Package 'Ravages'

October 1, 2020

Type Package

Title Rare Variant Analysis and Genetic Simulations

Version 0.1

Date 2020-06-08

Encoding UTF-8

Author Hervé Perdry and Ozvan Bocher

Maintainer Ozvan Bocher <bocherozvan@gmail.com>

Description Rare variant association tests (burden tests and SKAT) and genetic simulations.

License GPL-3

LinkingTo Rcpp, RcppParallel, RcppEigen, gaston, BH

Depends R (>= 3.5.0), Rcpp, RcppParallel, methods, gaston, mlogit (>= 1.1-0)

Imports Formula, dfidx

NeedsCompilation yes

Suggests knitr, rmarkdown

VignetteBuilder knitr

LazyLoad yes

LazyData yes

R topics documented:

`bed.matrix.split.genomic.region`
Bed matrix for variants associated to multiple genomic regions

---

Description

Creates a new bed matrix with variants associated to multiple genomic regions being duplicated

Usage

```
bed.matrix.split.genomic.region(x, changeID=TRUE, genomic.region=NULL,
                                split.pattern=",")
```

Arguments

`x`	A bed.matrix
`changeID`	TRUE/FALSE: whether to change the variants ID by including the gene name
`genomic.region`	A vector containing the genomic region of each variant
`split.pattern`	The character separating the genomic regions

Details

If changeID=TRUE, variants will have new IDs being CHR:POS:A1:A2:genomic.region. The genomic region(s) associated to each varaint should be in x@snps$genomic.region or given as a vector to genomic.region. If both are present, genomic.region is used.

Value

A bed matrix with variants assigned to multiple genomic regions being duplicated and the corresponding genomic regions separated and transformed into factors.

Examples

```
#Example bed matrix with 4 variants
  x.ex <- as.bed.matrix(x=matrix(0, ncol=4, nrow=10),
                    bim=data.frame(chr=1:4, id=paste("rs", 1:4, sep=""), dist = rep(0,4),
                                pos=c(150,150,200,250), A1=rep("A", 4), A2=rep("T", 4)))

#Example genes dataframe
  genes.ex <- data.frame(Chr=c(1,1,3,4), Start=c(10,110,190,220), End=c(170,180,250,260),
                         Gene_Name=letters[1:4])

#Attribute genomic regions
  x.ex <- set.genomic.region(x.ex, genes = genes.ex)

#Split genomic regions
  x.ex.split <- bed.matrix.split.genomic.region(x.ex, split.pattern = ",")
```

---

burden *Linear, logistic or multinomial regression on a genetic score*

Description

Performs burden tests on categorial or continuous phenotypes

Usage

```
burden(x, NullObject, genomic.region = x@snps$genomic.region, burden,
       maf.threshold = 0.5, get.OR.value = FALSE, alpha = 0.05, cores = 10)
```

Arguments

x	A bed matrix, only needed if burden="CAST" or burden="WSS"
NullObject	A list returned from NullObject.parameters
genomic.region	A factor containg the genomic region of each SNP, x@snps$genomic.region by default, only needed if burden="CAST" or burden="WSS"
burden	"CAST" or "WSS" to directly compute the CAST or the WSS genetic score; or a matrix with one row per individual and one column per genomic.region if another genetic score is wanted.

maf.threshold	The MAF threshold to use for the definition of a rare variant in the CAST score
get.OR.value	TRUE/FALSE: whether to return the OR values associated to the regression
alpha	The alpha threshold to use for the OR confidence interval
cores	How many cores to use, set at 10 by default. Only needed if NullObject$pheno.type = "categorial"

Details

This function will return results from the regression of the phenotype on the genetic score for each genomic region.

If only two groups of individuals are present, a classical logistic regression is performed. If more than two groups of individuals are present, a non-ordinal multinomial regression is performed, comparing each group of individuals to the reference group indicated by the argument ref.level in NullObject.parameters. The choice of the reference group won't affect the p-values, but only the Odds Ratios. In both types of regression, the p-value is estimated using the Likelihood Ratio test and the function burden.mlogit.

If the phenotype is continuous, a linear regression is performed using the function burden.continuous.

The type of phenotype is determined from NullObject$pheno.type.

If another genetic score than CAST or WSS is wanted, a matrix with one row per individual and one column per genomic.region containing this score should be given to burden. In this situation, no bed matrix x is needed.

Value

A dataframe with one row per genomic region and at least two columns:

p.value	The p.value of the regression
is.err	0/1: whether there was a convergence problem with the regression

If NullObject$pheno.type = "categorial" and get.OR.value=TRUE, additional columns are present:

OR	The OR value(s) associated to the regression. If there are more than two groups, there will be one OR value per group compared to the reference group
l.lower	The lower bound of the confidence interval of each OR
l.upper	The upper bound of the confidence interval of each OR

References

Bocher O, et al. *Rare variant associationtesting for multicategory phenotype.* Genet.Epidemiol. 2019;43:646–656.

