Package 'epistasisGA'

November 9, 2026

Type Package

Title An R package to identify multi-snp effects in nuclear family studies using the GADGETS method

Version 1.14.0

Description This package runs the GADGETS method to identify epistatic effects in nuclear family studies. It also provides functions for permutation-based inference and graphical visualization of the results.

License GPL-3

Encoding UTF-8

Imports BiocParallel, data.table, matrixStats, stats, survival, igraph, batchtools, qgraph, grDevices, parallel, ggplot2, grid, bigmemory, graphics, utils

Suggests BiocStyle, knitr, rmarkdown, magrittr, kableExtra, testthat (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo, BH, bigmemory

RoxygenNote 7.2.2

Depends R (>= 4.2)

biocViews Genetics, SNP, GeneticVariability

VignetteBuilder knitr

URL https://github.com/mnodzenski/epistasisGA

BugReports https://github.com/mnodzenski/epistasisGA/issues

Config/testthat/edition 3

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

git_branch RELEASE_3_23

git_last_commit 43510a0

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-08

Author Michael Nodzenski [aut, cre],
Juno Krahn [ctb]

Maintainer Michael Nodzenski <michael.nodzenski@gmail.com>

Contents

case . . . 2
case.gxe . . . 3
case.mci . . . 3
chrom.fitness.score . . . 4
combine.islands . . . 6
compute.graphical.scores . . . 8
dad . . . 11
dad.gxe . . . 11
dad.mci . . . 12
epistasis.test . . . 12
epistasisGA . . . 14
exposure . . . 14
GADGETS . . . 15
global.test . . . 19
GxE.fitness.score . . . 24
GxE.test . . . 26
mom . . . 28
mom.gxe . . . 28
mom.mci . . . 29
network.plot . . . 29
permute.dataset . . . 32
preprocess.genetic.data . . . 33
run.gadgets . . . 37
snp.annotations . . . 41
snp.annotations.mci . . . 41

Index **42**

`case` *Genotypes for the affected children of case-parent triads.*

Description

A simulated dataset containing the counts of the alternate allele for 100 SNPs for the affected child in 1000 simulated case-parent triads. Columns represent SNPs, rows are individuals. SNPs in columns 51, 52, 76, and 77 represent a true risk pathway.

Usage

```
data(case)
```

Format

A data frame with 1000 rows and 100 variables

`case.gxe` *Genotypes for the cases of case-parent triads with a simulated gene environment interaction.*

Description

A simulated dataset containing the counts of the alternate allele for 24 SNPs for the cases in 1000 simulated case-parent triads. Columns represent SNPs, rows are individuals. SNPs in columns 6, 12, and 18 represent a simulated risk pathway, where, in the child, at least one copy of the alternate allele for each path SNP in addition to exposure 1 confers increased disease risk. .

A simulated dataset containing the counts of the alternate allele for 24 SNPs for the cases in 1000 simulated case-parent triads. Columns represent SNPs, rows are individuals. SNPs in columns 6, 12, and 18 represent a simulated risk pathway, where, in the child, at least one copy of the alternate allele for each path SNP in addition to exposure 1 confers increased disease risk. .

Usage

```
data(case.gxe)
```

```
data(case.gxe)
```

Format

A data frame with 1000 rows and 24 variables

A data frame with 1000 rows and 24 variables

`case.mci` *Genotypes for the affected cases of case-parent triads with a simulated maternal-fetal interaction.*

Description

A simulated dataset containing the counts of the alternate allele for 24 SNPs for the cases in 1000 simulated case-parent triads. Columns represent SNPs, rows are individuals. The SNP in column 6 of the corresponding maternal dataset `mom.mci` interacts with the SNPs in columns 12 and 18 of `case.mci` to increase risk of disease in the child, where at least one copy of the alternate allele (genotype 1 or 2) is required at each implicated locus. .

Usage

```
data(case.mci)
```

Format

A matrix with 1000 rows and 24 variables

`chrom.fitness.score` *A function to assign a fitness score to a chromosome*

Description

This function assigns a fitness score to a chromosome. It is a wrapper for the Rcpp function chrom_fitness_score.

Usage

```
chrom.fitness.score(
  case.genetic.data,
  complement.genetic.data,
  target.snps,
  ld.block.vec,
  weight.lookup,
  n.different.snps.weight = 2,
  n.both.one.weight = 1,
  recessive.ref.prop = 0.75,
  recode.test.stat = 1.64,
  epi.test = FALSE
)
```

Arguments

`case.genetic.data`

The genetic data of the disease affected children from case-parent trios or disease-discordant sibling pairs. If searching for maternal SNPs that are related to risk of disease in the child, some of the columns in `case.genetic.data` may contain maternal SNP genotypes (See argument `mother.snps` for how to indicate which SNPs columns correspond to maternal genotypes). Columns are SNP allele counts, and rows are individuals. This object may either be of class matrix' OR of class 'big.matrix'. If of class 'big.matrix' it must be file backed as type 'integer' (see the `bigmemory` package for more information). The ordering of the columns must be consistent with the LD structure specified in `ld.block.vec`. The genotypes cannot be dosages imputed with uncertainty.

`complement.genetic.data`

A genetic dataset for the controls corresponding to the genotypes in `case.genetic.data`.For SNPs that correspond to the affected child in `case.genetic.data`, the corresponding column in `complement.genetic.data` should be set equal to mother allele count + father allele count - case allele count. If using disease-discordant siblings this argument should be the genotypes for the unaffected siblings. For SNPs in `case.genetic.data` that represent maternal genotypes (if any) the

corresponding column in complement.genetic.data should be the paternal genotypes for that SNP. Regardless, complement.genetic.data may be an object of either class matrix' OR of class 'big.matrix'. If of class 'big.matrix' it must be file backed as type 'integer' (see the bigmemory package for more information). Columns are SNP allele counts, rows are families. If not specified, father.genetic.data and mother.genetic.data must be specified. The genotypes cannot be dosages imputed with uncertainty.

target.snps An integer vector of the columns corresponding to the collection of SNPs, or chromosome, for which the fitness score will be computed.

ld.block.vec An integer vector specifying the linkage blocks of the input SNPs. As an example, for 100 candidate SNPs, suppose we specify ld.block.vec <- c(25, 75, 100). This vector indicates that the input genetic data has 3 distinct linkage blocks, with SNPs 1-25 in the first linkage block, 26-75 in the second block, and 76-100 in the third block. Note that this means the ordering of the columns (SNPs) in case.genetic.data must be consistent with the LD blocks specified in ld.block.vec. In the absence of outside information, a reasonable default is to consider SNPs to be in LD if they are located on the same biological chromosome. If case.genetic.data includes both maternal and child SNP genotypes, we recommend considering any maternal SNP and any child SNP located on the same nominal biological chromosome as 'in linkage'. E.g., we recommend considering any maternal SNPs located on chromosome 1 as being 'linked' to any child SNPs located on chromosome 1, even though, strictly speaking, the maternal and child SNPs are located on separate pieces of DNA.If not specified, this defaults to assuming all input SNPs are in linkage, which may be overly conservative and could adversely affect performance.

weight.lookup A vector that maps a family weight to the weighted sum of the number of different SNPs and SNPs both equal to one.

n.different.snps.weight
The number by which the number of different SNPs between a case and complement/unaffected sibling is multiplied in computing the family weights. Defaults to 2.

n.both.one.weight
The number by which the number of SNPs equal to 1 in both the case and complement/unaffected sibling is multiplied in computing the family weights. Defaults to 1.

recessive.ref.prop
The proportion to which the observed proportion of informative cases with the provisional risk genotype(s) will be compared to determine whether to recode the SNP as recessive. Defaults to 0.75.

recode.test.stat
For a given SNP, the minimum test statistic required to recode and recompute the fitness score using recessive coding. Defaults to 1.64.

epi.test A logical indicating whether the function should return the information required to run function epistasis.test for a given SNP-set.

Value

A list:

fitness_score The chromosome fitness score.

sum_dif_vecs The weighted mean difference vector corresponding to the chromosome, with each element divided by it's pseudo-standard error. The magnitudes of these values are not particularly important, but the sign is useful. A positive value for a given SNP indicates the minor allele is positively associated with disease status, while a negative value implies the reference ('wild type') allele is positively associated with the disease.

q The number of cases with a risk-related genotype at each locus over the total number of cases or controls that have a full set of risk genotypes at each locus, among families where only one of the case or control has the full risk set.

risk_set_alleles A vector indicating the number risk alleles a case or complement must have for each SNP in `target.snps` for the case or complement to be classified as having the proposed risk set. '1+' indicates at least one copy of the risk allele is required, while '2' indicates 2 copies are needed. The risk allele can be determined based on the signs of the elements of `sum_dif_vecs`, where a negative value indicates the major allele for a given SNP is the risk allele, while a positive value implicates the minor allele.

inf_families An integer vector of the informative family rows. Only returned if `epi.test` = TRUE.

Examples

```
data(case)
data(dad)
data(mom)
case <- as.matrix(case)
dad <- as.matrix(dad)
mom <- as.matrix(mom)
comp <- mom + dad - case
weight.lookup <- vapply(seq_len(6), function(x) 2^x, 1)
storage.mode(weight.lookup) <- "integer"
block.ld.vec <- cumsum(rep(25, 4))
chrom.fitness.score(case, comp, c(1, 4, 7),
                    block.ld.vec, weight.lookup)
```

`combine.islands` *A function to combine GADGETS results for individual islands into a single dataset.*

Description

This function combines GADGETS results for individual islands into a single dataset.

