Package 'GeneScan3DKnock'

August 29, 2021

Type Package

Title Powerful gene-based testing by integrating long-range chromatin interactions and knockoff genotypes

Version 0.3

Author Shiyang Ma, James Dalgleish, Zihuai He, Iuliana Ionita-Laza

Maintainer Shiyang Ma <sm4857@cumc.columbia.edu>

Description Functions for the gene-based association tests that integrate both common and rare genetic variation from promoter and enhancers for each gene, along with the knockoff-enhanced tests. The GeneScan3DKnock has two steps: Step 1. Knockoff generation using function GeneScan3D.KnockoffGeneration() and Step 2. Knockoff filter using function GeneScan3DKnock(). To deal with case-control imbalance issue for binary traits, we apply the Saddle-point approximation (SPA) of the gene-based tests to avoid the inflation of Type I error rate. The knockoff generations are optimized using shrinkage leveraging algorithm.

License GPL-3

Depends R(>= 3.5.0)

Imports SKAT,
Matrix,
MASS,
WGScan,
SPAtest,
CompQuadForm,
abind,
irlba

NeedsCompilation no

Repository CRAN

Encoding UTF-8

RoxygenNote 7.1.1

R topics documented:

`Example.GeneScan3D` *Data example for GeneScan3D (gene-based testing by integrating long-range chromatin interactions).*

Description

This simulated example dataset contains outcome variable Y, covariate X, genotype and functional annotation matrices for gene and buffer region, promoter and two enhancers, positions of genetic variants in gene and buffer region.

Usage

```
data("GeneScan3D.example")
```

Format

An object of class `list` of length 12.

Details

We generated genotypes for 2,000 individuals in a 14.5 Kb gene region, a promoter as a 0.5 Kb segment upstream of the TSS (the start point of the gene) and R = 2 enhancers with length 2 KB, which are outside the 15 Kb gene plus promoter region.

Examples

```
data("GeneScan3D.example")

Y=GeneScan3D.example$Y; X=GeneScan3D.example$X; n=length(Y)

G_gene_buffer=GeneScan3D.example$G_gene_buffer
G_promoter=GeneScan3D.example$G_promoter
G_EnhancerAll=cbind(GeneScan3D.example$G_Enhancer1,GeneScan3D.example$G_Enhancer2)

Z_gene_buffer=GeneScan3D.example$Z_gene_buffer
Z_promoter=GeneScan3D.example$Z_promoter
Z_EnhancerAll=rbind(GeneScan3D.example$Z_Enhancer1,GeneScan3D.example$Z_Enhancer2)

pos_gene_buffer=GeneScan3D.example$pos_gene_buffer
```

`Example.GeneScan3DKnock`

Data example for GeneScan3DKnock.

Description

This example dataset contains the original and M=5 knockoff p-values for N=100 genes. Each row presents gene id, original GeneScan3D p-value and M knockoff GeneScan3D p-values. The original and knockoff GeneScan3D p-values are generated using GeneScan3D.KnockoffGeneration() function.

Usage

```
data("GeneScan3DKnock.example")
```

Format

An object of class `data.frame` with 100 rows and 7 columns.

Details

This example dataset can be used to calculate the knockoff statistics and q-values for GeneScan3DKnock() function.

`Example.KnockoffGeneration`

Data example for AR Knockoff Generation.

Description

This simulated example dataset contains outcome variable Y, covariate X, genotype matrices and genetic variants of surrounding regions for gene buffer and two enhancers separately, positions and functional annotations for gene buffer region, promoter and two enhancers.

Usage

```
data("KnockoffGeneration.example")
```

Format

An object of class `list` of length 17.

Details

We provide genotypes of +-10 Kb neighborhoods for gene buffer region and two enhancers separately. In real data analyses, the neighborhood for knockoff generation should be increase to +-100 Kb.