See Also

NullObject.parameters, burden.continuous, burden.mlogit, CAST, WSS, burden.weighted.matrix

Examples

```
#Importation of data in a bed matrix
x <- as.bed.matrix(x=LCT.matrix.bed, fam=LCT.matrix.fam, bim=LCT.snps)

#Add population
x@ped[,c("pop", "superpop")] <- LCT.matrix.pop1000G[,c("population", "super.population")]

#Select EUR superpopulation
x <- select.inds(x, superpop=="EUR")
x@ped$pop <- droplevels(x@ped$pop)

#Group variants within known genes
x <- set.genomic.region(x)

#Filter of rare variants: only non-monomorphic variants with
#a MAF lower than 2.5%
#keeping only genomic regions with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.025, min.nb.snps = 10)

#run null model, using the 1000Genome population as "outcome"
x1.H0 <- NullObject.parameters(pheno = x1@ped$pop, ref.level = "CEU",
                               RVAT = "burden", pheno.type = "categorial")

#run burden test WSS
burden(x1, NullObject = x1.H0, burden = "WSS", get.OR.value=TRUE, cores = 1)


#Simulation of a covariate + Sex as a covariate
sex <- x1@ped$sex
set.seed(1) ; u <- runif(nrow(x1))
covar <- cbind(sex, u)

#Null model with the covariate sex and a continuous phenotype
x1.H0.covar <- NullObject.parameters(pheno = x1@ped$pheno <- rnorm(nrow(x1)),
                                     RVAT = "burden", pheno.type = "continuous",
                                     data = covar, formula = ~ sex)
#WSS test
burden(x1, NullObject = x1.H0.covar, burden = "WSS", get.OR.value=TRUE)
```

`burden.continuous` *Linear regression on a genetic score*

Description

Performs a linear regression on a genetic score

Usage

```
burden.continuous(x, NullObject, genomic.region = x@snps$genomic.region,
                  burden, maf.threshold = 0.5)
```

Arguments

x	A bed matrix, only needed if burden="CAST" or burden="WSS"
NullObject	A list returned from NullObject.parameters
genomic.region	A factor containg the genomic region of each SNP, x@snps$genomic.region by default, only needed if burden="CAST" or burden="WSS"
burden	"CAST" or "WSS" to directly compute the CAST or the WSS genetic score; or a matrix with one row per individual and one column per genomic.region if another genetic score is wanted.
maf.threshold	The MAF threshold to use for the definition of a rare variant in the CAST score

Details

This function will return results from the regression of the continuous phenotype on the genetic score for each genomic region.

If another genetic score than CAST or WSS is wanted, a matrix with one row per individual and one column per genomic.region containing this score should be given to burden. In this situation, no bed matrix x is needed.

Value

A dataframe with one row per genomic region and at least two columns:

p.value	The p.value of the regression
is.err	0/1: whether there was a convergence problem with the regression

See Also

CAST, WSS, burden.weighted.matrix

Examples

```
#Importation of data in a bed matrix
x <- as.bed.matrix(x=LCT.matrix.bed, fam=LCT.matrix.fam, bim=LCT.snps)

#Add population
x@ped[,c("pop", "superpop")] <- LCT.matrix.pop1000G[,c("population", "super.population")]

#Select EUR superpopulation
x <- select.inds(x, superpop=="EUR")
x@ped$pop <- droplevels(x@ped$pop)

#Group variants within known genes
x <- set.genomic.region(x)

#Filter of rare variants: only non-monomorphic variants with
#a MAF lower than 2.5%
#keeping only genomic regions with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.025, min.nb.snps = 10)
```

```
#run burden test WSS, using a random continuous variable as phenotype
x1@ped$pheno <- rnorm(nrow(x1))
#Null model
x1.H0 <- NullObject.parameters(pheno = x1@ped$pheno,
                               RVAT = "burden", pheno.type = "continuous")
burden.continuous(x1, NullObject = x1.H0, burden = "WSS")
```

`burden.mlogit` *Logistic or multinomial regression on a genetic score*

Description

Performs a logistical or a non-ordinal multinomial regression on a genetic score

Usage

```
burden.mlogit(x, NullObject, genomic.region = x@snps$genomic.region, burden,
              maf.threshold = 0.5, get.OR.value = FALSE, alpha = 0.05, cores = 10)
```

Arguments

`x`	A bed matrix, only needed if `burden="CAST"` or `burden="WSS"`
`NullObject`	A list returned from `NullObject.parameters`
`genomic.region`	A factor containg the genomic region of each SNP, `x@snps$genomic.region` by default, only needed if `burden="CAST"` or `burden="WSS"`
`burden`	"CAST" or "WSS" to directly compute the CAST or the WSS genetic score; or a matrix with one row per individual and one column per `genomic.region` if another genetic score is wanted.
`maf.threshold`	The MAF threshold to use for the definition of a rare variant in the CAST score
`get.OR.value`	TRUE/FALSE: whether to return the OR values associated to the regression
`alpha`	The alpha threshold to use for the OR confidence interval
`cores`	How many cores to use for moments computation, set at 10 by default

Details

This function will return results from the regression of the phenotype on the genetic score for each genomic region.