Usage

```
combine.islands(
  results.dir,
  annotation.data,
  preprocessed.list,
  n.top.chroms.per.island = 1
)
```

Arguments

`results.dir`	The directory in which individual island results from `run.gadgets` are saved.
`annotation.data`	A data frame containing columns 'RSID', 'REF' and 'ALT'. Column 'RSID' gives the RSIDs for the input SNPs, with the rows ordered such that the first RSID entry corresponds to the first SNP column in the data passed to function `preprocess.genetic.data`, the second RSID corresponds to the second SNP column, etc.
`preprocessed.list`	The initial list produced by function `preprocess.genetic.data`.
`n.top.chroms.per.island`	The number of top chromosomes per island to save in the final combined list. Defaults to the single top chromosome.

Value

A data.table containing the results aggregated across islands. Note these results be written to `results.dir` as combined.island.unique.chromosome.results.rds'. See the package vignette for more detailed descriptions of the content of each output column. Secondarily, this will concatenate all individual island results files and store them in a single file, called "all.island.results.concatenated.rds".

Examples

```
data(case)
data(dad)
data(mom)
data(snp.annotations)

pp.list <- preprocess.genetic.data(as.matrix(case[, 1:10]),
                                   father.genetic.data = as.matrix(dad[ , 1:10]),
                                   mother.genetic.data = as.matrix(mom[ , 1:10]),
                                   ld.block.vec = c(10))

run.gadgets(pp.list, n.chromosomes = 4, chromosome.size = 3,
       results.dir = 'tmp',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)

combined.res <- combine.islands('tmp', snp.annotations[ 1:10, ], pp.list)

unlink("tmp", recursive = TRUE)
unlink("tmp_reg", recursive = TRUE)
```

compute.graphical.scores *A function to compute SNP-pair scores for network plots of results.*

Description

This function returns a data.table of graphical SNP-pair scores for use in network plots of GADGETS results.

Usage

```
compute.graphical.scores(
  results.list,
  preprocessed.list,
  score.type = "logsum",
  pval.thresh = 0.05,
  n.permutes = 10000,
  n.different.snps.weight = 2,
  n.both.one.weight = 1,
  weight.function.int = 2,
  recessive.ref.prop = 0.75,
  recode.test.stat = 1.64,
  bp.param = bpparam(),
  null.mean.vec.list = NULL,
  null.sd.vec.list = NULL
)
```

Arguments

results.list	A list of length d, where d is the number of chromosome sizes to be included in the network plot. Each element of the list should be a data.table from combine.islands for a given chromosome size. Each data.table in the list should be subset to only include those chromosomes whose fitness scores are high enough to contribute to the network plot. The selection of the chromosomes that contribute to these plots is at the analyst's discretion. We have found success in just using the top 10 scoring chromosomes, and also by restricting attention to those chromosomes that exceed the 95th percentile of the maxima observed after running GADGETS on data-sets permuted under a global, no-association null. For the latter, see also function global.test.
preprocessed.list	The list output by preprocess.genetic.data run on the observed data.
score.type	A character string specifying the method for aggregating SNP-pair scores across chromosome sizes. Options are 'max', 'sum', or 'logsum', defaulting to 'logsum'. For a given SNP-pair, it's graphical score will be the score.type of all graphical scores of chromosomes containing that pair across chromosome sizes. The choice of 'logsum' rather than 'sum' may be useful in cases where there are

multiple risk-sets, and one is found much more frequently. However, it may be of interest to examine plots using both `score.type` approaches.

`pval.thresh` A numeric value between 0 and 1 specifying the epistasis test p-value threshold for a chromosome to contribute to the network. Any chromosomes with epistasis p-value greater than `pval.thresh` will not contribute to network plots. The argument defaults to 0.05. It must be <= 0.6.

`n.permutes` The number of permutations on which to base the epistasis tests. Defaults to 10000.

`n.different.snps.weight`
The number by which the number of different SNPs between a case and complement/unaffected sibling is multiplied in computing the family weights. Defaults to 2.

`n.both.one.weight`
The number by which the number of SNPs equal to 1 in both the case and complement/unaffected sibling is multiplied in computing the family weights. Defaults to 1.

`weight.function.int`
An integer used to assign family weights. Specifically, we use `weight.function.int` in a function that takes the weighted sum of the number of different SNPs and SNPs both equal to one as an argument, denoted as x, and returns a family weight equal to `weight.function.int`^x. Defaults to 2.

`recessive.ref.prop`
The proportion to which the observed proportion of informative cases with the provisional risk genotype(s) will be compared to determine whether to recode the SNP as recessive. Defaults to 0.75.

`recode.test.stat`
For a given SNP, the minimum test statistic required to recode and recompute the fitness score using recessive coding. Defaults to 1.64.

`bp.param` The BPPARAM argument to be passed to bplapply. See `BiocParallel::bplapply` for more details.

`null.mean.vec.list`
(experimental) A list, equal in length to `results.list`, where the i^th element of the list is the vector of null means (stored in the 'null.mean.sd.info.rds') corresponding to the d (chromosome size) used to generate the results stored in the i^th element of `results.list`. This only needs to be specified if based on the experimental E-GADGETS method, and otherwise can be left at its default.

`null.sd.vec.list`
(experimental) A list, equal in length to `results.list`, where the i^th element of the list is the vector of null standard deviations (stored in the 'null.mean.sd.info.rds') corresponding to the d (chromosome size) used to generate the results stored in the i^th element of `results.list`. This only needs to be specified if based on the experimental E-GADGETS method, and otherwise can be left at its default.

Value

A list of two elements:

pair.scores A data.table containing SNP-pair graphical scores, where the first four columns represent SNPs and the fifth column (pair.score) is the graphical SNP-pair score.

snp.scores A data.table containing individual SNP graphical scores, where the first two columns represent SNPs and the third column (snp.score) is the graphical SNP score.

Examples

```
data(case)
data(dad)
data(mom)
data(snp.annotations)
set.seed(1400)

# preprocess data
target.snps <- c(1:3, 30:32, 60:62, 85)
preprocessed.list <- preprocess.genetic.data(as.matrix(case[, target.snps]),
                        father.genetic.data = as.matrix(dad[ , target.snps]),
                        mother.genetic.data = as.matrix(mom[ , target.snps]),
                      ld.block.vec = c(3, 3, 3, 1))
## run GA for observed data

#observed data chromosome size 2
run.gadgets(preprocessed.list, n.chromosomes = 5, chromosome.size = 2,
       results.dir = 'tmp_2',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)
 combined.res2 <- combine.islands('tmp_2',
                     snp.annotations[ target.snps, ], preprocessed.list, 2)
 unlink('tmp_reg', recursive = TRUE)

 #observed data chromosome size 3
 run.gadgets(preprocessed.list, n.chromosomes = 5,
       chromosome.size = 3, results.dir = 'tmp_3',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)
 combined.res3 <- combine.islands('tmp_3', snp.annotations[ target.snps, ],
                                  preprocessed.list, 2)
 unlink('tmp_reg', recursive = TRUE)

## create list of results

final.results <- list(combined.res2[1:3, ], combined.res3[1:3, ])

 ## compute edge scores
 edge.dt <- compute.graphical.scores(final.results,
                                     preprocessed.list,
                                     pval.thresh = 0.5)
```

```
lapply(c("tmp_2", "tmp_3"), unlink, recursive = TRUE)
```

---

dad *Genotypes for the fathers of case-parent triads.*

Description

A simulated dataset containing the counts of the alternate allele for 100 SNPs for the fathers in 1000 simulated case-parent triads. Columns represent SNPs, rows are individuals. SNPs in columns 51, 52, 76, and 77 represent a true risk pathway.

Usage

```
data(dad)
```

Format

A data frame with 1000 rows and 100 variables

---

dad.gxe *Genotypes for the fathers of case-parent triads with a simulated gene environment interaction.*

Description

A simulated dataset containing the counts of the alternate allele for 24 SNPs for the fathers in 1000 simulated case-parent triads. Columns represent SNPs, rows are individuals. SNPs in columns 6, 12, and 18 represent a simulated risk pathway, where, in the child, at least one copy of the alternate allele for each path SNP in addition to exposure 1 confers increased disease risk. .

A simulated dataset containing the counts of the alternate allele for 24 SNPs for the fathers in 1000 simulated case-parent triads. Columns represent SNPs, rows are individuals. SNPs in columns 6, 12, and 18 represent a simulated risk pathway, where, in the child, at least one copy of the alternate allele for each path SNP in addition to exposure 1 confers increased disease risk. .

Usage

```
data(dad.gxe)

data(dad.gxe)
```

Format

A data frame with 1000 rows and 24 variables

A data frame with 1000 rows and 24 variables

dad.mci

Genotypes for the fathers of case-parent triads with a simulated maternal-fetal interaction.

Description

A simulated dataset containing the counts of the alternate allele for 24 SNPs for the fathers in 1000 simulated case-parent triads. Columns represent SNPs, rows are individuals. The SNP in column 6 of the corresponding maternal dataset `mom.mci` interacts with the SNPs in columns 12 and 18 of the corresponding child dataset `case.mci` to increase risk of disease in the child, where at least one copy of the alternate allele (genotype 1 or 2) is required at each implicated locus. .

