Package 'GIGSEA'

October 17, 2024

Type Package

Title Genotype Imputed Gene Set Enrichment Analysis

Version 1.22.0

Author Shijia Zhu

Maintainer Shijia Zhu <shijia.zhu@mssm.edu>

Description 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.

License LGPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.5), Matrix, MASS, locfdr, stats, utils

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

biocViews GeneSetEnrichment,SNP,VariantAnnotation,GeneExpression,GeneRegulation,Regression,DifferentialExpression

git_url https://git.bioconductor.org/packages/GIGSEA

git_branch RELEASE_3_19

git_last_commit 56024fd

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-10-16

Contents

`dataframe2geneSet` *dataframe2geneSet*

Description

dataframe2geneSet transforms a data frame (1term-1gene) into geneSets (1term-Ngenes).

Usage

```
dataframe2geneSet(term, gene, value = NULL)
```

Arguments

`term`	a character value incidating the name of the column for the gene set (terms).
`gene`	a character value incidating the name of the column for the genes.
`value`	a vector of numeric values indicating the connectivity of between terms and genes. It could take either discrete values (0 and 1) or continuous values.

Value

a data frame, comprising three vectors: term (like pathway names), geneset (a gene symbol list separate by comma), and value (either discrete or continuous separated by comma)

Author(s)

Shijia Zhu, <shijia.zhu@mssm.edu>

See Also

geneSet2Net; geneSet2sparseMatrix;

---

geneSet2Net *geneSet2Net*

Description

geneSet2Net transforms gene sets to a matrix, which represents the connectivity between terms and genes.

Usage

```
geneSet2Net(term, geneset, value = NULL, sep = ",")
```

Arguments

term	a vector of character values incidating the names of gene sets, like pathway names and miRNA names.
geneset	a vector of character values, where each value is a gene list separated by 'sep'.
value	a vector of numeric values indicating the connectivity of between terms and genes. It could take either discrete values (0 and 1) or continuous vlaues.
sep	a character which separates the genes in the geneset.

Value

a matrix of numeric values where the column corresponds to the term and the row corresponds to the geneset.

Author(s)

Shijia Zhu, <shijia.zhu@mssm.edu>

See Also

geneSet2sparseMatrix; gmt2geneSet;

Examples

```
# download the gmt file
gmt <- readLines( paste0('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 (here is drug), geneset (a gene symbol list separate by comma),
# and value (1 and -1 separate by comma)
gff_up <- gmt2geneSet( gmt[index_up], termCol=c(1,2), singleValue = 1 )
gff_down <- gmt2geneSet( gmt[index_down], termCol=c(1,2), singleValue = -1 )

# combine up and down-regulated gene sets, and use 1 and -1 to indicate
# their direction
# extract the drug names
term_up<-vapply( gff_up$term, function(x) gsub('-up','',x), character(1) )
term_down<-vapply( gff_down$term, function(x) gsub('-dn','',x), character(1))
all(term_up==term_down)

# combine the up-regulated and down-regulated gene names for each
# drug perturbation
geneset <- vapply(1:nrow(gff_up),function(i) paste(gff_up$geneset[i],
gff_down$geneset[i],sep=','), character(1) )

# use 1 and -1 to indicate the direction of up and down-regulated genes
value <- vapply( 1:nrow(gff_up) , function(i) paste(gff_up$value[i],
gff_down$value[i],sep=',') , character(1) )

# transform the gene set into matrix, where the row represents the gene,
# the column represents the drug perturbation, and each entry takes values
# of 1 and -1
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 the drug perturbation, and each entry takes
# values of 1 and -1
net2 <- geneSet2sparseMatrix( term=term_up , geneset=geneset , value=value )
tail(net1[,1:30])
tail(net2[,1:30])
# the size of sparse matrix is much smaller than the matrix
format( object.size(net1), units = "auto")
format( object.size(net2), units = "auto")
```

geneSet2sparseMatrix *geneSet2sparseMatrix*

Description

geneSet2sparseMatrix transforms gene sets to a sparse matrix, which represents the connectivity between terms and genes.