If only two groups of individuals are present, a classical logistic regression is performed. If more than two groups of individuals are present, a non-ordinal multinomial regression is performed, comparing each group of individuals to the reference group indicated by the argument `ref.level` in `NullObject.parameters`. The choice of the reference group won't affect the p-values, but only the Odds Ratios. In both types of regression, the p-value is estimated using the Likelihood Ratio test.

If another genetic score than CAST or WSS is wanted, a matrix with one row per individual and one column per `genomic.region` containing this score should be given to `burden`. In this situation, no bed matrix `x` is needed.

Value

A dataframe with one row per genomic region and at least two columns:

`p.value`	The p.value of the regression
`is.err`	0/1: whether there was a convergence problem with the regression

If `get.OR.value=TRUE`, additional columns are present:

`OR`	The OR value(s) associated to the regression. If there are more than two groups, there will be one OR value per group compared to the reference group
`l.lower`	The lower bound of the confidence interval of each OR
`l.upper`	The upper bound of the confidence interval of each OR

References

Bocher O, et al. *Rare variant associationtesting for multicategory phenotype.* Genet.Epidemiol. 2019;43:646–656.

See Also

NullObject.parameters, burden.NullObject, CAST, WSS, burden.weighted.matrix

Examples

```
#Importation of data in a bed matrix
x <- as.bed.matrix(x=LCT.matrix.bed, fam=LCT.matrix.fam, bim=LCT.snps)

#Add population
x@ped[,c("pop", "superpop")] <- LCT.matrix.pop1000G[,c("population", "super.population")]

#Select EUR superpopulation
x <- select.inds(x, superpop=="EUR")
x@ped$pop <- droplevels(x@ped$pop)

#Group variants within known genes
x <- set.genomic.region(x)

#Filter of rare variants: only non-monomorphic variants with
#a MAF lower than 2.5%
#keeping only genomic regions with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.025, min.nb.snps = 10)

#Simulation of a covariate + Sex as a covariate
sex <- x1@ped$sex
set.seed(1) ; u <- runif(nrow(x1))
covar <- cbind(sex, u)

#run null model, using the 1000Genome population as "outcome"
#Null model with the covariate sex
x1.H0.covar <- NullObject.parameters(pheno = x1@ped$pop, ref.level = "CEU",
                                     RVAT = "burden", pheno.type = "categorial",
```

```
                                        data = covar, formula = ~ sex)
#WSS test
burden.mlogit(x1, NullObject = x1.H0.covar, burden = "WSS", get.OR.value=TRUE, cores = 1)
```

`burden.NullObject` *Null Model burden tests on multinomial phenotypes*

Description

Get the parameters under the null model to peforms burden tests on continuous phenotypes

Usage

```
burden.NullObject(group, ref.level, data, formula)
```

Arguments

`group`	A factor containing the group of each individual
`ref.level`	The reference group of individuals for the regression
`data`	Optional, a matrix containing the covariates with one column per covariate and one row per individual
`formula`	Optional, an R formula corresponding to the regression model indicating which covariates from `data` to include in the model if only some of them are to be included

Details

This function get the parameters under the null model for the burden tests: it computes the Log-Likelihood under the null model used to performe the Likelihood Ratio Test.

If `formula` is missing, all columns from data will be included as covariates.

Value

A list containing:

`group`	A factor containing the group of each individual as given
`ref.level`	The reference group of individuals for the regression as given
`H0.LogLik`	The Log-Likelihood of the null model
`covar.toinclude`	Which covariates to include in the regression, depends on the argument `formula`
`data`	The `data` argument containing covariates, NULL if it was missing

See Also

`NullObject.parameters`

`burden.NullObject.continuous` *Null Model burden tests on continuous phenotypes*

Description

Get the parameters under the null model to peforms burden tests on continuous phenotypes

Usage

```
burden.NullObject.continuous(pheno, data, formula)
```

Arguments

`pheno`	A numeric vector containing the phenotype value for each individual
`data`	Optional, a matrix containing the covariates with one column per covariate and one row per individual
`formula`	Optional, an R formula corresponding to the regression model indicating which covariates from `data` to include in the model if only some of them are to be included

Details

This function get the parameters under the null model for the burden tests: it creates an object containing parameters to run `burden` on a continuous phenotype.

If `formula` is missing, all columns from data will be included as covariates.

