"   "setTypeInfo"

> head(e2_minp$enrich_test)
   pid size genes sigGenes       effect         sd      pval
1 1264   62    50        1 -0.030047438 0.03083592 0.9244223
2 1265   32    24        0 -0.050047438 0.04450781 1.0000000
3 1266   27    18        0 -0.050047438 0.05139320 1.0000000
4 1267   28    22        0 -0.050047438 0.04648690 1.0000000
5 1268   34    24        1 -0.008380772 0.04450781 0.7093687
6 1269   26    25        1 -0.010047438 0.04360857 0.7239973

> length(e2_minp$useGenes)
[1] 4216

> e2_minp$nGenes
[1] 4216
```

```
> e2_minp$nSigGenes
[1] 211
```

```
> e2_minp$setTypeInfo
$id
[1] 6

$symbol
[1] "c2_kegg"

$name
[1] "C2_CP:KEGG: KEGG gene sets"

$description
[1] "Gene sets derived from the KEGG pathway database, http://www.genome.jp/kegg/pathway.html"
```

```
>e2_minp$enrich_test[order(e2_minp$enrich_test$pval),][1:10,]
```

	pid	size	genes	sigGenes	effect	sd	pval
184	1447	76	69	14	0.15285111	0.02624928	5.369381e-06
116	1379	75	70	11	0.10709542	0.02606111	6.049336e-04
183	1446	85	76	11	0.09468940	0.02501123	1.229043e-03
185	1448	92	81	11	0.08575503	0.02422699	2.087956e-03
117	1380	134	120	14	0.06661923	0.01990450	2.392497e-03
138	1401	70	65	9	0.08841410	0.02704489	4.600389e-03
105	1368	115	104	11	0.05572179	0.02138086	1.402733e-02
9	1272	17	12	3	0.19995256	0.06294355	1.943844e-02
147	1410	47	44	6	0.08631620	0.03287120	2.101725e-02
104	1367	80	57	7	0.07275958	0.02888048	2.236318e-02

For *U*-score of *minP*, the *enrichTest2()* function returned the results to the *e2_minp* variable, which consists of a data frame of *"enrich_test"*, an integer vector of *"useGenes"*, a number of *"nGenes"*, a number of *"nSigGenes"* and a list of *"setTypeInfo"*. The *"enrich_test"* shows the enrichment test results for every gene set in the specified category defined by *setType*. The *"useGenes"* lists GWS genes used for enrichment test. The *"nGenes"* is the total number of GWS genes in the specified category (i.e. the length of *"useGenes"*) and the *"nSigGenes"* is the number of GWS significant genes for enrichment test. The *"setTypeInfo"* presents description of the specified category.

The examples below similarly used *U*-scores of *2ndP, simP* and *fishP* for pathway tests.

(Notes: Either a gene measure or its U-score can be used for type II pathway test. Since a gene measure will automatically be converted to its U-score by enrichTest2 function, they will present the same results.)

```
> e2_sndp<-enrichTest2(geneDF =
data.frame(gene_id=T2DGWASGene$gene_id,score=T2DGWASGene$sndp), setType=6)
```

```
> e2_sndp$enrich_test[order(e2_sndp$enrich_test$pval),][1:10,]
```

```
     pid size genes sigGenes     effect         sd         pval
184 1447   76    69       17 0.19632937 0.02624928 2.410774e-08
185 1448   92    81       15 0.13513775 0.02422699 8.096803e-06
183 1446   85    76       14 0.13416309 0.02501123 1.745725e-05
86  1349  267   237       25 0.05543779 0.01416341 2.532973e-04
105 1368  115   104       13 0.07495256 0.02138086 1.824845e-03
104 1367   80    57        9 0.10784730 0.02888048 1.831769e-03
117 1380  134   120       14 0.06661923 0.01990450 2.392497e-03
6   1269   26    25        5 0.14995256 0.04360857 6.995776e-03
113 1376  201   180       17 0.04439701 0.01625196 7.927101e-03
34  1297   44    36        6 0.11661923 0.03634048 8.076124e-03
```

> e2_simp<-enrichTest2(geneDF = data.frame(gene_id=T2DGWASGene$gene_id,score=T2DGWASGene$simp), setType=6)

> e2_simp$enrich_test[order(e2_simp$enrich_test$pval),][1:10,]