Usage

```
data(dad.mci)
```

Format

A matrix with 1000 rows and 24 variables

epistasis.test

A function to run a test of the null hypothesis that a collection of SNPs do not exhibit epistasis, conditional upon observed marginal SNP-disease associations.

Description

This function runs a permutation based test of the null hypothesis that a collection of SNPs do not exhibit epistasis, conditional upon observed marginal SNP-disease associations.

Usage

```
epistasis.test(
  snp.cols,
  preprocessed.list,
  n.permutes = 10000,
  n.different.snps.weight = 2,
  n.both.one.weight = 1,
  weight.function.int = 2,
  recessive.ref.prop = 0.75,
  recode.test.stat = 1.64,
  maternal.fetal.test = FALSE
)
```

Arguments

`snp.cols`	An integer vector specifying the columns in the input data containing the SNPs to be tested.
`preprocessed.list`	The initial list produced by function `preprocess.genetic.data`.
`n.permutes`	The number of permutations on which to base the test. Defaults to 10000.
`n.different.snps.weight`	The number by which the number of different SNPs between a case and complement/unaffected sibling is multiplied in computing the family weights. Defaults to 2.
`n.both.one.weight`	The number by which the number of SNPs equal to 1 in both the case and complement/unaffected sibling is multiplied in computing the family weights. Defaults to 1.
`weight.function.int`	An integer used to assign family weights. Specifically, we use `weight.function.int` in a function that takes the weighted sum of the number of different SNPs and SNPs both equal to one as an argument, denoted as x, and returns a family weight equal to `weight.function.int`^x. Defaults to 2.
`recessive.ref.prop`	The proportion to which the observed proportion of informative cases with the provisional risk genotype(s) will be compared to determine whether to recode the SNP as recessive. Defaults to 0.75.
`recode.test.stat`	For a given SNP, the minimum test statistic required to recode and recompute the fitness score using recessive coding. Defaults to 1.64.
`maternal.fetal.test`	A boolean indicating whether the test specifically for a maternal-fetal interaction should be run. Defaults to FALSE.

Value

A list of thee elements:

pval The p-value of the test. (In GADGETS papers, these are instead referred to as h-values)

obs.fitness.score The fitness score from the observed data

perm.fitness.scores A vector of fitness scores for the permuted datasets.

Examples

```
data(case)
data(dad)
data(mom)
data(snp.annotations)

pp.list <- preprocess.genetic.data(as.matrix(case),
                                   father.genetic.data = as.matrix(dad),
                                   mother.genetic.data = as.matrix(mom),
```

```
                                ld.block.vec = rep(25, 4))

run.gadgets(pp.list,
    n.chromosomes = 5, chromosome.size = 3,
    results.dir = "tmp", cluster.type = "interactive",
    registryargs = list(file.dir = "tmp_reg", seed = 1300),
    n.islands = 8, island.cluster.size = 4,
    n.migrations = 2
)

combined.res <- combine.islands("tmp", snp.annotations, pp.list, 2)

top.snps <- as.vector(t(combined.res[1, 1:3]))
set.seed(10)
epi.test.res <- epistasis.test(top.snps, pp.list)

unlink('tmp', recursive = TRUE)
unlink('tmp_reg', recursive = TRUE)
```

`epistasisGA` `epistasisGA` *package*

Description

A package implementing the GADGETS method to detect multi-SNP effects in case-parent triad or affected/unaffected sibling studies.

`exposure` *Exposures for the cases of case-parent triads with a simulated gene environment interaction.*

Description

A data.frame containing simulated exposure status for each case of the case-parent triads data. Rows correspond to different families. The single column represents a binary exposure, where in combination with the relevant risk-associated alleles (columns 6, 12, and 18 in data set case.gxe), is associated with increased risk. .

Usage

```
data(exposure)
```

Format

A data frame with 1000 rows and 1 variables

GADGETS *A function to run the GADGETS method*

Description

This function runs the GADGETS method on a given cluster of islands. It is a wrapper for the underlying Rcpp function run_GADGETS.

Usage

```
GADGETS(
  cluster.number,
  results.dir,
  case.genetic.data,
  complement.genetic.data,
  case.genetic.data.n,
  mother.genetic.data.n,
  father.genetic.data.n,
  exposure.mat,
  weight.lookup.n,
  ld.block.vec,
  n.chromosomes,
  chromosome.size,
  snp.chisq,
  weight.lookup,
  null.mean.vec = c(0, 0),
  null.se.vec = c(1, 1),
  island.cluster.size = 4,
  n.migrations = 20,
  n.different.snps.weight = 2,
  n.both.one.weight = 1,
  migration.interval = 50,
  gen.same.fitness = 50,
  max.generations = 500,
  initial.sample.duplicates = FALSE,
  crossover.prop = 0.8,
  recessive.ref.prop = 0.75,
  recode.test.stat = 1.64,
  E_GADGETS = FALSE,
  continuous.exposure = FALSE,
  exposure.min.max = c(0, 0)
)
```

Arguments

cluster.number	An integer indicating the cluster number (used for labeling the output file).
results.dir	The directory to which island results will be saved.

case.genetic.data
: The genetic data of the disease affected children from case-parent trios or disease-discordant sibling pairs. If searching for maternal SNPs that are related to risk of disease in the child, some of the columns in case.genetic.data may contain maternal SNP genotypes (See argument mother.snps for how to indicate which SNPs columns correspond to maternal genotypes). Columns are SNP allele counts, and rows are individuals.This object should be of class 'matrix'. The ordering of the columns must be consistent with the LD structure specified in ld.block.vec. The genotypes cannot be dosages imputed with uncertainty. If any data are missing for a particular family member for a particular SNP, that SNP's genotype should be coded as -9 for each member of the entire family: (case.genetic.data and father.genetic.data/mother.genetic.data, or case.genetic.data and complement.genetic.data). If running the experimental E-GADGETS method, this argument should be set to a 1x1 matrix whose only value is 0.0, and case.genetic.data.n will be used to specify the case genotypes.

complement.genetic.data
: A genetic dataset for the controls corresponding to the genotypes in case.genetic.data.For SNPs that correspond to the affected child in case.genetic.data, the corresponding column in complement.genetic.data should be set equal to mother allele count + father allele count - case allele count. If using disease-discordant siblings this argument should be the genotypes for the unaffected siblings. For SNPs in case.genetic.data that represent maternal genotypes (if any) the corresponding column in complement.genetic.data should be the paternal genotypes for that SNP. This object should be of class 'matrix'. Columns are SNP allele counts, rows are families. If not specified, father.genetic.data and mother.genetic.data must be specified. The genotypes cannot be dosages imputed with uncertainty. If any data are missing for a particular family for a particular SNP, that SNP's genotype should be coded as -9 for the entire family (case.genetic.data and complement.genetic.data) for that SNP. If running the experimental E-GADGETS method, this argument should be set to a 1x1 matrix whose only value is 0.0, and complement.genetic.data.n will be used to specify the complement genotypes.

case.genetic.data.n
: (experimental) A matrix, to be used in the experimental E-GADGETS method, containing the same data as described above for case.genetic.data, but the genotypes here are stored as floating point values, as opposed to integer values in case.genetic.data. If not running E-GADGETS, this should be specified as a 1x1 matrix whose only value is 0.0.

mother.genetic.data.n
: (experimental) A matrix, to be used in the experimental E-GADGETS method, containing the genotypes for the mothers of case.genetic.data, where the genotypes are stored as floating point values, as opposed to integer values. If not running E-GADGETS, this should be specified as a 1x1 matrix whose only value is 0.0.

father.genetic.data.n
: (experimental) A matrix, to be used in the experimental E-GADGETS method, containing the genotypes for the fathers of case.genetic.data, where the

genotypes are stored as floating point values, as opposed to integer values. If not running E-GADGETS, this should be specified as a 1x1 matrix whose only value is 0.0.

`exposure.mat` (experimental) A matrix of the input categorical and continuous exposures, if specified, to be used in the experimental E-GADGETS method. If not running E-GADGETS, this should be a 1x1 matrix whose only entry is 0.0.

`weight.lookup.n`
(experimental) A vector that maps a family weight to the weighted sum of the number of different SNPs and SNPs both equal to one, to be used by the experimetnal E-GADGETS method. The vector should store values as floating point values, not integers. If not running E-GADGETS, this argument should be specified as 0.0, and will not be used in the GA. Instead, for GADGETS, computation of the family weights will be based on argument `weight.lookup`, which is computed in the same way, except stores values as integers.

`ld.block.vec` An integer vector specifying the linkage blocks of the input SNPs. As an example, for 100 candidate SNPs, suppose we specify `ld.block.vec <- c(25, 75, 100)`. This vector indicates that the input genetic data has 3 distinct linkage blocks, with SNPs 1-25 in the first linkage block, 26-75 in the second block, and 76-100 in the third block. Note that this means the ordering of the columns (SNPs) in `case.genetic.data` must be consistent with the LD blocks specified in `ld.block.vec`. In the absence of outside information, a reasonable default is to consider SNPs to be in LD if they are located on the same biological chromosome. If `case.genetic.data` includes both maternal and child SNP genotypes, we recommend considering any maternal SNP and any child SNP located on the same nominal biological chromosome as 'in linkage'. E.g., we recommend considering any maternal SNPs located on chromosome 1 as being 'linked' to any child SNPs located on chromosome 1, even though, strictly speaking, the maternal and child SNPs are located on separate pieces of DNA. If running E-GADGETS, this argument should be specified as 0, and will not be used.