Value

A list containing:

`pheno`	A numeric vector containing the phenotype value for each individual as given
`covar.toinclude`	Which covariates to include in the regression, depends on the argument `formula`
`data`	The `data` argument containing covariates, NULL if it was missing

See Also

`NullObject.parameters`

`burden.weighted.matrix` *Score matrix for burden tests*

Description

Computes the score matrix for burden tests based on variants' weights

Usage

```
burden.weighted.matrix(x, weights, genomic.region = x@snps$genomic.region)
```

Arguments

`x`	A bed.matrix
`weights`	A vector containing the weight of each variant
`genomic.region`	A vector containing the genomic region of each variant

Details

For variant *i* and individual *j*, the genetic score will be computed as weight of variant *i* * number of minor alleles for individual *j*.

Value

A matrix containing the computed genetic score with one row per individual and one column per `genomic.region`.

See Also

`CAST`, `WSS`, `burden.mlogit`

Examples

```
#Importation of data in a bed matrix
x <- as.bed.matrix(x=LCT.matrix.bed, fam=LCT.matrix.fam, bim=LCT.snps)

# Group variants within known genes
x <- set.genomic.region(x)

# Filter variants with maf (computed on whole sample) < 0.025
# keeping only genomic region with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.025, min.nb.snps = 10)

#Compute burden score with weights = 1-maf
score.burden <- burden.weighted.matrix(x1, weights=1-x1@snps$maf)
```

CAST *Cohort Allelic Sum Test*

Description

Calculates the CAST genetic score

Usage

```
CAST(x, genomic.region = x@snps$genomic.region, maf.threshold = 0.5)
```

Arguments

x	A bed.matrix
genomic.region	A factor defining the genomic region of each variant
maf.threshold	The MAF used for the definition of a rare variant, set at 0.5 by default, i.e. all variants are kept

Value

A matrix containing the CAST genetic score with one row per individual and one column per genomic.region

References

Morgenthaler S and Thilly WG. *A strategy to discover genes that carry multi-allelic or mono-allelic risk for common diseases: a cohort allelic sums test (CAST).* Mutat Res. 2007

See Also

WSS, burden.weighted.matrix, burden.mlogit

Examples

```
#Importation of data in a bed matrix
x <- as.bed.matrix(x=LCT.matrix.bed, fam=LCT.matrix.fam, bim=LCT.snps)

# Group variants within known genes
x <- set.genomic.region(x)

# Filter variants with maf (computed on whole sample) < 0.025
# keeping only genomic region with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.025, min.nb.snps = 10)

# Compute burden score CAST
score.CAST <- CAST(x1, maf.threshold=0.025)
```

`filter.rare.variants` *Rare variants filtering*

Description

Filter rare variants based on a MAF threshold and a given number of SNP per genomic region

Usage

```
filter.rare.variants(x, ref.level, filter=c("whole", "controls", "any"),
                     maf.threshold=0.01, min.nb.snps, group)
```

Arguments

`x`	A bed.matrix
`ref.level`	The level corresponding to the controls group, only needed if `filter=="controls"`
`filter`	On which group the filter will be applied
`maf.threshold`	The MAF threshold used to define a rare variant
`min.nb.snps`	The minimum number of snps needed to keep a genomic region
`group`	A vector or factor indicating the group of each individual, if missing, `x@ped$pheno` is taken

Details

If `filter="whole"`, only the variants having a MAF lower than the threshold in the entire sample are kept.

If `filter="controls"`, only the variants having a MAF lower than the threshold in the controls group are kept.

If `filter="any"`, only the variants having a MAF lower than the threshold in any of the groups are kept.

Value

A bed.matrix with filtered variants

Examples

```
#Import 1000Genome data from region around LCT gene
x <- as.bed.matrix(LCT.gen, LCT.fam, LCT.bim)
table(x@snps$genomic.region, useNA="ifany")

#Group variants within known genes
x <- set.genomic.region(x)

#Filter of rare variants: only non-monomorphic variants with
#a MAF lower than 2.5%
```

```
#keeping only genomic regions with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.025, min.nb.snps = 10)
table(x1@snps$genomic.region, useNA="ifany")
```

`genes.positions` *Genes positions*

Description

Positions of human genes. These data were downloaded from Biomart on the Ensembl website with the GRCh37 and GRCh38 versions. Only genes present in GnomAD were kept.

Data contain the `Chr`, the `Start` position, the `End` position and the `Gene_Name` of all the genes in chromosomes 1 to 22 representing 19375 and 18278 genes in the two GRCh versions respectively.

Usage

```
data(genes.b37)
data(genes.b38)
```

Format

The data contain one dataframe with four columns:

`Chr` The chromosome of the gene

`Start` The start position of the gene

`End` The end position of the gene

`Gene_Name` The name of the gene

Source

The data were obtained from the Ensembl website (see http://grch37.ensembl.org/biomart/martview/ and http://www.ensembl.org/biomart/martview/).