Usage

```
geneSet2sparseMatrix(term, geneset, value = NULL, sep = ",")
```

Arguments

`term` a vector of character values incidating the names of gene sets, e.g., pathway names and miRNA names.

`geneset` a vector of character values, where each value is a gene list separated by 'sep'.

`value` a vector of numeric values indicating the connectivity of between terms and genes. It could take either discrete values (0 and 1) or continuous values.

`sep` a character which separates the genes in the geneset.

Value

a sparse matrix where the column corresponds to the term and the row corresponds to the geneset.

Author(s)

Shijia Zhu, <`shijia.zhu@mssm.edu`>

See Also

`gmt2geneSet`; `geneSet2Net`;

Examples

```
# download the gmt file
gmt <- readLines( paste0('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 (here is drug), geneset (a gene symbol list separate by comma),
# and value (1 and -1 separate by comma)
gff_up <- gmt2geneSet( gmt[index_up], termCol=c(1,2), singleValue = 1 )
gff_down <- gmt2geneSet( gmt[index_down], termCol=c(1,2), singleValue = -1 )

# combine up and down-regulated gene sets, and use 1 and -1 to indicate
# their direction
# extract the drug names
term_up<-vapply(gff_up$term, function(x) gsub('-up','',x), character(1))
```

```
term_down<-vapply(gff_down$term, function(x) gsub('-dn','',x), character(1))
all(term_up==term_down)

# combine the up-regulated and down-regulated gene names for each
# drug perturbation
geneset <- vapply(1:nrow(gff_up),function(i) paste(gff_up$geneset[i],
gff_down$geneset[i],sep=','), character(1) )

# use 1 and -1 to indicate the direction of up and down-regulated genes
value <- vapply( 1:nrow(gff_up) , function(i) paste(gff_up$value[i],
gff_down$value[i],sep=',') , character(1) )


# transform the gene set into matrix, where the row represents the gene,
# the column represents the drug perturbation, and each entry takes values
# of 1 and -1
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 the drug perturbation, and each entry takes
# values of 1 and -1
net2 <- geneSet2sparseMatrix( term=term_up , geneset=geneset , value=value )
tail(net1[,1:30])
tail(net2[,1:30])
# the size of sparse matrix is much smaller than the matrix
format( object.size(net1), units = "auto")
format( object.size(net2), units = "auto")
```

gmt2geneSet *gmt2geneSet*

Description

gmt2geneSet transforms a gmt format file into geneSets.

Usage

```
gmt2geneSet(gmt, termCol = 1, nonGeneCol = termCol, singleValue = NULL)
```

Arguments

`gmt`	a vector of character values. Each item is a list of words comprising a term and its corresponding gene set, which are separated by tab.
`termCol`	an integer value indicating in each item of gmt, which word is the term , by default, 1.
`nonGeneCol`	an integer value indicating in each item of gmt, which words are not the gene set, by default, termCol.
`singleValue`	a numeric value, which assigns the same value to all genes in a given gene set. This is useful when combining together the up-regulated gene sets (regularly, singleValue=1) and the down-regulated gene sets (regularly, singleValue=-1)

Value

a data frame, comprising three vectors: term (like pathway names), geneset (a gene symbol list separate by comma), and value (either discrete or continuous separated by comma)

Author(s)

Shijia Zhu, <shijia.zhu@mssm.edu>

See Also

geneSet2Net; geneSet2sparseMatrix;

Examples

```
# download the gmt file
gmt <- readLines( paste0('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 (here is drug), geneset (a gene symbol list separate by comma),
# and value (1 and -1 separate by comma)
gff_up <- gmt2geneSet( gmt[index_up], termCol=c(1,2), singleValue = 1 )
gff_down <- gmt2geneSet( gmt[index_down], termCol=c(1,2), singleValue = -1 )
```

---

heart.metaXcan *heart.metaXcan*

Description

The MetaXcan-predicted differential gene expression from the cardiovascular disease (CVD) GWAS, CARDIoGRAMplusC4D (60,801 cases, 123,504 controls and 9.4M SNPs).