Examples

```
data("KnockoffGeneration.example")

Y=KnockoffGeneration.example$Y; X=KnockoffGeneration.example$X;

G_gene_buffer_surround=KnockoffGeneration.example$G_gene_buffer_surround
variants_gene_buffer_surround=KnockoffGeneration.example$variants_gene_buffer_surround
G_Enhancer1_surround=KnockoffGeneration.example$G_Enhancer1_surround
variants_Enhancer1_surround=KnockoffGeneration.example$variants_Enhancer1_surround
G_Enhancer2_surround=KnockoffGeneration.example$G_Enhancer2_surround
variants_Enhancer2_surround=KnockoffGeneration.example$variants_Enhancer2_surround

gene_buffer.pos=KnockoffGeneration.example$gene_buffer.pos
promoter.pos=KnockoffGeneration.example$promoter.pos
Enhancer1.pos=KnockoffGeneration.example$Enhancer1.pos
Enhancer2.pos=KnockoffGeneration.example$Enhancer2.pos

Z_gene_buffer=KnockoffGeneration.example$Z_gene_buffer
Z_promoter=KnockoffGeneration.example$Z_promoter
Z_Enhancer1=KnockoffGeneration.example$Z_Enhancer1
Z_Enhancer2=KnockoffGeneration.example$Z_Enhancer2
```

GeneScan.Null.Model

The preliminary data management for GeneScan3DKnock.

Description

This function does the preliminary data management and fit the generalized linear model under null hypothesis for unrelated samples. The output will be used in the other GeneScan functions.

Usage

```
GeneScan.Null.Model(
  Y,
  X = NULL,
  id = NULL,
  out_type = "C",
  resampling = FALSE,
  B = 1000
)
```

Arguments

Y	The outcome variable, an n*1 matrix where n is the number of individuals.
X	An n*d covariates matrix where d is the number of covariates.
id	The subject id. This is used to match phenotype with genotype. The default is NULL, where the matched phenotype and genotype matrices are assumed.
out_type	Type of outcome variable. Can be either "C" for continuous or "D" for dichotomous. The default is "C".

resampling	Resampling indicator. The default is FALSE, do not conduct resampling-based moment matching when the sample size is large, especially for UK biobank-scale data.
B	Number of resampling replicates. The default is 1000, only run resampling replicates when the resampling indicator is TRUE. A larger value leads to more accurate and stable p-value calculation, but requires more computing time.

Value

It returns a list used for function GeneScan1D(), GeneScan3D() and GeneScan3D.KnockoffGeneration().

Examples

```
library(GeneScan3DKnock)

# Load data example
data("GeneScan3D.example")
# Y: outcomes, n by 1 matrix for n=2000 individuals
# X: covariates, n by d matrix for d=1 covariate
Y=GeneScan3D.example$Y; X=GeneScan3D.example$X;

# Preliminary data management
result.null.model=GeneScan.Null.Model(Y, X, out_type="C", resampling=FALSE)
```

GeneScan1D *Conduct GeneScan1D analysis on the gene buffer region.*

Description

This function conducts gene-based scan test on the gene buffer region using 1D windows with sizes 1-5-10 Kb. For binary traits, we conduct SPA gene-based tests to deal with imbalance case-control issues.

Usage

```
GeneScan1D(
  G = G_gene_buffer,
  Z = NULL,
  window.size = c(1000, 5000, 10000),
  pos = pos_gene_buffer,
  MAC.threshold = 10,
  MAF.threshold = 0.01,
  Gsub.id = NULL,
  resampling = FALSE,
  result.null.model = result.null.model
)
```

Arguments

G	The genotype matrix in the gene buffer region, which is a n*p matrix where n is the number of individuals and p is the number of genetic variants in the gene buffer region.
Z	A p*q functional annotation matrix where p is the number of genetic variants in the gene buffer region and q is the number of functional annotations. If Z is NULL (do not incorporate any functional annotations), the minor allele frequency weighted dispersion and/or burden tests are applied. Specifically, Beta(MAF; 1; 25) weights are used for rare variants and weights one are used for common variants.
window.size	The 1-D window sizes in base pairs to scan the gene buffer region. The recommended window sizes are c(1000,5000,10000).
pos	The positions of genetic variants in the gene buffer region, an p dimensional vector. Each position corresponds to a column in the genotype matrix and a row in the functional annotation matrix.
MAC.threshold	Threshold for minor allele count. Variants below MAC.threshold are ultra-rare variants. The recommended level is 10.
MAF.threshold	Threshold for minor allele frequency. Variants below MAF.threshold are rare variants. The recommended level is 0.01.
Gsub.id	The subject id corresponding to the genotype matrix, an n dimensional vector. The default is NULL, where the matched phenotype and genotype matrices are assumed.
resampling	Resampling indicator. The default is FALSE, do not conduct resampling-based moment matching when the sample size is large, especially for UK biobank-scale data.
result.null.model	The output of function "GeneScan.Null.Model()".