See Also

`set.genomic.region`

genotypic.freq *Genotypic frequencies calculation for data simulations*

Description

Calculates the three genotypic frequencies in the controls group and each group of cases based on MAF in the general population and GRR values

Usage

```
genotypic.freq(genes.maf = Kryukov, GRR.het, GRR.homo.alt, prev,
               genetic.model = c("general", "multiplicative",
                                 "dominant", "recessive"), select.gene)
```

Arguments

genes.maf	A file containing the MAF in the general population (column maf) for variants with their associated gene (column gene), by default the file Kryukov is used
GRR.het	A matrix giving the GRR of the heterozygous genotype compared to the homozygous reference genotype with one row per cases group and one column per variant
GRR.homo.alt	A matrix giving the GRR of the homozygous alternative genotype compared to the homozygous reference genotype with one row per cases group and one column per variant, only need if genetic.model="general"
prev	A vector containing the prevalence of each group of cases
genetic.model	The genetic model of the disease
select.gene	Which gene to choose from genes.maf$gene if multiple genes are present. If missing, only the first level is kept.

Details

This function is used to simulate genetic data.

The genetic model of the disease needs to be specified to genetic.model:

If genetic.model="general", there is no link between the GRR associated to the heterozygous genotype and the GRR associated to the homozygous alternative genotype. Therefore, the user has to give two matrices of GRR, one for each of these genotypes.

If genetic.model="multiplicative", we assume that the GRR associated to the homozygous alternative genotype is the square of the GRR associated to the heterozygous genotype.

If genetic.model="dominant", we assume that the GRR associated to the heterozygous genotype and the GRR associated to the homozygous alternative genotype are equal.

If genetic.model="recessive", we assume that the GRR associated to the heterozygous genotype is equal to 1: the GRR given is the one associated to the homozygous alternative genotype.

prev corresponds to the proportion of each sub-group of cases in the population. It is used only to calculate the MAF in the controls group.

The dataframes Kryukov or GnomADgenes available with the package Ravages can be used for the argument genes.maf.

Value

A matrix of MAF values with one column per variant and one row per group (the first one being the controls group)

See Also

GRR.matrix, random.bed.matrix, GnomADgenes, Kryukov

Examples

```
#Construction of the GRR matrix using the formula from SKAT
#to compute the GRR (higher weights to rarer variants)
#GRR in the second group are twice as high as in the first group
GRR.del <- GRR.matrix(GRR = "SKAT", GRR.multiplicative.factor=2,
                      select.gene="R1")

#Calculation of frequency in the three groups of individuals
#under a multilpicative model of the disease
geno.freq.groups <- genotypic.freq(genes.maf = Kryukov, GRR.het = GRR.del,
                                   prev = c(0.001, 0.001), select.gene="R1",
                                   genetic.model = "multiplicative")
```

GnomADgenes *GnomADgenes dataset*

Description

This dataframe contains variants from the GnomAD database with MAF values in the Non-Finnish European (NFE) and their consequences from VEP with each associated gene.

Usage

```
data(GnomADgenes)
```

Format

GnomADgenes is a dataframe with five columns:

chr The chromosome of the variant

pos The position of the variant

consequence The functionnal consequence of the variant predicted by Variant Effect Predictor (VEP)

gene The gene associated to each variant predicted by VEP

maf The MAF of the variant in the NFE population

Source

The data were obtained from the GnomAD website (see http://gnomad.broadinstitute.org/) and the VEP website (see https://www.ensembl.org/info/docs/tools/vep/).

GRR.matrix *GRR matrix for genetic data simulation*

Description

Computes a GRR matrix based on a simulation model

Usage

```
GRR.matrix(genes.maf = Kryukov, n.case.groups = 2,
           GRR = c("SKAT", "constant", "variable"),
           GRR.value, GRR.function, GRR.multiplicative.factor = 2, select.gene)
```

Arguments

genes.maf	A dataframe containing at least the MAF in the general population (column maf) with their associated gene (column gene). By default, maf from the file Kryukov are used
n.case.groups	The number of cases groups (set at 2 by default), i.e. the number of groups where variants will have a GRR greater than 1
GRR	How to calculate the GRR
GRR.value	GRR value if GRR="constant"
GRR.function	A function indicating how to calculate the GRR depending on MAF in the general population, only needed if GRR="variable"
GRR.multiplicative.factor	A vector of size (n.case.groups-1) containing the multiplicative factor for the GRR for each group of cases compared to the first group of cases
select.gene	The gene(s) to be selected from the file genes.maf if multiple genes are present. If missing, the first level of genes.maf$gene is kept.

Details

The GRR can be computed in three ways using the argument GRR.