`n.chromosomes` An integer specifying the number of chromosomes to use in the GA.

`chromosome.size`
An integer specifying the number of SNPs on each chromosome.

`snp.chisq` A vector of statistics to be used in sampling SNPs for mutation. By default, these are the square roots of the chi-square marginal SNP-disease association statistics for each column in `case.genetic.data`, but can also be manually specified or uniformly 1 (corresponding to totally random sampling).

`weight.lookup` A vector that maps a family weight to the weighted sum of the number of different SNPs and SNPs both equal to one. This should store values as integers.

`null.mean.vec` (experimental) A vector of estimated null means for each component of the E-GADGETS (GxGxE) fitness score. For all other uses, this should be specified as rep(0, 2) and will not be used.

`null.se.vec` (experimental) A vector of estimated null standard deviations for each component of the E-GADGETS (GxGxE) fitness score. For all other uses, this should be specified as rep(0, 2) and will not be used.

`island.cluster.size`
An integer specifying the number of islands in the cluster. See `run.gadgets` for additional details.

`n.migrations` The number of chromosomes that migrate among islands. This value must be less than `n.chromosomes` and greater than 0, defaulting to 20.

`n.different.snps.weight`
The number by which the number of different SNPs between a case and complement is multiplied in computing the family weights. Defaults to 2.

`n.both.one.weight`
The number by which the number of SNPs equal to 1 in both the case and complement is multiplied in computing the family weights. Defaults to 1.

`migration.interval`
The interval of generations for which GADGETS will run prior to migration of top chromosomes among islands in a cluster. Defaults to 50. In other words, top chromosomes will migrate among cluster islands every `migration.interval` generations. We also check for convergence at each of these intervals.

`gen.same.fitness`
The number of consecutive generations with the same fitness score required for algorithm termination. Defaults to 50.

`max.generations`
The maximum number of generations for which GADGETS will run. Defaults to 500.

`initial.sample.duplicates`
A logical indicating whether the same SNP can appear in more than one chromosome in the initial sample of chromosomes (the same SNP may appear in more than one chromosome thereafter, regardless). Defaults to FALSE.

`crossover.prop` A numeric between 0 and 1 indicating the proportion of chromosomes to be subjected to cross over. The remaining proportion will be mutated. Defaults to 0.8.

`recessive.ref.prop`
The proportion to which the observed proportion of informative cases with the provisional risk genotype(s) will be compared to determine whether to recode the SNP as recessive. Defaults to 0.75.

`recode.test.stat`
For a given SNP, the minimum test statistic required to recode and recompute the fitness score using recessive coding. Defaults to 1.64. See the GADGETS paper for specific details.

`E_GADGETS` (experimental) A boolean indicating whether to run the experimental 'E_GADGETS' method.

`continuous.exposure`
A boolean indicating whether, for E-GADGETS, the exposure is continuous.

`exposure.min.max`
A vector, to be used in E-GADGETS when the exposure is continuous, whose first element is the minimum observed exposure value and second element is the maximum observed exposure value.

top.chromosome.results A data.table of the final generation chromosomes, their fitness scores, and, for GADGETS, additional information pertaining to nominated risk-related genotypes. See the package vignette for an example and for additional details.

n.generations The total number of generations run.

Value

For each island in the cluster, an rds object containing a list with the following elements will be written to `results.dir`.

Examples

```
set.seed(10)
data(case)
case <- as.matrix(case)
data(dad)
dad <- as.matrix(dad)
data(mom)
mom <- as.matrix(mom)
data.list <- preprocess.genetic.data(case[, 1:10],
                                     father.genetic.data = dad[ , 1:10],
                                     mother.genetic.data = mom[ , 1:10],
                                     ld.block.vec = c(10))

 chisq.stats <- sqrt(data.list$chisq.stats)
 ld.block.vec <- data.list$ld.block.vec
 case.genetic.data <- data.list$case.genetic.data
 complement.genetic.data <- data.list$complement.genetic.data

 #required inputs but not actually used in function below
 case.genetic.data.n <- matrix(0.0, 1, 1)
 mother.genetic.data.n <- matrix(0.0, 1, 1)
 father.genetic.data.n <- matrix(0.0, 1, 1)
 exposure.mat <- data.list$exposure.mat + 0.0

 weight.lookup <- vapply(seq_len(6), function(x) 2^x, 1)
 dir.create('tmp')
GADGETS(cluster.number = 1, results.dir = 'tmp',
        case.genetic.data = case.genetic.data,
        complement.genetic.data = complement.genetic.data,
        case.genetic.data.n = case.genetic.data.n,
        mother.genetic.data.n = mother.genetic.data.n,
        father.genetic.data.n = father.genetic.data.n,
        exposure.mat = exposure.mat,
        weight.lookup.n = weight.lookup + 0.0,
        ld.block.vec = ld.block.vec,
        n.chromosomes = 10, chromosome.size = 3, snp.chisq = chisq.stats,
        weight.lookup = weight.lookup, n.migrations = 2,
        migration.interval = 5,
        gen.same.fitness = 10, max.generations = 10)
```

`global.test` *A function to run a global test of the null hypothesis that there are no SNP-disease associations across a range of chromosome sizes*

Description

This function runs a global test of the null hypothesis that there are no SNP-disease associations across a range of chromosome sizes

Usage

```
global.test(results.list, n.top.scores = 10)
```

Arguments

`results.list`	A list of length d, where d is the number of chromosome sizes to be included in a global test. Each element of the list must itself be a list whose first element `observed.data` is a vector of fitness scores from `combine.islands` for the observed data for a given chromosome size. The second element `permutation.list` is a list containing vectors of all permutation results fitness scores, again using the results output by `combine.islands` for each permutation.
`n.top.scores`	The number of top scoring chromosomes, for each chromosome size, to be used in calculating the global test. Defaults to 10.

Value

A list containing the following:

obs.test.stat The observed test statistic.

perm.test.stats A vector of test statistics from permuted data.

pval The p-value for the global test.

obs.marginal.test.stats A vector of observed test statistics for each chromosome size.

perm.marginal.test.stats.mat A matrix of test statistics for the permutation datasets, where rows correspond to permutations and columns correspond to chromosome sizes.

marginal.pvals A vector containing marignal p-values for each chromosome size.

max.obs.fitness A vector of the maximum fitness score for each chromosome size in the observed data.

max.perm.fitness A list of vectors for each chromosome size of maximum observed fitness scores for each permutation.

max.order.pvals A vector of p-values for the maximum observed order statistics for each chromosome size. P-values are the proportion of permutation based maximum order statistics that exceed the observed maximum fitness score.

boxplot.grob A grob of a ggplot plot of the observed vs permuted fitness score densities for each chromosome size.

chrom.size.k A vector indicating the number of top scores (k) from each chromosome size that the test used. This will be equal to `n.top.scores` unless GADGETS returns fewer than `n.top.scores` unique chromosomes for the observed data or any permute, in which case the chromosome size-specific value will be equal to the smallest number of unique chromosomes returned.

max.perm.95th.pctl The 95th percentile of the permutation maximum order statistics for each chromosome size.