```
     pid size genes sigGenes     effect         sd         pval
82  1345   44    35        6 0.12138113 0.03685597 0.007015925
124 1387   71    50        7 0.08995256 0.03083592 0.011337674
180 1443   38    28        5 0.12852399 0.04120623 0.011456474
16  1279   31    20        4 0.14995256 0.04875587 0.015692518
41  1304   25    21        4 0.14042875 0.04758085 0.018642681
148 1411   44    35        5 0.09280970 0.03685597 0.028533841
134 1397   48    37        5 0.08508770 0.03584603 0.035362619
22  1285   29    26        4 0.10379872 0.04276172 0.038352462
77  1340  128    92        9 0.04777865 0.02273254 0.039193127
76  1339   36    27        4 0.09810071 0.04196237 0.043318076
```

> e2_fishp<-enrichTest2(geneDF = data.frame(gene_id=T2DGWASGene$gene_id,score=T2DGWASGene$fishp), setType=6)

> e2_fishp$enrich_test[order(e2_fishp$enrich_test$pval),][1:10,]

```
     pid size genes sigGenes     effect         sd         pval
184 1447   76    69       16 0.18183662 0.02624928 1.589111e-07
117 1380  134   120       18 0.09995256 0.01990450 2.169739e-05
86  1349  267   237       27 0.06387661 0.01416341 3.656730e-05
132 1395   97    81       14 0.12279207 0.02422699 3.704396e-05
113 1376  201   180       22 0.07217478 0.01625196 7.065220e-05
183 1446   85    76       13 0.12100519 0.02501123 7.907991e-05
185 1448   92    81       13 0.11044639 0.02422699 1.555760e-04
114 1377   84    75       12 0.10995256 0.02517742 2.878815e-04
131 1394   79    71       11 0.10488214 0.02587693 6.849502e-04
176 1439   54    50        9 0.12995256 0.03083592 6.918384e-04
```

The top 10 gene sets for each gene measure were shown above. Consistent with the type *I* analysis, the most enriched gene set is the pathway of *PID=1447* for *minP*, *2ndP* and *fishP* measures. However, the most enriched pathway for *simP* is the 'KEGG_NUCLEOTIDE_EXCISION_REPAIR' (*PID=1345*) in contrast to the pathway of *PID=1387* by enrichment analysis *I*.

```
> KEGG_rst<-rbind(
    e2_minp$enrich_test[e2_minp$enrich_test$pval==min(e2_minp$enrich_test$pval),],
    e2_sndp$enrich_test[e2_sndp$enrich_test$pval==min(e2_sndp$enrich_test$pval),],
    e2_simp$enrich_test[e2_simp$enrich_test$pval==min(e2_simp$enrich_test$pval),],
    e2_fishp$enrich_test[e2_fishp$enrich_test$pval==min(e2_fishp$enrich_test$pval),])

> KEGG_rst<-cbind(measure=c("minp","2ndp","simp","fishp"),
                  topGenes=c(e2_minp$nSigGenes,e2_sndp$nSigGenes,
                  e2_simp$nSigGenes,e2_fishp$nSigGenes), KEGG_rst)
```

```
> colnames(KEGG_rst)<-c("measure",
                        "sigGenes","pid","size","effectGenes","setSigGenes","effect","sd","p")