Value

GeneScan1D.Cauchy.pvalue	Cauchy combination p-values of all, common and rare variants for GeneScan1D analysis.
M	Number of 1D scanning windows.

Examples

```
library(GeneScan3DKnock)

# Load data example
# Y: outcomes, n by 1 matrix for n=2000 individuals
# X: covariates, n by d matrix for d=1 covariate
# G_gene_buffer: genotype matrix of gene buffer region, n by p matrix, p=287 variants
# Z_gene_buffer: p by q functional annotation matrix, q=1 functional annotation
# pos_gene_buffer: positions of p=287 genetic variants

data("GeneScan3D.example")
Y=GeneScan3D.example$Y; X=GeneScan3D.example$X;
G_gene_buffer=GeneScan3D.example$G_gene_buffer;Z_gene_buffer=GeneScan3D.example$Z_gene_bu
pos_gene_buffer=GeneScan3D.example$pos_gene_buffer;
```

```
# Preliminary data management
set.seed(12345)
result.null.model=GeneScan.Null.Model(Y, X, out_type="C", resampling=FALSE)

#Conduct GeneScan1D analysis
result.GeneScan1D=GeneScan1D(G=G_gene_buffer,Z=Z_gene_buffer,pos=pos_gene_buffer,
                             window.size=c(1000,5000,10000),
                             MAC.threshold=10,MAF.threshold=0.01,Gsub.id=NULL,resampling=F
                             result.null.model=result.null.model)
result.GeneScan1D$GeneScan1D.Cauchy.pvalue
result.GeneScan1D$M
```

`GeneScan3D` *Conduct GeneScan3D analysis on the gene buffer region, integrating promoter and R enhancers.*

Description

This function conducts gene-based scan test on the gene buffer region, integrating proximal and distal regulatory elements for a gene, i.e., promoter and R enhancers. For binary traits, we conduct SPA gene-based tests to deal with imbalance case-control issues.

Usage

```
GeneScan3D(
  G = G_gene_buffer,
  Z = Z_gene_buffer,
  G.promoter = G_promoter,
  Z.promoter = Z_promoter,
  G.EnhancerAll = G_EnhancerAll,
  Z.EnhancerAll = Z_EnhancerAll,
  R = length(p_EnhancerAll),
  p_Enhancer = p_EnhancerAll,
  window.size = c(1000, 5000, 10000),
  pos = pos_gene_buffer,
  MAC.threshold = 10,
  MAF.threshold = 0.01,
  Gsub.id = NULL,
  resampling = FALSE,
  result.null.model = result.null.model
)
```

Arguments

`G`	The genotype matrix in the gene buffer region, which is a n*p matrix where n is the number of individuals and p is the number of genetic variants in the gene buffer region.
`Z`	A p*q functional annotation matrix, where p is the number of genetic variants in the gene buffer region and q is the number of functional annotations.

If Z is NULL (do not incorporate any functional annotations), the minor allele frequency weighted dispersion and/or burden tests are applied. Specifically, Beta(MAF; 1; 25) weights are used for rare variants and weights one are used for common variants.

`G.promoter` The genotype matrix for promoter, which can be NULL, that is, do not integrate promoter.

`Z.promoter` The functional annotation matrix for promoter. Z.promoter can be NULL.

`G.EnhancerAll`
The genotype matrix for R enhancers, by combining the genotype matrix of each enhancer by columns.

`Z.EnhancerAll`
The functional annotation matrix for R enhancers, by combining the functional annotation matrix of each enhancer by rows. Z.EnhancerAll can be NULL.

`R` Number of enhancers.

`p_Enhancer` Number of variants in R enhancers, which is a 1*R vector.

`window.size` The 1-D window sizes in base pairs to scan the gene buffer region. The recommended window sizes are c(1000,5000,10000).

`pos` The positions of genetic variants in the gene buffer region, an p dimensional vector. Each position corresponds to a column in the genotype matrix G and a row in the functional annotation matrix Z.

`MAC.threshold`
Threshold for minor allele count. Variants below MAC.threshold are ultra-rare variants. The recommended level is 5.

`MAF.threshold`
Threshold for minor allele frequency. Variants below MAF.threshold are rare variants. The recommended level is 0.01.

`Gsub.id` The subject id corresponding to the genotype matrix, an n dimensional vector. The default is NULL, where the matched phenotype and genotype matrices are assumed.