Examples

```
data(case)
data(dad)
data(mom)
case <- as.matrix(case)
dad <- as.matrix(dad)
mom <- as.matrix(mom)
data(snp.annotations)
set.seed(1400)

pp.list <- preprocess.genetic.data(case[, 1:10],
                                   father.genetic.data = dad[ , 1:10],
                                   mother.genetic.data = mom[ , 1:10],
                                   ld.block.vec = c(10))
## run GA for observed data

#observed data chromosome size 2
run.gadgets(pp.list, n.chromosomes = 5, chromosome.size = 2,
       results.dir = 'tmp_2',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)
 combined.res2 <- combine.islands('tmp_2', snp.annotations[ 1:10, ],
                                  pp.list, 2)
 unlink('tmp_reg', recursive = TRUE)

 #observed data chromosome size 3
 run.gadgets(pp.list, n.chromosomes = 5, chromosome.size = 3,
       results.dir = 'tmp_3',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)
 combined.res3 <- combine.islands('tmp_3', snp.annotations[ 1:10, ],
                                  pp.list, 2)
 unlink('tmp_reg', recursive = TRUE)

# create three permuted datasets
set.seed(1400)
perm.data.list <- permute.dataset(pp.list, "perm_data",
                                  n.permutations = 3)

#pre-process permuted data
case.p1 <- readRDS("perm_data/case.permute1.rds")
comp.p1 <- readRDS("perm_data/complement.permute1.rds")
p1.list <- preprocess.genetic.data(case.p1,
                                   complement.genetic.data = comp.p1,
                                    ld.block.vec = c(10))

case.p2 <- readRDS("perm_data/case.permute2.rds")
comp.p2 <- readRDS("perm_data/complement.permute2.rds")
```

```
p2.list <- preprocess.genetic.data(case.p2,
                                   complement.genetic.data = comp.p2,
                                    ld.block.vec = c(10))

case.p3 <- readRDS("perm_data/case.permute3.rds")
comp.p3 <- readRDS("perm_data/complement.permute3.rds")
p3.list <- preprocess.genetic.data(case.p3,
                                   complement.genetic.data = comp.p3,
                                   ld.block.vec = c(10))

#permutation 1, chromosome size 2
run.gadgets(p1.list, n.chromosomes = 5, chromosome.size = 2,
       results.dir = 'p1_tmp_2',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)
 p1.combined.res2 <- combine.islands('p1_tmp_2', snp.annotations[ 1:10, ],
                                     p1.list, 2)
 unlink('tmp_reg', recursive = TRUE)

#permutation 1, chromosome size 3
run.gadgets(p1.list, n.chromosomes = 5, chromosome.size = 3,
       results.dir = 'p1_tmp_3',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)
 p1.combined.res3 <- combine.islands('p1_tmp_3', snp.annotations[ 1:10, ],
                                     p1.list, 2)
 unlink('tmp_reg', recursive = TRUE)

#permutation 2, chromosome size 2
run.gadgets(p2.list, n.chromosomes = 5, chromosome.size = 2,
       results.dir = 'p2_tmp_2',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)
 p2.combined.res2 <- combine.islands('p2_tmp_2', snp.annotations[ 1:10, ],
                                     p2.list, 2)
 unlink('tmp_reg', recursive = TRUE)

#permutation 2, chromosome size 3
run.gadgets(p2.list, n.chromosomes = 5, chromosome.size = 3,
       results.dir = 'p2_tmp_3',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)
 p2.combined.res3 <- combine.islands('p2_tmp_3', snp.annotations[ 1:10, ],
                                     p2.list, 2)
 unlink('tmp_reg', recursive = TRUE)
```

```
#permutation 3, chromosome size 2
run.gadgets(p3.list, n.chromosomes = 5, chromosome.size = 2,
       results.dir = 'p3_tmp_2',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)
 p3.combined.res2 <- combine.islands('p3_tmp_2', snp.annotations[ 1:10, ],
                                     p3.list, 2)
 unlink('tmp_reg', recursive = TRUE)

#permutation 3, chromosome size 3
run.gadgets(p3.list, n.chromosomes = 5, chromosome.size = 3,
       results.dir = 'p3_tmp_3',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)
 p3.combined.res3 <- combine.islands('p3_tmp_3', snp.annotations[ 1:10, ],
                                     p3.list, 2)
 unlink('tmp_reg', recursive = TRUE)

## create list of results

# chromosome size 2 results
chrom2.list <- list(
    observed.data = combined.res2$fitness.score,
    permutation.list = list(
        p1.combined.res2$fitness.score,
        p2.combined.res2$fitness.score,
        p3.combined.res2$fitness.score
    )
)

# chromosome size 3 results
chrom3.list <- list(
    observed.data = combined.res3$fitness.score,
    permutation.list = list(
        p1.combined.res3$fitness.score,
        p2.combined.res3$fitness.score,
        p3.combined.res3$fitness.score
    )
)

final.results <- list(chrom2.list, chrom3.list)

 lapply(c('tmp_2', 'tmp_3', 'p1_tmp_2', 'p2_tmp_2', 'p3_tmp_2',
          'p1_tmp_3', 'p2_tmp_3', 'p3_tmp_3', 'perm_data'), unlink,
          recursive = TRUE)
```

`GxE.fitness.score` *A function to assign an E-GADGETS (GxGxE) fitness score to a chromosome*

Description

This function assigns the (currently experimental) E-GADGETS fitness score to a chromosome.

Usage

```
GxE.fitness.score(
  case.genetic.data,
  mother.genetic.data,
  father.genetic.data,
  exposure.mat,
  target.snps,
  weight.lookup,
  null.mean.vec = c(0, 0),
  null.sd.vec = c(1, 1),
  n.different.snps.weight = 2,
  n.both.one.weight = 1
)
```

Arguments

`case.genetic.data`
: The genetic data of the disease affected children from case-parent trios or disease-discordant sibling pairs. If searching for maternal SNPs that are related to risk of disease in the child, some of the columns in `case.genetic.data` may contain maternal SNP genotypes (See argument `mother.snps` for how to indicate which SNPs columns correspond to maternal genotypes). Columns are SNP allele counts, and rows are individuals. This object may either be of class matrix' OR of class 'big.matrix'. If of class 'big.matrix' it must be file backed as type 'integer' (see the `bigmemory` package for more information). The ordering of the columns must be consistent with the LD structure specified in `ld.block.vec`. The genotypes cannot be dosages imputed with uncertainty.

`mother.genetic.data`
: The genetic data for the mothers of the cases in `case.genetic.data`. This should only be specified when searching for epistasis or GxGxE effects based only on case-parent triads, and not when searching for maternal SNPs that are related to the child's risk of disease. Columns are SNP allele counts, rows are individuals. This object may either be of class 'matrix' OR of class 'big.matrix'. If of class big.matrix' it must be file backed as type 'integer' (see the bigmemory package for more information). The genotypes cannot be dosages imputed with uncertainty.

`father.genetic.data`
: The genetic data for the fathers of the cases in `case.genetic.data`. This should only be specified when searching for epistasis or GxGxE effects based only on case-parent triads, and not when searching for maternal SNPs that are related to the child's risk of disease. Columns are SNP allele counts, rows are individuals. This object may either be of class 'matrix' OR of class 'big.matrix'. If of class big.matrix' it must be file backed as type 'integer' (see the bigmemory package for more information). The genotypes cannot be dosages imputed with uncertainty.

`exposure.mat`
: A matrix of the input categorical and continuous exposures to be used in the experimental E-GADGETS fitness score. If there are categorical exposure variables with more than 2 levels, those should be dummy coded.

`target.snps`
: An integer vector of the columns corresponding to the collection of SNPs, or chromosome, for which the fitness score will be computed.

`weight.lookup`
: A vector that maps a family weight to the weighted sum of the number of different SNPs and SNPs both equal to one.

`null.mean.vec`
: A vector of estimated null means for each of the components of the E-GADGETS fitness score. It should be set to the values of the "null.mean" element of the file "null.mean.sd.info.rds" for the observed data, that is saved by the `run.gadgets` function.

`null.sd.vec`
: A vector of estimated null means for each of the components of the E-GADGETS fitness score. It should be set to the values of the "null.se" element of the file "null.mean.sd.info.rds" for the observed data, that is saved by the `run.gadgets` function.

`n.different.snps.weight`
: The number by which the number of different SNPs between a case and complement/unaffected sibling is multiplied in computing the family weights. Defaults to 2.

`n.both.one.weight`
: The number by which the number of SNPs equal to 1 in both the case and complement/unaffected sibling is multiplied in computing the family weights. Defaults to 1.

Value

A list:

fitness.score The chromosome fitness score.

sum.dif.vecs The element of the Hotelling-Lawley trace matrix corresponding to each SNP. Larger magnitudes indicate larger contributions to the score, but are otherwise difficult to interpret.

ht_trace The Hotelling-Lawley trace statistic from the transmission-based fitness score component.

wald_stat The Wald statistic from the family-based component of the fitness score.

Examples

```
data(case.gxe)
data(dad.gxe)
data(mom.gxe)
data(exposure)
case.gxe <- case.gxe + 0.0
mom.gxe <- mom.gxe + 0.0
dad.gxe <- dad.gxe + 0.0
exposure <- as.matrix(exposure + 0.0)
weight.lookup <- vapply(seq_len(6), function(x) 2^x, 1.0)
res <- GxE.fitness.score(case.gxe, mom.gxe, dad.gxe, exposure, c(1, 4, 7),
                         weight.lookup)
```

GxE.test *A function to run a test of interaction between a collection of SNPs and exposures (experimental).*

Description

This function runs a permutation based test run a test of interaction between a collection of SNPs and exposure variables.

Usage

```
GxE.test(
  snp.cols,
  preprocessed.list,
  null.mean.vec = c(0, 0),
  null.sd.vec = c(1, 1),
  n.permutes = 10000,
  n.different.snps.weight = 2,
  n.both.one.weight = 1,
  weight.function.int = 2
)
```

Arguments

`snp.cols`	An integer vector specifying the columns in the input data containing the SNPs to be tested.
`preprocessed.list`	The initial list produced by function `preprocess.genetic.data`.
`null.mean.vec`	A vector of estimated null means for each of the components of the E-GADGETS fitness score. It should be set to the values of the "null.mean" element of the file "null.mean.sd.info.rds" for the observed data, that is saved by the `run.gadgets` function.

`null.sd.vec` A vector of estimated null means for each of the components of the E-GADGETS fitness score. It should be set to the values of the "null.se" element of the file "null.mean.sd.info.rds" for the observed data, that is saved by the `run.gadgets` function.

`n.permutes` The number of permutations on which to base the test. Defaults to 10000.

`n.different.snps.weight`
The number by which the number of different SNPs between a case and complement/unaffected sibling is multiplied in computing the family weights. Defaults to 2.

`n.both.one.weight`
The number by which the number of SNPs equal to 1 in both the case and complement/unaffected sibling is multiplied in computing the family weights. Defaults to 1.