If GRR="constant", the same GRR is given to all the variants, its value being specified to GRR.value. If GRR="SKAT", the GRR are calculating using the formula from the paper presenting the SKAT method and thus depend on MAF. If GRR="variable", the GRR are calculating using a function given by the user to GRR.function depending only on the MAF in the general population.

The argument multiplicative.factor contains n.case.groups-1 values; if multiplicative.factor=1, GRR will be the same between the different groups of cases.

The two dataframes Kryukov (used by default) and GnomADgenes (containing MAF in the NFE population) can be used as genes.maf.

GRR.matrix returns a matrix that can be used in other simulation functions such as random.bed.matrix.

Value

A matrix containing the GRR values with one column per variant and one line per cases group

See Also

random.bed.matrix, GnomADgenes, Kryukov

Examples

```
#GRR calculated on the MAF from the first unit of the file Kryukov
#using the formula from the SKAT paper, with the second group of cases
#having GRR values twice as high as the first one
GRR.del <- GRR.matrix(GRR = "SKAT", genes.maf = Kryukov,
                      GRR.multiplicative.factor=2, select.gene = "R1")
```

Jaccard *Jaccard index*

Description

Calculates the Jaccard index for each pair of individuals using a bed.matrix

Usage

```
Jaccard(x, maf.threshold = 0.01)
```

Arguments

x	A bed.matrix
maf.threshold	The MAF used for the definition of a rare variant, set at 0.01 by default

Details

The individuals carrying no rare variants will have a null Jaccard index with all the individuals including themselves.

Value

A squared matrix giving the Jaccard index for each pair of individuals

References

Jaccard, P. (1908) *Nouvelles researches sur la distribution florale*, Bulletin de la Société vaudoise des sciences naturelles, **44, 223-270**

Examples

```
#Simulation of genetic data with GRR values according to the SKAT formula
GRR.del <- GRR.matrix(GRR = "SKAT", genes.maf = Kryukov,
                      n.case.groups = 2, select.gene = "R1",
                      GRR.multiplicative.factor=2)

#Simulation of one group of 1,000 controls and two groups of 500 cases,
#50% of causal variants, 5 genomic regions are simulated.
x <- random.bed.matrix(genes.maf=Kryukov, size = c(1000, 500, 500),
                       prev = c(0.001, 0.001), select.gene = "R1",
                       GRR.matrix.del = GRR.del, p.causal = 0.5,
                       genetic.model = "multiplicative", replicates = 5)

#Calculate the Jaccard matrix
J <- Jaccard(x, maf.threshold = 0.01)
```

Kryukov *Kryukov data set*

Description

The data from *Kryukov et al, 2009*, contain simulated site frequency spectrum data using European demographic models with purifying selection.

Usage

```
data(Kryukov)
```

Format

Kryukov is a dataframe with four columns:

gene The unit of each variant

maf The maf of each variant in the European population

selection.coefficient The selction coefficient of each variant in the European population

position The position of each variant

Details

200 units are present corresponding to 200 genes. For each unit, the data set contains the maf in the European population, the selection coefficient and the position of each variant.

Source

The data were obtained from the SeqPower software (see also http://www.bioinformatics.org/spower/input#data_download).

References

Kryukov et al, 2009, *Power of deep, all-exon resequencing for discovery of human trait genes*, Proceedings of the National Academy of Sciences, DOI:10.1073/pnas.0812824106

`LCT.haplotypes` *LCT haplotypes data set*

Description

These data contain the haplotype matrix `LCT.hap` of the 5008 individuals from the 1000 Genomes data for a ~300kb segment containing the Lactase gene. Information about individuals (sex, population and super population) is present in `LCT.sample`, and information about snps is available in `LCT.snps`.

Usage

```
data(LCT.haplotypes)
```

Format

Three data objects are present in `LCT.haplotypes`:

`LCT.hap` A matrix of haplotypes

`LCT.sample` A data frame with information on individuals (sex, population, super.population)

`LCT.snps` A data frame with information on snps (chr, id, dist, pos, A1, A2)

Source

Data were obtained from the 1000 Genomes Project (https://www.internationalgenome.org/).

References

McVean et al, 2012, *An integrated map of genetic variation from 1,092 human genomes*, Nature **491, 56-65** doi:10.1038/nature11632

See Also

`LCT.matrix`

LCT.matrix *LCT genotypes matrix*

Description

These data contain the genotype matrix corresponding to haplotypes present in LCT.haplotypes from the 1000 Genomes data for a ~300kb segment containing the Lactase gene. Information about individuals is present in LCT.matrix.fam, and information about population (population and super population) is present in LCT.matrix.pop1000G, in a format needed to generate a bedmatrix. LCT.snps from LCT.haplotypes can be used as the corresponding bim file of this genotypes matrix.