> KEGG_rst

     measure sigGenes  pid size effectGenes setSigGenes    effect         sd
184     minp      211 1447   76          69          14 0.1528511 0.02624928
1841    2ndp      211 1447   76          69          17 0.1963294 0.02624928
82      simp      211 1345   44          35           6 0.1213811 0.03685597
1842   fishp      211 1447   76          69          16 0.1818366 0.02624928
                 p
184  5.369381e-06
1841 2.410774e-08
82   7.015925e-03
1842 1.589111e-07
```

For enrichment analysis *II*, the pathway of *'1447'*, containing *69* GWAS genes, involves *14* significant genes from *minP* (*effect=15.3%, p_e=5.37E-06*), *17* significant genes from *2ndP* (*effect=19.6%, p_e =2.41E-08*) and *16* significant genes from *fishP* (*effect=18.2%, p_e =1.59E-07*); and the pathway of *'1345'*, containing *35* GWAS genes, involves *6* significant genes from *simP* (*effect=12.1%, p_e =7.02E-03*).

To adjust for pathway dependence and multiple testing, the *enrichTest2_Perm()* function calculates the adjusted p-value (*p_perm*) by *1,000* permutations. The argument of *geneDF* is the data frame for enrichment test II by *enrichTest2()* function. The argument of *setType* defines the category of gene sets for permutation adjusting. The argument of *times* specifies the number of permutations for generating distribution table and the argument of *seed* assigns a random seed for permutation. The permutation adjusted p-value is saved at the variable of *p_perm*.

```
> minp_dist =
enrichTest2_Perm(data.frame(gene_id=T2DGWASGene$gene_id,score=T2DGWASGene$minp),
setType=6,times=1000, seed=1)

> sndp_dist =
enrichTest2_Perm(data.frame(gene_id=T2DGWASGene$gene_id,score=T2DGWASGene$sndp),
setType=6,times=1000, seed=1)

> simp_dist =
enrichTest2_Perm(data.frame(gene_id=T2DGWASGene$gene_id,score=T2DGWASGene$simp),
setType=6,times=1000, seed=1)

> fishp_dist =
enrichTest2_Perm(data.frame(gene_id=T2DGWASGene$gene_id,score=T2DGWASGene$fishp),
setType=6,times=1000, seed=1)

> minp_min=apply(minp_dist,2,min)

> sndp_min=apply(sndp_dist,2,min)

> simp_min=apply(simp_dist,2,min)
```

```
> fishp_min=apply(fishp_dist,2,min)

> KEGG_rst$p_perm<-c(sum(minp_min<=KEGG_rst[1,"p"]),sum(sndp_min<=KEGG_rst[2,"p"]),
                     sum(simp_min<=KEGG_rst[3,"p"]),sum(fishp_min<=KEGG_rst[4,"p"]))
```

	measure	sigGenes	pid	size	effectGenes	setSigGenes	effect	sd
184	minp	211	1447	76	69	14	0.1528511	0.02624928
1841	2ndp	211	1447	76	69	17	0.1963294	0.02624928
82	simp	211	1345	44	35	6	0.1213811	0.03685597
1842	fishp	211	1447	76	69	16	0.1818366	0.02624928

	p	p_perm
184	5.369381e-06	0
1841	2.410774e-08	0
82	7.015925e-03	480
1842	1.589111e-07	0

The *p_perm* is <1E-3, <1E-3, 0.480 and <1E-3 for the most enriched pathways based on gene measures of *minP*, *2ndP, simP* and *fishP* respectively.

To enable direct calculation of permutation p-value, a pre-generated distribution table based on *10,000* permutations is made [8] and *getEnrich2P()* function is provided to obtain the permutation p-value (*p_table*) directly. The *p_table* is *4.00E-04*, *<1E-4*, *0.4927* and *<1E-4* for the most enriched pathways based on gene measures of *minP*, *2ndP, simP* and *fishP* respectively. The codes are shown below:

```
> KEGG_rst$p_table<-getEnrich2P(setP=KEGG_rst$p, setType=6)$perm$p

> KEGG_rst
```

	measure	sigGenes	pid	size	effectGenes	setSigGenes	effect	sd
184	minp	211	1447	76	69	14	0.1528511	0.02624928
1841	2ndp	211	1447	76	69	17	0.1963294	0.02624928
82	simp	211	1345	44	35	6	0.1213811	0.03685597
1842	fishp	211	1447	76	69	16	0.1818366	0.02624928

	p	p_perm	p_table
184	5.369381e-06	0	0.0004
1841	2.410774e-08	0	0.0000
82	7.015925e-03	480	0.4927
1842	1.589111e-07	0	0.0000

8.2 ***Example: Enrichment analysis II of T2D-GWES***

For type II enrichment analysis of GWES, differential expression p-value is typically used as measure of gene effect. As pathway analysis of GWAS, both gene measure and its calculated *U*-score can be applied to test pathway enrichment by *enrichTest2* function. For the example of T2D-GWES data, the default value of *U*-score threshold=*0.05* were used for enrichment test of KEGG pathways (i.e. *setType=6*).

```
> library(snpGeneSets)

> data(T2DGWES)

> expGeneSets_KEGG<-
enrichTest2(data.frame(gene_id=T2DExpression$gene_id,score=uscore(T2DExpression$p)),
```

setType=6)

The *10* most enriched pathways for significant GWES genes were identified as below and results were saved to *exp_rst* variable.

> exp_rst<-expGeneSets_KEGG$enrich_test[order(expGeneSets_KEGG$enrich_test$pval),][1:10,]

The permutation test was applied to obtain permutation p-value (*p_perm*) by *enrichTest2_Perm()* function.

> exp_dist =
enrichTest2_Perm(data.frame(gene_id=T2DExpression$gene_id,score=uscore(T2DExpression$p)),

setType=6,times=1000, seed=1)

> exp_min=apply(exp_dist,2,min)

> exp_rst$p_perm<-unlist(lapply(exp_rst$pval, function(x) sum(exp_min<=x)/1000))

To enable direct calculation of permutation p-value, a pre-generated distribution table based on *10,000* permutations [8] is made and *getEnrich2P()* function is provided to obtain the permutation p-value (*p_table*) directly.

> exp_rst$p_table<-getEnrich2P(setP=exp_rst$pval, setType=6)$perm$p

> exp_rst

	pid	size	genes	sigGenes	effect	sd	pval	p_perm	p_table
86	1349	267	259	22	0.03487844	0.01355060	0.009573327	0.583	0.5852
155	1418	53	52	7	0.08455174	0.03024174	0.014044987	0.717	0.7214
152	1415	23	23	4	0.12384940	0.04547205	0.025583260	0.906	0.8974
133	1396	118	112	10	0.03922207	0.02060627	0.052547979	0.994	0.9909
2	1265	32	29	4	0.08786739	0.04049575	0.054401762	0.996	0.9923
99	1362	121	119	10	0.03396997	0.01999102	0.073193725	1.000	0.9993
70	1333	22	20	3	0.09993636	0.04876334	0.075311731	1.000	0.9995
108	1371	47	46	5	0.05863201	0.03215360	0.077967940	1.000	0.9995
157	1420	35	34	4	0.06758342	0.03739978	0.087729163	1.000	0.9998
143	1406	101	95	8	0.03414689	0.02237416	0.101550921	1.000	1.0000

9. Pathway Enrichment Analysis of GWAS by ALIGATOR

The ALIGATOR (Association LIst Go AnnoTatOR)[9] method is also implemented in the *snpGeneSets* package by the function *alligator().* The method tests pathway enrichment for GWAS significant gene that is defined through p-value threshold *pcut* of SNP association. The default value of *pcut* is 0.05 for *alligator()*, and any gene with a SNP p-value < *pcut* is defined as significant. The method applies permutation to obtain empirical unadjusted p-value and the number of permutation is defined through parameter *Nsample* that takes default value of *5000*. The adjusted p-value is obtained through bootstrap sampling and the number of bootstrapping is set through parameter Btimes that takes default value of *1000*.

The example below shows the analysis of pathway enrichment for T2DGWAS by ALIGATOR method. The first parameter *snpGeneP* is a data frame containing at least columns of '*snp*' (SNP rsid) , *'gene_id'* (Entrez gene ID) and '*p'* (SNP association p-value) . The data of *T2DGWAS* comes with the *snpGeneP* data frame and *pcut* of *0.001* is applied to test pathway enrichment.