`resampling` Resampling indicator. The default is FALSE, do not conduct resampling-based moment matching when the sample size is large, especially for UK biobank-scale data.

`result.null.model`
The output of function "GeneScan.Null.Model()".

Value

`GeneScan3D.Cauchy.pvalue`
Cauchy combination p-values of all, common and rare variants for GeneScan3D analysis.

`M` Number of 1D scanning windows.

`minp` Minimum p-values of all, common and rare variants for 3D windows.

`RE_minp` The regulartory elements in the 3D windows corresponding to the minimum p-values, for all, common and rare variants. 0 represents promoter and a number from 1 to R represents promoter and r-th enhancer.

Examples

```
library(GeneScan3DKnock)

# Load data example
# Y: outcomes, n by 1 matrix for n=2000 individuals
# X: covariates, n by d matrix for d=1 covariate
# G_gene_buffer: genotype matrix of gene buffer region, n by p matrix, p=287 variants
# pos_gene_buffer: positions of p=287 genetic variants
# Z_gene_buffer: p by q functional annotation matrix, q=1 functional annotation
# G_promoter: 2000 by 6 genotype matrix of promoter
# Z_promoter: 6 by 1 functional annotation matrix of promoter
# G_EnhancerAll: 2000 by 86 genotype matrix of R=2 enhancers;
# Z_EnhancerAll: 86 by 1 functional annotation matrix of R=2 enhancers
# p_EnhancerAll: Number of variants for R=2 enhancers.

data("GeneScan3D.example")

Y=GeneScan3D.example$Y; X=GeneScan3D.example$X; n=length(Y)

G_gene_buffer=GeneScan3D.example$G_gene_buffer
G_promoter=GeneScan3D.example$G_promoter
G_EnhancerAll=cbind(GeneScan3D.example$G_Enhancer1,GeneScan3D.example$G_Enhancer2)

Z_gene_buffer=GeneScan3D.example$Z_gene_buffer
Z_promoter=GeneScan3D.example$Z_promoter
Z_EnhancerAll=rbind(GeneScan3D.example$Z_Enhancer1,GeneScan3D.example$Z_Enhancer2)

pos_gene_buffer=GeneScan3D.example$pos_gene_buffer
p_EnhancerAll=c(dim(GeneScan3D.example$G_Enhancer1)[2],dim(GeneScan3D.example$G_Enhancer2

# Preliminary data management
set.seed(12345)
result.null.model=GeneScan.Null.Model(Y, X, out_type="C", resampling=FALSE)

# Conduct GeneScan3D analysis
result.GeneScan3D=GeneScan3D(G=G_gene_buffer,Z=Z_gene_buffer,
                             G.promoter=G_promoter,Z.promoter=Z_promoter,
                             G.EnhancerAll=G_EnhancerAll,Z.EnhancerAll=Z_EnhancerAll,
                             R=2,p_Enhancer=p_EnhancerAll,
                             pos=pos_gene_buffer,
                             window.size=c(1000,5000,10000),MAC.threshold=10,MAF.threshold
                             result.null.model=result.null.model)
result.GeneScan3D$GeneScan3D.Cauchy.pvalue
```

`GeneScan3D.KnockoffGeneration` *GeneScan3D AR Knockoff Generation: an auto-regressive model for knockoff generation.*

Description

This function generates multiple knockoff genotypes for a gene and the corresponding regulatory elements based on an auto-regressive model. Additionally, it computes p-values from the GeneS-

can3D test for a gene based on the original data, and each of the knockoff replicates. The knockoff generations are optimized using shrinkage leveraging algorithm.

Usage

```
GeneScan3D.KnockoffGeneration(
  G_gene_buffer_surround = G_gene_buffer_surround,
  variants_gene_buffer_surround = variants_gene_buffer_surround,
  gene_buffer.pos = gene_buffer.pos,
  promoter.pos = promoter.pos,
  R = 2,
  G_EnhancerAll_surround = G_EnhancerAll_surround,
  variants_EnhancerAll_surround = variants_EnhancerAll_surround,
  p_EnhancerAll_surround = p_EnhancerAll_surround,
  Enhancer.pos = Enhancer.pos,
  p.EnhancerAll = p_EnhancerAll,
  Z = Z_gene_buffer,
  Z.promoter = Z_promoter,
  Z.EnhancerAll = Z_EnhancerAll,
  window.size = c(1000, 5000, 10000),
  MAC.threshold = 10,
  MAF.threshold = 0.01,
  Gsub.id = NULL,
  result.null.model = result.null.model,
  M = 5
)
```

Arguments

`G_gene_buffer_surround`
: The genotype matrix of the surrounding region for gene buffer region.