`weight.function.int`
An integer used to assign family weights. Specifically, we use `weight.function.int` in a function that takes the weighted sum of the number of different SNPs and SNPs both equal to one as an argument, denoted as x, and returns a family weight equal to `weight.function.int`^x. Defaults to 2.

Value

A list of three elements:

pval The p-value of the test.

obs.fitness.score The fitness score from the observed data

perm.fitness.scores A vector of fitness scores for the permuted datasets.

Examples

```
data(case.gxe)
data(dad.gxe)
data(mom.gxe)
data(exposure)
data(snp.annotations.mci)
pp.list <- preprocess.genetic.data(case.gxe, father.genetic.data = dad.gxe,
                                mother.genetic.data = mom.gxe,
                                ld.block.vec = rep(6, 4),
                                categorical.exposures = exposure)
 run.gadgets(pp.list, n.chromosomes = 5, chromosome.size = 3,
             results.dir = "tmp_gxe", cluster.type = "interactive",
             registryargs = list(file.dir = "tmp_reg_gxe", seed = 1300),
             n.islands = 8, island.cluster.size = 4,
             n.migrations = 1)


combined.res <- combine.islands('tmp_gxe', snp.annotations.mci, pp.list, 1)
top.snps <- as.vector(t(combined.res[1, 1:3]))
set.seed(10)
GxE.test.res <- GxE.test(top.snps, pp.list)
```

```
unlink('tmp_gxe', recursive = TRUE)
unlink('tmp_reg_gxe', recursive = TRUE)
```

`mom` *Genotypes for the mothers of case-parent triads.*

Description

A simulated dataset containing the counts of the alternate allele for 100 SNPs for the mothers in 1000 simulated case-parent triads. Columns represent SNPs, rows are individuals. SNPs in columns 51, 52, 76, and 77 represent a true risk pathway.

Usage

```
data(mom)
```

Format

A data frame with 1000 rows and 100 variables

`mom.gxe` *Genotypes for the mothers of case-parent triads with a simulated gene environment interaction.*

Description

A simulated dataset containing the counts of the alternate allele for 24 SNPs for the mothers in 1000 simulated case-parent triads. Columns represent SNPs, rows are individuals. SNPs in columns 6, 12, and 18 represent a simulated risk pathway, where, in the child, at least one copy of the alternate allele for each path SNP in addition to exposure 1 confers increased disease risk. .

Usage

```
data(mom.gxe)
```

Format

A data frame with 1000 rows and 24 variables

mom.mci — *Genotypes for the mothers of case-parent triads with a simulated maternal-fetal interaction.*

Description

A simulated dataset containing the counts of the alternate allele for 24 SNPs for the mothers in 1000 simulated case-parent triads. Columns represent SNPs, rows are individuals. The SNP in column 6 interacts with the SNPs in columns 12 and 18 of dataset `case.mci` to increase risk of disease in the child, where at least one copy of the alternate allele (genotype 1 or 2) is required at each implicated locus. .

Usage

```
data(mom.mci)
```

Format

A matrix with 1000 rows and 24 variables

network.plot — *A function to plot a network of SNPs with potential multi-SNP effects.*

Description

This function plots a network of SNPs with potential multi-SNP effects.

Usage

```
network.plot(
  graphical.score.list,
  preprocessed.list,
  n.top.scoring.pairs = NULL,
  node.shape = "circle",
  repulse.rad = 1000,
  node.size = 25,
  graph.area = 100,
  vertex.label.cex = 0.5,
  edge.width.cex = 12,
  plot = TRUE,
  edge.color.ramp = c("lightblue", "blue"),
  node.color.ramp = c("white", "red"),
  plot.legend = TRUE,
  high.ld.threshold = 0.1,
  plot.margins = c(2, 1, 2, 1),
```

```
  legend.title.cex = 1.75,
  legend.axis.cex = 1.75,
  ...
)
```

Arguments

graphical.score.list	The list returned by function compute.graphical.scores, or a subset of it, if there are too many returned SNP-pairs to plot without the figure becoming too crowded. By default, the SNPs will be labeled with their RSIDs, listed in columns 3 and 4. Users can create custom labels by changing the values in these two columns.
preprocessed.list	The initial list produced by function preprocess.genetic.data.
n.top.scoring.pairs	An integer indicating the number of top scoring SNP-pairs to plot. Defaults to, NULL, which plots all pairs. For large networks, plotting a subset of the top scoring pairs can improve the appearance of the graph.
node.shape	The desired node shape. See names(igraph:::.igraph.shapes) for available shapes. Defaults to circle. If both maternal and child SNPs are to be plotted, this argument should be a vector of length 2, whose first element is the desired child SNP shape, and second SNP is the desired mother SNP shape.
repulse.rad	A scalar affecting the graph shape. Decrease to reduce overlapping nodes, increase to move nodes closer together.
node.size	A scalar affecting the size of the graph nodes. Increase to increase size.
graph.area	A scalar affecting the size of the graph area. Increase to increase graph area.
vertex.label.cex	A scalar controlling the size of the vertex label. Increase to increase size.
edge.width.cex	A scalar controlling the width of the graph edges. Increase to make edges wider.
plot	A logical indicating whether the network should be plotted. If set to false, this function will return an igraph object to be used for manual plotting.
edge.color.ramp	A character vector of colors. The coloring of the network edges will be shown on a gradient, with the lower scoring edge weights closer to the first color specified in edge.color.ramp, and higher scoring weights closer to the last color specified. By default, the low scoring edges are light blue, and high scoring edges are dark blue.
node.color.ramp	A character vector of colors. The coloring of the network nodes will be shown on a gradient, with the lower scoring nodes closer to the first color specified in node.color.ramp, and higher scoring nodes closer to the last color specified. By default, the low scoring nodes are whiter, and high scoring edges are redder.
plot.legend	A boolean indicating whether a legend should be plotted. Defaults to TRUE.

high.ld.threshold
: A numeric value between 0 and 1, indicating the r^2 threshold in complements (or unaffected siblings) above which a pair of SNPs in the same LD block (as specified in preprocessed.list) should be considered in high LD. Connections between these high LD SNPs will be dashed instead of solid lines. Defaults to 0.1. If both maternal and child SNPs are among the input variants in preprocessed.list, dashed lines can only appear between SNPs of the same type, i.e., between two maternal SNPs, or between two child SNPs.

plot.margins
: A vector of length 4 passed to par(mar =). Defaults to c(2, 1, 2, 1).

legend.title.cex
: A numeric value controlling the size of the legend titles. Defaults to 1.75. Increase to increase font size, decrease to decrease font size.

legend.axis.cex
: A numeric value controlling the size of the legend axis labels. Defaults to 1.75. Increase to increase font size, decrease to decrease font size.

...
: Additional arguments to be passed to plot.igraph.

Value

An igraph object, if plot is set to FALSE.

Examples

```
data(case)
data(dad)
data(mom)
case <- as.matrix(case)
dad <- as.matrix(dad)
mom <- as.matrix(mom)
data(snp.annotations)
set.seed(1400)

# preprocess data
target.snps <- c(1:3, 30:32, 60:62, 85)
pp.list <- preprocess.genetic.data(case[, target.snps],
                                   father.genetic.data = dad[ , target.snps],
                                   mother.genetic.data = mom[ , target.snps],
                                   ld.block.vec = c(3, 3, 3, 1))
## run GA for observed data

#observed data chromosome size 2
run.gadgets(pp.list, n.chromosomes = 5, chromosome.size = 2,
       results.dir = 'tmp_2',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)
 combined.res2 <- combine.islands('tmp_2', snp.annotations[ target.snps, ],
                                   pp.list, 2)
 unlink('tmp_reg', recursive = TRUE)
```

```
 #observed data chromosome size 3
 run.gadgets(pp.list, n.chromosomes = 5, chromosome.size = 3,
       results.dir = 'tmp_3',
       cluster.type = 'interactive',
       registryargs = list(file.dir = 'tmp_reg', seed = 1500),
       generations = 2, n.islands = 2, island.cluster.size = 1,
       n.migrations = 0)
 combined.res3 <- combine.islands('tmp_3', snp.annotations[ target.snps, ],
                                  pp.list, 2)
 unlink('tmp_reg', recursive = TRUE)

## create list of results
final.results <- list(combined.res2[1:3, ], combined.res3[1:3, ])

 ## compute edge scores
 set.seed(20)
 graphical.list <- compute.graphical.scores(final.results, pp.list,
                                            pval.thresh = 0.5)

## plot
set.seed(10)
network.plot(graphical.list, pp.list)

lapply(c("tmp_2", "tmp_3"), unlink, recursive = TRUE)
```

`permute.dataset` *A function to create permuted datasets for permutation based hypothesis testing.*

Description

This function creates permuted datasets for permutation based hypothesis testing of GADGETS fitness scores.

Usage

```
permute.dataset(
  preprocessed.list,
  permutation.data.file.path,
  n.permutations = 100,
  bp.param = bpparam()
)
```

Arguments

`preprocessed.list`
: The output list from `preprocess.genetic.data` for the original genetic data.

`permutation.data.file.path`
: If running GADGETS for GxG interactions, this argument specifies a directory where each permuted dataset will be saved on disk. If searching for GxE interactions, permuted versions of the exposure matrix will be saved to this directory.

`n.permutations`
: The number of permuted datasets to create.

`bp.param`
: The BPPARAM argument to be passed to bplapply. See `BiocParallel::bplapply` for more details.