```
> library(snpGeneSets)

> data(T2DGWAS)

>head(snpGeneP)
```

```
             snp gene_id         p
99333     rs3830       1 0.1499238
160287  rs893184       1 0.6181964
68197   rs226380       2 0.6519623
68198   rs226381       2 0.6678955
68199   rs226389       2 0.7031433
114987 rs4882978       2 0.6321011
```

```
> path0=aligator(snpGeneP, pcut=0.001)

>  path0[order(path0$p),][1:10,]
```

```
      pid      p adj_p
4232 6252 0.0002 0.459
3354 5374 0.0008 0.743
2129 4149 0.0012 0.839
4259 6279 0.0016 0.896
28   1291 0.0030 0.978
4619 6639 0.0034 0.983
4515 6535 0.0036 0.986
4153 6173 0.0038 0.990
16   1279 0.0044 0.994
4095 6115 0.0046 0.994
```

It was shown that the first pathway with *pid=6252* has empirical unadjusted p-value of 2E-04, but the permutation adjusted p-value is 0.459.

References:

1. Mei H, Li L, Jiang F, Simino J, Griswold M, Mosley T, Liu S: **snpGeneSets: An R Package for Genome-Wide Study Annotation**. *G3* 2016, **6**(12):4087-4095.
2. Sherry ST, Ward MH, Kholodov M, Baker J, Phan L, Smigielski EM, Sirotkin K: **dbSNP: the NCBI database of genetic variation**. *Nucleic Acids Res* 2001, **29**(1):308-311.
3. Maglott D, Ostell J, Pruitt KD, Tatusova T: **Entrez Gene: gene-centered information at NCBI**. *Nucleic Acids Res* 2011, **39**(Database issue):D52-57.
4. Liberzon A: **A description of the Molecular Signatures Database (MSigDB) Web site**. *Methods in molecular biology* 2014, **1150**:153-160.
5. Scott LJ, Mohlke KL, Bonnycastle LL, Willer CJ, Li Y, Duren WL, Erdos MR, Stringham HM, Chines PS, Jackson AU *et al*: **A genome-wide association study of type 2 diabetes in Finns detects multiple susceptibility variants**. *Science* 2007, **316**(5829):1341-1345.
6. Marselli L, Thorne J, Dahiya S, Sgroi DC, Sharma A, Bonner-Weir S, Marchetti P, Weir GC: **Gene expression profiles of Beta-cell enriched tissue obtained by laser capture microdissection from subjects with type 2 diabetes**. *PLoS One* 2010, **5**(7):e11499.
7. Smyth GK: **Linear models and empirical Bayes methods for assessing differential expression in microarray experiments.** . *Statistical Applications in Genetics and Molecular Biology 3, No 1, Article 3* 2004.
8. Mei H, Li L, Liu S, Jiang F, Griswold M, Mosley T: **The uniform-score gene set analysis for identifying common pathways associated with different diabetes traits**. *BMC genomics* 2015, **16**(1):336.
9. Holmans P, Green EK, Pahwa JS, Ferreira MA, Purcell SM, Sklar P, Owen MJ, O'Donovan MC, Craddock N: **Gene ontology analysis of GWA study data sets provides insights into the biology of bipolar disorder**. *American journal of human genetics* 2009, **85**(1):13-24.