Usage

```
heart.metaXcan
```

Format

A data frame with the following items:

gene a gene's id

gene_name a gene's name

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 transcriptome prediction model's correlation to gene's measured transcriptome

pred_perf_pval pval of transcriptome prediction model's correlation to gene's measured transcriptome

pred_perf_qval qval of transcriptome prediction 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 prediction model

var_g variance of the gene expression ...

Source

`http://www.cardiogramplusc4d.org/data-downloads/;https://cloud.hakyimlab.org/s-predixcan`

`matrixPval` *matrixPval*

Description

matrixPval calculates the p values for the correlation coefficients based on t-statistics

Usage

```
matrixPval(r, df)
```

Arguments

`r`	a vector or a matrix of Pearson correlation coefficients taking values in [-1,+1]
`df`	the degree of freedom

Value

a vector or matrix of p values taking values in [0,1]

Examples

```
r <- cor(USArrests)
df <- nrow(USArrests) - 2
pval1 <- matrixPval(r,df)

pval2 <- matrix(ncol=ncol(USArrests),nrow=ncol(USArrests),data=0)
for(i in 1:ncol(USArrests))
{
   for(j in 1:ncol(USArrests))
   {
     pval2[i,j] <- cor.test(USArrests[,i],USArrests[,j])$p.val
   }
}

head(pval1)
head(pval2)
```

MSigDB.KEGG.Pathway *MSigDB.KEGG.Pathway*

Description

Gene sets derived from the KEGG pathway database.

Usage

```
MSigDB.KEGG.Pathway
```

Format

A list with two items:

net a sparse matrix, the connectivity between terms and genes, comprising 186 pathways (column) and 5267 genes (row)

annot a data frame, description of terms ...

Source

software.broadinstitute.org/gsea/msigdb/collections.jsp#C2

`MSigDB.miRNA` *MSigDB.miRNA*

Description

Gene sets that contain genes sharing putative target sites (seed matches) of human mature miRNA in their 3'-UTRs.

Usage

```
MSigDB.miRNA
```

Format

A list with two items:

net a sparse matrix, the connectivity between terms and genes, comprising 221 miRNAs (column) and 7444 genes (row)

annot a data frame, description of terms ...

Source

software.broadinstitute.org/gsea/msigdb/collections.jsp#C3

`MSigDB.TF` *MSigDB.TF*

Description

Gene sets that share upstream cis-regulatory motifs which can function as potential transcription factor binding sites.

Usage

```
MSigDB.TF
```

Format

A list with two items:

net a sparse matrix, the connectivity between terms and genes, comprising 615 TFs (column) and 12774 genes (row)

annot a data frame, description of terms ...

Source

software.broadinstitute.org/gsea/msigdb/collections.jsp#C3

orderedIntersect *orderedIntersect*

Description

orderedIntersect sorts a data frame based on a given collumn and intersects with another vector.

Usage

```
orderedIntersect(x, by.x, by.y)
```

Arguments

x	a data frame
by.x	a vector of character values. The data frame is sorted based on by.x
by.y	a vector of character values. After being sorted, the rows of x are further filtered by intersecting by.x with by.y

Value

a data frame sorted by "by.x" and intersected with "by.y"

Author(s)

Shijia Zhu, <shijia.zhu@mssm.edu>

Examples

```
# load data
data(heart.metaXcan)
gene <- heart.metaXcan$gene_name

# extract the imputed Z-score of gene differential expression, which follows
# 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)

net <- MSigDB.KEGG.Pathway$net
```

```
# intersect the user-provided imputed genes with the gene set of interest
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) )
```

permutationMultipleLm *permutationMultipleLm*

Description

permutationMultipleLm is a permutation test to calculate the empirical p values for a weighted multiple linear regression.

Usage

```
permutationMultipleLm(fc, net, weights = rep(1, nrow(net)), num = 100,
  verbose = TRUE)
```

Arguments

`fc`	a vector of numeric values representing gene expression fold change
`net`	a matrix of numeric values in the size of gene number x gene set number, representing the connectivity between genes and gene sets
`weights`	a vector of numeric values representing the weights of permuated genes
`num`	an integer value representing the number of permutations
`verbose`	an boolean value indicating whether or not to print output to the screen

Value

a data frame comprising the following columns:

- term a vector of character incidating the names of gene sets.
- usedGenes a vector of numeric values indicating the number of genes used in the model.
- Estimate a vector of numeric values indicating the regression coefficients.
- Std..Error a vector of numeric values indicating the standard errors of regression coefficients.
- t.value a vector of numeric values indicating the t-statistics of regression coefficients.
- observedPval a vector of numeric values [0,1] indicating the p values from the multiple weighted regression model.
- empiricalPval a vector of numeric values [0,1] indicating the empirical p values from the permutation test.