`variants_gene_buffer_surround`
: The genetic variants in the surrounding region for gene buffer region. Each position corresponds to a column in the genotype matrix G_gene_buffer_surround.

`gene_buffer.pos`
: The start and end positions of gene buffer region.

`promoter.pos`
: The start and end positions of promoter.

`R`
: Number of enhancers.

`G_EnhancerAll_surround`
: The genotype matrix of the surrounding regions for R enhancers, by combining the genotype matrix of the surrounding regions for each enhancer by columns.

`variants_EnhancerAll_surround`
: The genetic variants in the surrounding region for R enhancers. Each position corresponds to a column in the genotype matrix G_EnhancerAll_surround.

`p_EnhancerAll_surround`
: Number of genetic variants in the surrounding region for R enhancers, which is a 1*R vector.

`Enhancer.pos`
: The start and end positions for R enhancers. One row represents one enhancer, which is a R by 2 matrix.

`p.EnhancerAll`
: Number of genetic variants in R enhancers, which is a 1*R vector.

`Z` A p*q functional annotation matrix, where p is the number of genetic variants in the gene buffer region and q is the number of functional annotations. If Z is NULL (do not incorporate any functional annotations), the minor allele frequency weighted dispersion and/or burden tests are applied. Specifically, Beta(MAF; 1; 25) weights are used for rare variants and weights one are used for common variants.

`Z.promoter` The functional annotation matrix for promoter. Z.promoter can be NULL.

`Z.EnhancerAll` The functional annotation matrix for R enhancers, by combining the functional annotation matrix of each enhancer by rows. Z.EnhancerAll can be NULL.

`window.size` The 1-D window sizes in base pairs to scan the gene buffer region. The recommended window sizes are c(1000,5000,10000).

`MAC.threshold` Threshold for minor allele count. Variants below MAC.threshold are ultra-rare variants. The recommended level is 10.

`MAF.threshold` Threshold for minor allele frequency. Variants below MAF.threshold are rare variants. The recommended level is 0.01.

`Gsub.id` The subject id corresponding to the genotype matrix, an n dimensional vector. The default is NULL, where the matched phenotype and genotype matrices are assumed.

`result.null.model` The output of function "GeneScan.Null.Model()".

`M` Numer of multiple knockoffs.

Value

`GeneScan3D.Cauchy` GeneScan3D p-values of all, common and rare variants for original genotypes.

`GeneScan3D.Cauchy_knockoff` A M by 3 GeneScan3D p-values matrix of all, common and rare variants for M knockoff genotypes.