Value

If genetic data are specified, a total of `n.permutations` datasets containing pairs of case and complement data, where the observed case/complement status has been randomly flipped or not flipped, will be saved to `permutation.data.file.path`. If exposure data are specified, a total of `n.permutations` exposure matrices, where the observed exposures have been randomly re-assigned across the permuted 'families'.

Examples

```
data(case)
case <- as.matrix(case)
data(dad)
dad <- as.matrix(dad)
data(mom)
mom <- as.matrix(mom)
pp.list <- preprocess.genetic.data(case[, 1:10],
                                   father.genetic.data = dad[ , 1:10],
                                   mother.genetic.data = mom[ , 1:10],
                                   ld.block.vec = c(10))
set.seed(15)
perm.data.list <- permute.dataset(pp.list, "tmp_perm", n.permutations = 1)
unlink("tmp_perm", recursive = TRUE)
```

`preprocess.genetic.data` *A function to pre-process case-parent triad or disease-discordant sibling data.*

Description

This function performs several pre-processing steps, intended for use before function run.gadgets.

Usage

```
preprocess.genetic.data(
  case.genetic.data,
  complement.genetic.data = NULL,
  father.genetic.data = NULL,
```

```
  mother.genetic.data = NULL,
  ld.block.vec = NULL,
  bp.param = bpparam(),
  snp.sampling.probs = NULL,
  categorical.exposures = NULL,
  continuous.exposures = NULL,
  mother.snps = NULL,
  child.snps = NULL,
  lower.order.gxe = FALSE
)
```

Arguments

`case.genetic.data`
: The genetic data of the disease affected children from case-parent trios or disease-discordant sibling pairs. If searching for maternal SNPs that are related to risk of disease in the child, some of the columns in `case.genetic.data` may contain maternal SNP genotypes (See argument `mother.snps` for how to indicate which SNPs columns correspond to maternal genotypes). Columns are SNP allele counts, and rows are individuals. This object may either be of class matrix' OR of class 'big.matrix'. If of class 'big.matrix' it must be file backed as type 'integer' (see the `bigmemory` package for more information). The ordering of the columns must be consistent with the LD structure specified in `ld.block.vec`. The genotypes cannot be dosages imputed with uncertainty.

`complement.genetic.data`
: A genetic dataset for the controls corresponding to the genotypes in `case.genetic.data`.For SNPs that correspond to the affected child in `case.genetic.data`, the corresponding column in `complement.genetic.data` should be set equal to mother allele count + father allele count - case allele count. If using disease-discordant siblings this argument should be the genotypes for the unaffected siblings. For SNPs in `case.genetic.data` that represent maternal genotypes (if any) the corresponding column in `complement.genetic.data` should be the paternal genotypes for that SNP. Regardless, `complement.genetic.data` may be an object of either class matrix' OR of class 'big.matrix'. If of class 'big.matrix' it must be file backed as type 'integer' (see the bigmemory package for more information). Columns are SNP allele counts, rows are families. If not specified, `father.genetic.data` and `mother.genetic.data` must be specified. The genotypes cannot be dosages imputed with uncertainty.

`father.genetic.data`
: The genetic data for the fathers of the cases in `case.genetic.data`. This should only be specified when searching for epistasis or GxGxE effects based only on case-parent triads, and not when searching for maternal SNPs that are related to the child's risk of disease. Columns are SNP allele counts, rows are individuals. This object may either be of class 'matrix' OR of class 'big.matrix'. If of class big.matrix' it must be file backed as type 'integer' (see the bigmemory package for more information). The genotypes cannot be dosages imputed with uncertainty.

`mother.genetic.data`
: The genetic data for the mothers of the cases in `case.genetic.data`. This

should only be specified when searching for epistasis or GxGxE effects based only on case-parent triads, and not when searching for maternal SNPs that are related to the child's risk of disease. Columns are SNP allele counts, rows are individuals. This object may either be of class 'matrix' OR of class 'big.matrix'. If of class big.matrix' it must be file backed as type 'integer' (see the bigmemory package for more information). The genotypes cannot be dosages imputed with uncertainty.

`ld.block.vec` An integer vector specifying the linkage blocks of the input SNPs. As an example, for 100 candidate SNPs, suppose we specify `ld.block.vec <- c(25, 50, 25)`. This vector indicates that the input genetic data has 3 distinct linkage blocks, with SNPs 1-25 in the first linkage block, 26-75 in the second block, and 76-100 in the third block. Note that this means the ordering of the columns (SNPs) in `case.genetic.data` must be consistent with the LD blocks specified in `ld.block.vec`. In the absence of outside information, a reasonable default is to consider SNPs to be in LD if they are located on the same biological chromosome. If `case.genetic.data` includes both maternal and child SNP genotypes, we recommend considering any maternal SNP and any child SNP located on the same nominal biological chromosome as 'in linkage'. E.g., we recommend considering any maternal SNPs located on chromosome 1 as being 'linked' to any child SNPs located on chromosome 1, even though, strictly speaking, the maternal and child SNPs are located on separate pieces of DNA. If not specified, `ld.block.vec` defaults to assuming all input SNPs are in linkage, which may be overly conservative and could adversely affect performance.

`bp.param` The BPPARAM argument to be passed to bplapply when estimating marginal disease associations for each SNP. If using a cluster computer, this parameter needs to be set with care. See `BiocParallel::bplapply` for more details.

`snp.sampling.probs`
A vector indicating the sampling probabilities of the SNPs in `case.genetic.data`. SNPs will be sampled in the genetic algorithm proportional to the values specified. If not specified, by default, chi-square statistics of association will be computed for each SNP, and sampling will be proportional to the square root of those statistics. If user specified, the values of `snp.sampling.probs` need not sum to 1, they just need to be positive real numbers. See argument `prob` from function `sample` for more details.

`categorical.exposures`
(experimental) A matrix or data.frame of integers corresponding to categorical exposures corresponding to the cases in `case.genetic.data`. Defaults to NULL, which will result in GADGETS looking for epistatic interactions, rather than SNP by exposure interactions. `categorical.exposures` should not be missing any data; families with missing exposure data should be removed from the analysis prior to input.

`continuous.exposures`
(experimental) A matrix or data.frame of numeric values representing continuous exposures corresponding to the families in `case.genetic.data`. Defaults to NULL, which will result in GADGETS searching for epistatic interactions, rather than SNP by exposure interactions. `continuous.exposures` should not be missing any data; families with missing exposure data should be removed from the analysis prior to input.

`mother.snps` If searching for maternal SNPs that are associated with disease in the child, the indices of the maternal SNP columns in object `case.genetic.data`. Otherwise does not need to be specified.

`child.snps` If searching for maternal SNPs that are associated with disease in the child, the indices of the child SNP columns in object `case.genetic.data`. Otherwise does not need to be specified.

`lower.order.gxe`
(experimental) A boolean indicating whether, if multiple exposures of interest are input, E-GADGETS should search for only for genetic interactions with the joint combination of exposures (i.e., GxGxExE interactions), or if it should additionally search for lower-order interactions that involve subsets of the exposures that were input (i.e., GxGxE in addition to GxGxExE). The default, FALSE, restricts the search to GxGxExE interactions. Users should be cautious about including large numbers of input exposures, and, if they do, very cautious about setting this argument to TRUE.

Value

A list containing the following:

case.genetic.data A matrix of case/maternal genotypes.

complement.genetic.data A matrix of complement/sibling/paternal genotypes. If running E-GADGETS, this is set to a 1x1 matrix whose single entry is 0, and not used

mother.genetic.data If running E-GADGETS, A matrix of maternal genotypes, otherwise a 1x1 matrix whose single entry is 0.0, and not used

father.genetic.data If running E-GADGETS, A matrix of mpaternal genotypes, otherwise a 1x1 matrix whose single entry is 0.0, and not used

chisq.stats A vector of chi-square statistics corresponding to marginal SNP-disease associations, if `snp.sampling.probs` is not specified, and `snp.sampling.probs` otherwise.

ld.block.vec A vector eaul to `cumsum(ld.block.vec)`.

exposure.mat A design matrix of the input categorical and continuous exposures, if specified. Otherwise NULL.

E_GADGETS A boolean indicating whether a GxGxE search is desired.

mother.snps A vector of the column indices of maternal SNPs in `case.genetic.data`, set to NULL if not applicable.

child.snps A vector of the column indices of child SNPs in `case.genetic.data`, set to NULL if not applicable.

continuous.exposure A boolean indicating whether, for E-GADGETS, the exposure is continuous. Set to FALSE and not used in GADGETS.

exposure.min.max A vector,to be used in E-GADGETS when the exposure is continuous, whose first element is the minimum observed exposure value and second element is the maximum observed exposure value. Otherwise set to a vector of zeroes and not used.

Examples

```
data(case)
data(dad)
data(mom)
case <- as.matrix(case)
dad <- as.matrix(dad)
mom <- as.matrix(mom)
res <- preprocess.genetic.data(case[, 1:10],
                               father.genetic.data = dad[ , 1:10],
                               mother.genetic.data = mom[ , 1:10],
                               ld.block.vec = c(10))
```

`run.gadgets` *A function to run the GADGETS algorithm to detect multi-SNP effects in case-parent triad studies.*

Description

This function runs the GADGETS algorithm to detect multi-SNP effects in case-parent triad studies.