Author(s)

Shijia Zhu, `<shijia.zhu@mssm.edu>`

See Also

orderedIntersect; permutationMultipleLmMatrix;

Examples

```
# load data
data(heart.metaXcan)
gene <- heart.metaXcan$gene_name

# extract the imputed Z-score of differential gene expression, which follows
# the normal distribution
fc <- heart.metaXcan$zscore

# use as weights the prediction R^2 and the fraction of imputation-used SNPs
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)

net <- MSigDB.KEGG.Pathway$net

# intersect the imputed genes with the gene sets of interest
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) )

# the MGSEA.res1 uses the weighted multiple linear regression to do
# permutation test,
# while MGSEA.res2 used the solution of weighted matrix operation. The
# latter one takes substantially less time.
# system.time( MGSEA.res1<-permutationMultipleLm(fc=data2$fc, net=net2,
# weights=data2$weights, num=1000))
# system.time( MGSEA.res2<-permutationMultipleLmMatrix(fc=data2$fc,
# net=net2, weights=data2$weights, num=1000))
# head(MGSEA.res2)
```

---

permutationMultipleLmMatrix *permutationMultipleLmMatrix*

Description

permutationMultipleLmMatrix is a permutation test to calculate the empirical p values for weighted multiple linear regression

Usage

```
permutationMultipleLmMatrix(fc, net, weights = rep(1, nrow(net)), num = 100,
  step = 1000, verbose = TRUE)
```

Arguments

fc	a vector of numeric values representing gene expression fold change
net	a matrix of numeric values in the size of gene number x gene set number, representing the connectivity between genes and gene sets
weights	a vector of numeric values representing the weights of permuated genes
num	an integer value representing the number of permutations
step	an integer value representing the number of permutations in each step
verbose	an boolean value indicating whether or not to print output to the screen

Value

a data frame comprising following columns:

- term a vector of character values incidating the names of gene sets.
- usedGenes a vector of numeric values indicating the number of genes used in the model.
- observedTstats a vector of numeric values indicating the observed t-statistics for the weighted multiple regression coefficients.
- empiricalPval a vector of numeric values [0,1] indicating the permutation-based empirical p values.
- BayesFactor a vector of numeric values indicating the Bayes Factor for the multiple test correction.

Author(s)

Shijia Zhu, <shijia.zhu@mssm.edu>

See Also

orderedIntersect; permutationMultipleLm;

Examples

```
# load data
data(heart.metaXcan)
gene <- heart.metaXcan$gene_name

# extract the imputed Z-score of differential gene expression, which
# follows the normal distribution
fc <- heart.metaXcan$zscore

# use as weights the prediction R^2 and the fraction of imputation-used SNPs
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)

net <- MSigDB.KEGG.Pathway$net

# intersect the imputed genes with the gene sets of interest
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) )

# the MGSEA.res1 uses the weighted multiple linear regression to do
# permutation test,
# while MGSEA.res2 used the solution of weighted matrix operation. The
# latter one takes substantially less time.
# system.time( MGSEA.res1<-permutationMultipleLm(fc=data2$fc, net=net2,
# weights=data2$weights, num=1000))
system.time( MGSEA.res2<-permutationMultipleLmMatrix(fc=data2$fc, net=net2,
weights=data2$weights, num=1000))
head(MGSEA.res2)
```

permutationSimpleLm *permutationSimpleLm*

Description

permutationSimpleLm is a permutation test to calculate the empirical p values for a weighted simple linear regression.