Usage

```
data(LCT.matrix)
```

Format

Three data objects are present in LCT.haplotypes:

LCT.matrix.bed The matrix of genotypes

LCT.matrix.fam The corresponding fam file

LCT.matrix.pop1000G A data frame with population information for individuals (population, superpopulation)

Source

Data were obtained from the 1000 Genomes Project (https://www.internationalgenome.org/).

References

McVean et al, 2012, *An integrated map of genetic variation from 1,092 human genomes*, Nature **491, 56-65** doi:10.1038/nature11632

See Also

LCT.haplotypes

Examples

```
#Importation of data in a bed matrix
x <- as.bed.matrix(x=LCT.matrix.bed, fam=LCT.matrix.fam, bim=LCT.snps)
#Add population
x@ped[,c("pop", "superpop")] <- LCT.matrix.pop1000G[,c("population", "super.population")]
```

`NullObject.parameters` *Null Model for SKAT and burden tests*

Description

Get the parameters under the null model to peforms burden tests or SKAT

Usage

```
NullObject.parameters(pheno, RVAT, pheno.type, ref.level, data, formula)
```

Arguments

`pheno`	The phenotype of each individual: a factor if `pheno.type = "categorial"`, and a numeric vector if `pheno.type = "continuous"`
`RVAT`	The type of Rare Variant Association Test (RVAT) to perform: should be "burden" or "SKAT"
`pheno.type`	The type of phenotype: "categorial" for binary or multinomial traits, or "continuous"
`ref.level`	The reference group of individuals for the regression, only needed if `RVAT = "burden"`
`data`	Optional, a matrix containing the covariates with one column per covariate and one row per individual
`formula`	Optional, an R formula corresponding to the regression model indicating which covariates from `data` to include in the model if only some of them are to be included

Details

This function get the parameters under the null model for SKAT or the burden tests: For burden tests, it computes the Log-Likelihood under the null model used to performe the Likelihood Ratio Test. For SKAT, it computes the probabilites for each individual of belonging to each group based on the group sizes and the potential covariates.

If `formula` is missing, all columns from data will be included as covariates.

Value

A list containing different elements depending on the `RVAT` performed and the `pheno.type`. See individual help pages for the corresponding functions: - `burden.NullObject` if `RVAT = "burden"` and `pheno.type = "categorial"` - `burden.NullObject.continuous` if `RVAT = "burden"` and `pheno.type = "continuous"` - `SKAT.NullObject` if `RVAT = "SKAT"` and `pheno.type = "categorial"` - `SKAT.NullObject.continuous` if `RVAT = "SKAT"` and `pheno.type = "continuous"`

See Also

`burden.NullObject`, `burden.NullObject.continuous`, `SKAT.NullObject`, `SKAT.NullObject.continuous`

`random.bed.matrix` *Simulation of genetic data using GRR values*

Description

Generates a simulated bed.matrix with genotypes for cases and controls based on GRR values

Usage

```
random.bed.matrix(genes.maf = Kryukov, size, prev, replicates,
                  GRR.matrix.del, GRR.matrix.pro,
                  p.causal = 0.5, p.protect = 0, same.variant = FALSE,
                  genetic.model=c("general", "multiplicative",
                                  "dominant", "recessive"), select.gene)
```

Arguments

`genes.maf`	A dataframe containing at least the MAF in the general population (column maf) for variants with their associated gene (column gene), by default the file `Kryukov` is used
`size`	A vector containing the size of each group (the first one being the control group)
`prev`	A vector containing the prevalence of each group of cases
`replicates`	The number of simulations to perform
`GRR.matrix.del`	A list containing the GRR matrix associated to the heterozygous genotype compared to the homozygous reference genotype as if all variants are deleterious. An additional GRR matrix associated to the homozygous for the alternate allele is needed if `genetic.genetic.model="general"`
`GRR.matrix.pro`	The same argument as `GRR.matrix.del` but for protective variants
`p.causal`	The proportion of causal variants in cases
`p.protect`	The proportion of protective variants in cases among causal variants
`same.variant`	TRUE/FALSE: whether the causal variants are the same in the different groups of cases
`genetic.model`	The genetic model of the disease
`select.gene`	Which gene to choose from `genes.maf$gene` if multiple genes are present. If missing, only the first level is kept.

Details

The genetic model of the disease needs to be specified in this function.

If `genetic.model="general"`, there is no link between the GRR for the heterozygous genotype and the GRR for the homozygous alternative genotype. Therefore, the user has to give two matrices of GRR, one for the heterozygous genotype, the other for the homozygous alternative genotype.

If `genetic.model="multiplicative"`, we assume that the the GRR for the homozygous alternative genotype is the square of the GRR for the heterozygous genotype.

If genetic.model="dominant", we assume that the GRR for the heterozygous genotype and the GRR for the homozygous alternative genotype are equal.

If genetic.model="recessive", we assume that the GRR for the heterozygous genotype is equal to 1: the GRR given is the one associated to the homozygous alternative genotype.