Usage

```
run.gadgets(
  data.list,
  n.chromosomes,
  chromosome.size,
  results.dir,
  cluster.type,
  registryargs = list(file.dir = NA, seed = 1500),
  resources = list(),
  cluster.template = NULL,
  n.workers = min(detectCores() - 2, n.islands/island.cluster.size),
  n.chunks = NULL,
  n.different.snps.weight = 2,
  n.both.one.weight = 1,
  weight.function.int = 2,
  generations = 500,
  gen.same.fitness = 50,
  initial.sample.duplicates = FALSE,
  snp.sampling.type = "chisq",
  crossover.prop = 0.8,
  n.islands = 1000,
  island.cluster.size = 4,
  migration.generations = 50,
  n.migrations = 20,
  recessive.ref.prop = 0.75,
```

```
  recode.test.stat = 1.64,
  n.random.chroms = 10000,
  null.mean.vec = NULL,
  null.sd.vec = NULL
)
```

Arguments

data.list	The output list from preprocess.genetic.data.
n.chromosomes	An integer specifying the number of chromosomes to use for each island in GADGETS.
chromosome.size	An integer specifying the number of SNPs in each chromosome.
results.dir	The directory to which island results will be saved.
cluster.type	A character string indicating the type of cluster on which to evolve solutions in parallel. Supported options are interactive, socket, multicore, sge, slurm, lsf, openlava, or torque. See the \ documentation for package batchtools for more information.
registryargs	A list of the arguments to be provided to batchtools::makeRegistry.
resources	A named list of key-value pairs to be substituted into the template file. Options available are specified in batchtools::submitJobs.
cluster.template	A character string of the path to the template file required for the cluster specified in cluster.type. Defaults to NULL. Required for options sge, slurm, lsf, openlava and torque of argument cluster.type.
n.workers	An integer indicating the number of workers for the cluster specified in cluster.type, if socket or multicore. Defaults to parallel::detectCores - 2.
n.chunks	An integer specifying the number of chunks jobs running island clusters should be split into when dispatching jobs using batchtools. For multicore or socket cluster.type, this defaults to n.workers, resulting in the total number of island cluster jobs (equal to n.islands\island.cluster.size) being split into n.chunks chunks. All chunks then run in parallel, with jobs within a chunk running sequentially. For other cluster types, this defaults to 1 chunk, with the recommendation that users of HPC clusters which support array jobs specify chunks.as.arrayjobs = TRUE in argument resources. For those users, the setup will submit an array of n.islands\island.cluster.size jobs to the cluster. For HPC clusters that do not support array jobs, the default setting should not be used. See batchtools::submitJobs for more information on job chunking.
n.different.snps.weight	The number by which the number of different SNPs between a case and complement or unaffected sibling is multiplied in computing the family weights. Defaults to 2.
n.both.one.weight	The number by which the number of SNPs equal to 1 in both the case and complement or unaffected sibling is multiplied in computing the family weights. Defaults to 1.

`weight.function.int`
: An integer used to assign family weights. Specifically, we use `weight.function.int` in a function that takes the weighted sum of the number of different SNPs and SNPs both equal to one as an argument, denoted as x, and returns a family weight equal to `weight.function.int`^x. Defaults to 2. If set to null, then the family weight will not be exponentiated and instead set to just x.

`generations`
: The maximum number of generations for which GADGETS will run. Defaults to 500.

`gen.same.fitness`
: The number of consecutive generations with the same fitness score required for algorithm termination. Defaults to 50.

`initial.sample.duplicates`
: A logical indicating whether the same SNP can appear in more than one chromosome in the initial sample of chromosomes (the same SNP may appear in more than one chromosome thereafter, regardless). Default to FALSE.

`snp.sampling.type`
: A string indicating how SNPs are to be sampled for mutations. Options are 'chisq', 'random', or 'manual'. The 'chisq' option takes into account the marginal association between a SNP and disease status, with larger marginal associations corresponding to higher sampling probabilities. The 'random' option gives each SNP the same sampling probability regardless of marginal association. The 'manual' option should be used when `snp.sampling.probs` are manually input into function `preprocess.genetic.data`. Defaults to 'chisq'.

`crossover.prop`
: A numeric between 0 and 1 indicating the proportion of chromosomes to be subjected to cross over.The remaining proportion will be mutated. Defaults to 0.8.

`n.islands`
: An integer indicating the number of islands to be used. Defaults to 1000.

`island.cluster.size`
: An integer specifying the number of islands in a given cluster. Must evenly divide `n.islands` and defaults to 4. More specifically, under the default settings, the 1000 `n.islands` are split into 250 distinct clusters each containing 4 islands (`island.cluster.size`). Within a cluster, migrations of top chromosomes from one cluster island to another are periodically permitted (controlled by `migration.generations`), and distinct clusters evolve completely independently.

`migration.generations`
: An integer equal to the number of generations between migrations among islands of a distinct cluster. Argument `generations` must be an integer multiple of this value. Defaults to 50.

`n.migrations`
: The number of chromosomes that migrate among islands. This value must be less than `n.chromosomes` and greater than 0, defaulting to 20.

`recessive.ref.prop`
: The proportion to which the observed proportion of informative cases with the provisional risk genotype(s) will be compared to determine whether to recode the SNP as recessive. Defaults to 0.75.

`recode.test.stat`
: For a given SNP, the minimum test statistic required to recode and recompute the fitness score using recessive coding. Defaults to 1.64. See the GADGETS paper for specific details.

`n.random.chroms`
: (experimental) The number of random chromosomes used to construct a reference null mean and standard deviations vectors to compute the E-GADGETS (GxGxE) fitness score.

`null.mean.vec`
: (experimental) A vector of estimated null means for each of the components of the E-GADGETS fitness score. This needs to be specified if running permutes under the no-GxE null, and should be set to the values in the "null.mean" element of the "null.mean.sd.info.rds" file stored in the `results.dir` directory for the observed data. It also should be specified if analyst wants to replicate the results of a previous E-GADGETS run, or if some of the islands of a run failed to complete, and the analyst forgot to set the seed prior to running `run.gadgets`.

`null.sd.vec`
: A vector of estimated null standard deviations for the components of the E-GADGETS fitness score. See argument `null.mean.vec` for reasons this argument might be specified. For a given run, the previously used vector can also be found in the "null.se" element of the file "null.mean.sd.info.rds" stored in the `results.dir` directory.

Value

For each island, a list of two elements will be written to `results.dir`:

top.chromosome.results A data.table of the final generation chromosomes, their fitness scores, and, for GADGETS, additional information pertaining to nominated risk-related genotypes. See the package vignette for an example and the documentation for `chrom.fitness.score` for additional details.

n.generations The total number of generations run.

Examples

```
data(case)
case <- as.matrix(case)
data(dad)
dad <- as.matrix(dad)
data(mom)
mom <- as.matrix(mom)
pp.list <- preprocess.genetic.data(case[, 1:10],
                                   father.genetic.data = dad[ , 1:10],
                                   mother.genetic.data = mom[ , 1:10],
                                   ld.block.vec = c(10))
run.gadgets(pp.list, n.chromosomes = 4, chromosome.size = 3,
            results.dir = 'tmp', cluster.type = 'interactive',
            registryargs = list(file.dir = 'tmp_reg', seed = 1500),
            generations = 2, n.islands = 2, island.cluster.size = 1,
            n.migrations = 0)

unlink('tmp_bm', recursive = TRUE)
```

```
unlink('tmp', recursive = TRUE)
unlink('tmp_reg', recursive = TRUE)
```

`snp.annotations` *RSID, REF, and ALT annotations for example dataset SNPs*

Description

A data.frame containing the RSID, REF allele and ALT allele for each SNP in the example datasets `case`, `mom`, and `dad`. The SNPs are in the same order as they appear in the example dataset columns.

Usage

```
data(snp.annotations)
```

Format

A data frame with 100 rows and 3 variables

`snp.annotations.mci` *RSID, REF, and ALT annotations for example dataset SNPs*

Description

A data.frame containing the RSID, REF allele and ALT allele for each SNP in the example datasets `case.mci`, `mom.mci`, `dad.mci`, `case.gxe`, `mom.gxe`, and `dad.gxe`. The SNPs are in the same order as they appear in the example dataset columns.

Usage

```
data(snp.annotations.mci)
```

Format

A matrix with 24 rows and 3 variables

Index

∗ **datasets**
  case, 2
  case.gxe, 3
  case.mci, 3
  dad, 11
  dad.gxe, 11
  dad.mci, 12
  exposure, 14
  mom, 28
  mom.gxe, 28
  mom.mci, 29
  snp.annotations, 41
  snp.annotations.mci, 41

case, 2
case.gxe, 3
case.mci, 3
chrom.fitness.score, 4
combine.islands, 6
compute.graphical.scores, 8

dad, 11
dad.gxe, 11
dad.mci, 12

epistasis.test, 12
epistasisGA, 14
exposure, 14

GADGETS, 15
global.test, 19
GxE.fitness.score, 24
GxE.test, 26

mom, 28
mom.gxe, 28
mom.mci, 29

network.plot, 29

permute.dataset, 32
preprocess.genetic.data, 33

run.gadgets, 37

snp.annotations, 41
snp.annotations.mci, 41