Usage

```
permutationSimpleLm(fc, net, weights = rep(1, nrow(net)), num = 100,
  verbose = TRUE)
```

Arguments

fc	a vector of numeric values representing the gene expression fold change
net	a matrix of numeric values in the size of gene number x gene set number, representing the connectivity betweeen genes and gene sets

`weights` a vector of numeric values representing the weights of permuted genes

`num` a vector of integer values representing the number of permutations

`verbose` an boolean value indicating whether or not to print output to the screen

Value

a data frame comprising the following columns:

- term a vector of character values incidating the name of gene set.
- usedGenes a vector of numeric values indicating the number of genes used in the model.
- Estimate a vector of numeric values indicating the regression coefficients.
- Std..Error a vector of numeric values indicating the standard errors of regression coefficients.
- t.value a vector of numeric values indicating the t-statistics of regression coefficients.
- observedPval a vector of numeric values [0,1] indicating the p values from weighted simple regression model.
- empiricalPval a vector of numeric values [0,1] indicating the empirical p values from the permutation test.

Author(s)

Shijia Zhu, <shijia.zhu@mssm.edu>

See Also

orderedIntersect; permutationSimpleLmMatrix;

Examples

```
# load data
data(heart.metaXcan)
gene <- heart.metaXcan$gene_name

# extract the imputed Z-score of gene differential expression, which
# follows the normal distribution
fc <- heart.metaXcan$zscore

# use as weights the prediction R^2 and the fraction of imputation-used SNPs
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)

net <- MSigDB.KEGG.Pathway$net

# intersect the permuted genes with the gene sets of interest
```

```
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) )

# the SGSEA.res1 uses the weighted simple linear regression model,
# while SGSEA.res2 used the weighted Pearson correlation. The latter one
# takes substantially less time.
# system.time(SGSEA.res1<-permutationSimpleLm(fc=data2$fc, net=net2,
# weights=data2$weights, num=1000))
# system.time(SGSEA.res2<-permutationSimpleLmMatrix(fc=data2$fc, net=net2,
# weights=data2$weights, num=1000))
# head(SGSEA.res2)
```

`permutationSimpleLmMatrix`

permutationSimpleLmMatrix

Description

permutationSimpleLmMatrix is a permutation test to calculate the empirical p values for the weighted simple linear regression model based on the weighted Pearson correlation.

Usage

```
permutationSimpleLmMatrix(fc, net, weights = rep(1, nrow(net)), num = 100,
  step = 1000, verbose = TRUE)
```

Arguments

`fc` a vector of numeric values representing the gene expression fold change

`net` a matrix of numeric values in the size of gene number x gene set number, representing the connectivity betwen genes and gene sets

`weights` a vector of numeric values representing the weights of permuted genes

`num` an integer value representing the number of permutations

`step` an integer value representing the number of permutations in each step

`verbose` an boolean value indicating whether or not to print output to the screen

Value

a data frame comprising following columns:

- term a vector of character values incidating the name of gene set.
- usedGenes a vector of numeric values indicating the number of gene used in the model.

- observedCorr a vector of numeric values indicating the observed weighted Pearson correlation coefficients.
- empiricalPval a vector of numeric values [0,1] indicating the permutation-based empirical p values.
- BayesFactor a vector of numeric values indicating the Bayes Factor for the multiple test correction.

Author(s)

Shijia Zhu, <shijia.zhu@mssm.edu>

See Also

orderedIntersect; permutationSimpleLm;

Examples

```
# load data
data(heart.metaXcan)
gene <- heart.metaXcan$gene_name

# extract the imputed Z-score of gene differential expression, which follows
# the normal distribution
fc <- heart.metaXcan$zscore

# use as weights the prediction R^2 and the fraction of imputation-used SNPs
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)

net <- MSigDB.KEGG.Pathway$net

# intersect the permuted genes with the gene sets of interest
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) )

# the SGSEA.res1 uses the weighted simple linear regression model,
# while SGSEA.res2 used the weighted Pearson correlation. The latter one
# takes substantially less time.
# system.time(SGSEA.res1<-permutationSimpleLm(fc=data2$fc, net=net2,
# weights=data2$weights, num=1000))
system.time(SGSEA.res2<-permutationSimpleLmMatrix(fc=data2$fc, net=net2,
weights=data2$weights, num=1000))
```

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

`runGIGSEA` *runGIGSEA*

Description

runGIGSEA use MetaXcan to impute the trait-associated differential gene expression from GWAS summary and eQTL database first, and next, performs gene set enrichment analysis for the trait-associated SNPs.