Examples

```
library(GeneScan3DKnock)
data(KnockoffGeneration.example)
Y=KnockoffGeneration.example$Y; X=KnockoffGeneration.example$X;

G_gene_buffer_surround=KnockoffGeneration.example$G_gene_buffer_surround
variants_gene_buffer_surround=KnockoffGeneration.example$variants_gene_buffer_surround
G_Enhancer1_surround=KnockoffGeneration.example$G_Enhancer1_surround
variants_Enhancer1_surround=KnockoffGeneration.example$variants_Enhancer1_surround
G_Enhancer2_surround=KnockoffGeneration.example$G_Enhancer2_surround
variants_Enhancer2_surround=KnockoffGeneration.example$variants_Enhancer2_surround

gene_buffer.pos=KnockoffGeneration.example$gene_buffer.pos
promoter.pos=KnockoffGeneration.example$promoter.pos
Enhancer1.pos=KnockoffGeneration.example$Enhancer1.pos
Enhancer2.pos=KnockoffGeneration.example$Enhancer2.pos

Z_gene_buffer=KnockoffGeneration.example$Z_gene_buffer
Z_promoter=KnockoffGeneration.example$Z_promoter
```

```
Z_Enhancer1=KnockoffGeneration.example$Z_Enhancer1
Z_Enhancer2=KnockoffGeneration.example$Z_Enhancer2

G_EnhancerAll_surround=cbind(G_Enhancer1_surround,G_Enhancer2_surround)
variants_EnhancerAll_surround=c(variants_Enhancer1_surround,variants_Enhancer2_surround)
p_EnhancerAll_surround=c(length(variants_Enhancer1_surround),length(variants_Enhancer2_su
Enhancer.pos=rbind(Enhancer1.pos,Enhancer2.pos)
p_EnhancerAll=c(dim(Z_Enhancer1)[1],dim(Z_Enhancer2)[1])
Z_EnhancerAll=rbind(Z_Enhancer1,Z_Enhancer2)

set.seed(12345)
result.null.model=GeneScan.Null.Model(Y, X, out_type="C", B=1000)

result.GeneScan3D.KnockoffGeneration=GeneScan3D.KnockoffGeneration(
G_gene_buffer_surround=G_gene_buffer_surround,
variants_gene_buffer_surround=variants_gene_buffer_surround,
gene_buffer.pos=gene_buffer.pos,
promoter.pos=promoter.pos,
R=2,
G_EnhancerAll_surround=G_EnhancerAll_surround,
variants_EnhancerAll_surround=variants_EnhancerAll_surround,
p_EnhancerAll_surround=p_EnhancerAll_surround,
Enhancer.pos=Enhancer.pos,
p.EnhancerAll=p_EnhancerAll,
Z=Z_gene_buffer,
Z.promoter=Z_promoter,
Z.EnhancerAll=Z_EnhancerAll,
window.size=c(1000,5000,10000),
MAC.threshold=10,
MAF.threshold=0.01,
Gsub.id=NULL,
result.null.model=result.null.model,
M=5)
result.GeneScan3D.KnockoffGeneration$GeneScan3D.Cauchy
result.GeneScan3D.KnockoffGeneration$GeneScan3D.Cauchy_knockoff
```

`GeneScan3DKnock` *GeneScan3DKnock: Knockoff-enhanced gene-based test for causal gene discovery (knockoff filter).*

Description

This function performs the knockoff filter, and computes the q-value for each gene. This function takes the results from the GeneScan3D.KnockoffGeneration() function and get knockoff statistics and q-values.

Usage

```
GeneScan3DKnock(
  M = 5,
  p0 = GeneScan3DKnock.example$GeneScan3D.original,
  p_ko = cbind(GeneScan3DKnock.example$GeneScan3D.ko1,
    GeneScan3DKnock.example$GeneScan3D.ko2, GeneScan3DKnock.example$GeneScan3D.k
    GeneScan3DKnock.example$GeneScan3D.ko4, GeneScan3DKnock.example$GeneScan3D.k
```

```
  fdr = 0.1,
  gene_id = GeneScan3DKnock.example$gene.id
)
```

Arguments

M	Number of multiple knockoffs.
p0	A N-dimensional vector of the original GeneScan3D p-values, calculated using GeneScan3D.KnockoffGeneration() function.
p_ko	A N*M matrix of M knockoff GeneScan3D p-values, calculated using GeneScan3D.KnockoffGeneration() function.
fdr	The false discovery rate (FDR) threshold. The default is 0.1.
gene_id	The genes id for N genes considered in the analysis. Usually we consider N=~20,000 protein-coding genes.

Value

W	The knockoff statistics for each gene.
W.threshold	Threshold of W statistics. A gene is significant under GeneScan3DKnock if W>W.threshold or equivalently, q-value<fdr threshold. There is no significant gene if W.threshold=Inf.
Qvalue	The q-values for each gene.
gene_sign	Significant genes with q-values less then the fdr threshold.

Examples

```
library(GeneScan3DKnock)

# Load data example
data("GeneScan3DKnock.example")

result.GeneScan3DKnock=GeneScan3DKnock(M=5,
p0=GeneScan3DKnock.example$GeneScan3D.original,
p_ko=cbind(GeneScan3DKnock.example$GeneScan3D.ko1,
          GeneScan3DKnock.example$GeneScan3D.ko2,
          GeneScan3DKnock.example$GeneScan3D.ko3,
          GeneScan3DKnock.example$GeneScan3D.ko4,
          GeneScan3DKnock.example$GeneScan3D.ko5),fdr = 0.1,
          gene_id=GeneScan3DKnock.example$gene.id)
result.GeneScan3DKnock$W
result.GeneScan3DKnock$W.threshold
result.GeneScan3DKnock$Qvalue
result.GeneScan3DKnock$gene_sign
```

Index