Usage

```
runGIGSEA(MetaXcan, model_db_path, covariance, gwas_folder, gwas_file_pattern,
  snp_column = "SNP", non_effect_allele_column = "A2",
  effect_allele_column = "A1", or_column = "OR", beta_column = "BETA",
  beta_sign_column = "direction", zscore_column = "Z",
  pvalue_column = "P", gene_set = c("MSigDB.KEGG.Pathway", "MSigDB.TF",
  "MSigDB.miRNA", "TargetScan.miRNA"), permutation_num = 1000,
  output_dir = "./GIGSEA", MGSEA_thres = NULL, verbose = TRUE)
```

Arguments

`MetaXcan` a character value indicating the path to the MetaXcan.py file.

`model_db_path` a character value indicating the path to tissue transriptome model.

`covariance` a character value indicating the path to file containing covariance information. This covariance should have information related to the tissue transcriptome model.

`gwas_folder` a character value indicating the folder containing GWAS summary statistics data.

`gwas_file_pattern`
a regular expression indicating the gwas summary files.

`snp_column` a character value indicating the name of column holding SNP data, by default, "SNP".

`non_effect_allele_column`
a character value indicating the name of column holding "other/non effect" allele data, by default, "A2".

`effect_allele_column`
a character value indicating the name of column holding effect allele data, by default, "A1".

`or_column` a character value indicating the name of column holding Odd Ratio data, by default, "OR".

`beta_column` a character value indicating the name of column holding beta data, by default, "BETA".

`beta_sign_column`
a character value indicating the name of column holding sign of beta, by default, "direction".

`zscore_column` a character value indicating the name of column holding zscore of beta, by default, "Z".

`pvalue_column` a character value indicating the name of column holding p-values data, by default, "P".

`gene_set` a vector of characters indicating the gene sets of interest for enrichment test, by default, c("MSigDB.KEGG.Pathway","MSigDB.TF", "MSigDB.miRNA","Fantom5.TF","TargetScan.mi "LINCS.CMap.drug")

`permutation_num`
an integer indicating the number of permutation.

`output_dir` a character value indicating the directory for saving the results.

`MGSEA_thres` an integer value indicating the thresfold for performing MGSEA. When the number of gene sets is smaller than MGSEAthres, we perform MGSEA.

`verbose` an boolean value indicating whether or not to print output to the screen

Value

TRUE

Author(s)

Shijia Zhu, <`shijia.zhu@mssm.edu`>

References

Barbeira, A., et al. Integrating tissue specific mechanisms into GWAS summary results. bioRxiv 2016:045260. https://github.com/hakyimlab/MetaXcan

See Also

`weightedGSEA`;

Examples

```
# runGIGSEA( MetaXcan="/MetaXcan/software/MetaXcan.py" ,
  # model_db_path="data/DGN-WB_0.5.db" ,
  # covariance="data/covariance.DGN-WB_0.5.txt.gz" ,
  # gwas_folder="data/GWAS" ,
  # gwas_file_pattern="heart.summary" ,
  # zscore_column="Z" ,
  # output_dir="./GIGSEA",
  # permutation_num=1000)
```

`TargetScan.miRNA` *TargetScan.miRNA*

Description

Gene sets of predicted human miRNA targets were obtained 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.

Usage

```
TargetScan.miRNA
```

Format

A list with two items:

net a sparse matrix, the connectivity between terms and genes, comprising 87 miRNA seed motifs and 9861 genes

annot a data frame, description of terms ...

Source

http://www.targetscan.org

`weightedGSEA` *weightedGSEA*

Description

weightedGSEA performs both SGSEA and MGSEA for a given list of gene sets, and writes out the results.

Usage

```
weightedGSEA(data, geneCol, fcCol, weightCol = NULL,
  geneSet = c("MSigDB.KEGG.Pathway", "MSigDB.TF", "MSigDB.miRNA",
  "TargetScan.miRNA"), permutationNum = 100, outputDir = getwd(),
  MGSEAthres = NULL, verbose = TRUE)
```

Arguments

`data`	a data frame comprising comlumns: gene names (characer), differential gene expression (numeric) and permuated gene weights (numeric and optional)
`geneCol`	an integer or a character value indicating the column of gene name
`fcCol`	an integer or a character value indicating the column of differential gene expression
`weightCol`	an integer or a character value indicating the column of gene weights
`geneSet`	a vector of character values indicating the gene sets of interest.
`permutationNum`	an integer value indicating the number of permutation
`outputDir`	a character value indicating the directory for saving the results
`MGSEAthres`	an integer value indicating the thresfold for MGSEA. MGSEA is performed with no more than "MGSEAthres" gene sets
`verbose`	an boolean value indicating whether or not to print output to the screen

Value

TRUE

Examples

```
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","MSigDB.TF","MSigDB.miRNA",
# "TargetScan.miRNA"), permutationNum=10000, outputDir="./GIGSEA" )
# dir("./GIGSEA")
```

`weightedMultipleLm` *weightedMultipleLm*

Description

weightedMultipleLm solves the weighted multiple linear regression model via matrix operation

Usage

```
weightedMultipleLm(x, y, w = rep(1, nrow(x))/nrow(x))
```

Arguments

x	a matrix of numeric values in the size of genes x featureA
y	a matrix of numeric values in the size of genes x featureB
w	a vector of numeric values indicating the weights of genes

Value

a matrix of numeric values in the size of featureA*featureB, indicating the weighted multiple regression coefficients

Author(s)

Shijia Zhu, <shijia.zhu@mssm.edu>

See Also

orderedIntersect; matrixPval;

Examples

```
# load data
data(heart.metaXcan)
gene <- heart.metaXcan$gene_name

# extract the imputed Z-score of gene differential expression, which follows
# the normal distribution
fc <- heart.metaXcan$zscore

# use as weights the prediction R^2 and the fraction of imputation-used SNPs
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)

net <- MSigDB.KEGG.Pathway$net

# intersect the permuated genes with the gene sets of interest
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) )

# perform the weighted multiple linear regression
observedTstats = weightedMultipleLm( x=net2 , y=data2$fc, w=data2$weights )
```

```
# calculate the p values of the weighted multiple regression coefficients
observedPval = 2 * pt(abs(observedTstats), df=sum(weights>0,na.rm=TRUE)-2,
lower.tail=FALSE)

res = data.frame( observedTstats , observedPval )
head(res)
```

weightedPearsonCorr *weightedPearsonCorr*

Description

weightedPearsonCorr caculates the weighted Pearson correlation

Usage

```
weightedPearsonCorr(x, y, w = rep(1, nrow(x))/nrow(x))
```

Arguments

x	a matrix of numeric values in the size of genes x featureA
y	a matrix of numeric values in the size of genes x featureB
w	a vector of numeric values indicating the weights of genes

Value

a matrix of numeric values in the size of featureA*featureB, indicating the weighted Pearson correlation coefficients

Author(s)

Shijia Zhu, <shijia.zhu@mssm.edu>

See Also

orderedIntersect; matrixPval;

Examples

```
# load data
data(heart.metaXcan)
gene <- heart.metaXcan$gene_name

# extract the imputed Z-score of gene differential expression, which follows
# the normal distribution
fc <- heart.metaXcan$zscore
```

```
# use as weights the prediction R^2 and fraction of imputation-used SNPs
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 simple linear
# regression-based enrichment analysis
data <- data.frame(gene,fc,weights)
head(data)

net <- MSigDB.KEGG.Pathway$net

# intersect the imputed genes with the gene sets of interest
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) )

# calculate the weighted Pearson correlation
observedCorr = weightedPearsonCorr( x=net2 , y=data2$fc, w=data2$weights )

# calculate the p values of the weighted Pearson correlation
observedPval = matrixPval( observedCorr, df=sum(weights>0,na.rm=TRUE)-2 )

res = data.frame( observedCorr , observedPval )
head(res)
```

Index