GRR.matrix.del contains GRR values as if all variants are deleterious. These values will be used only for the proportion p.causal of variants that will be sampled as causal.

The files Kryukov or GnomADgenes available with the package Ravages can be used as the argument genes.maf.

If GRR.matrix.del (or GRR.matrix.pro) has been generated using the function GRR.matrix, the arguments genes.maf and select.gene should have the same value as in GRR.matrix.

Value

A bed.matrix with as much columns (variants) as replicates*number of variants. The field x@snps$genomic.region contains the replicate number and the field x@ped$pheno contrains the group of each individual, "0" being the controls group.

See Also

GRR.matrix, Kryukov, GnomADgenes

Examples

```
#GRR values calculated with the SKAT formula
GRR.del <- GRR.matrix(GRR = "SKAT", genes.maf = Kryukov,
                      n.case.groups = 2, select.gene = "R1",
                      GRR.multiplicative.factor=2)

#Simulation of one group of 1,000 controls and two groups of 500 cases,
#each one with a prevalence of 0.001
#with 50% of causal variants, 5 genomic regions are simulated.
x <- random.bed.matrix(genes.maf = Kryukov, size = c(1000, 500, 500),
                       prev = c(0.001, 0.001), GRR.matrix.del = GRR.del,
                       p.causal = 0.5, p.protect = 0, select.gene="R1",
                       same.variant = FALSE,
                       genetic.model = "multiplicative", replicates = 5)
```

---

rbm.haplos.freqs *Simulation of genetic data based on haplotypic frequencies*

Description

Simulates genetic data with respect to allele frequency spectrum and linkage disequilibrium pattern observed on given haplotypes and their frequencies

Usage

```
rbm.haplos.freqs(haplos, freqs, size, replicates)
```

Arguments

haplos	A matrix of haplotypes with one row per haplotype and one column per variant
freqs	A matrix of haplotypes frequencies in each group of individuals
size	The sizes of each group of individuals
replicates	The number of simulations to perform

Details

Simulations are performed to respect linkage disequilibrium pattern and allelic frequency spectrum in each group of individuals The phenotypic values will be the colnames of `freqs` and stored in `@ped$pheno`. The simulation number will be in `@snps$genomic.region`.

Value

x	A bed matrix with simulated genotypes

Examples

```
#Simulations of 5 groups of individuals with haplotypes frequencies
#from the 5 EUR populations

#Load LCT dataset for haplotype matrix
data(LCT.haplotypes)
#Haplotypes for the variants in the LCT gene in the EUR population
LCT.gene.hap <- LCT.hap[which(LCT.sample$super.population=="EUR"),
                        which(LCT.snps$pos>=136545410 & LCT.snps$pos<=136594750)]

#Individuals from EUR
LCT.sample.EUR <- subset(LCT.sample, super.population=="EUR")
#Matrix of haplotypic frequencies
LCT.freqs <- sapply(unique(LCT.sample.EUR$population), function(z)
                    ifelse(LCT.sample.EUR$population==z,
                           1/table(LCT.sample.EUR$population)[z], 0))

#Simulation of genetic data for five groups of 50 individuals
x <- rbm.haplos.freqs(haplos=LCT.gene.hap, freqs=LCT.freqs, size=rep(50,5), replicates=5)
```

---

rbm.haplos.thresholds *Simulation of genetic data based on haplotypes and a libaility model*

Description

Simulates genetic data with respect to allele frequency spectrum and linkage disequilibrium pattern observed on given haplotype data under a libaility model

Usage

```
rbm.haplos.thresholds(haplos, weights = -0.4*log10(colMeans(haplos)),
                      maf.threshold = 0.01, nb.causal, p.protect = 0,
                      h2, prev, normal.approx = TRUE, size,
                      replicates, rep.by.causal)
```

Arguments

haplos	A matrix of haplotypes with one row per haplotype and one column per variant
weights	A vector of weights for each variant to compute the burden used in the liability model. By default, wieghts=-0.4*log10(MAF)
maf.threshold	The maf threshold to consider a rare variant (set at 0.01 by default), variants with a MAF upper this threshold will have a weight of 0
nb.causal	The number of causal variants
p.protect	The proportion of protective variants among causal variants
h2	The variance explained by the gene
prev	A vector with the prevalence in each group of individuals
normal.approx	TRUE/FALSE: whether to use the normal approximation to compute thresholds. Set at TRUE by default
size	The sizes of each group of individuals
replicates	The number of simulations to perform
rep.by.causal	The number of time causal variants will be sampled

Details

nb.causal, p.protect, h2 and prev should be vectors of length corresponding to the number of groups to simulate. If they are of size 1, values will be duplicated.

All monomorphic variants and variants with a MAF higher than maf.threshold will have a weight of 0. Causal variants are sampled among variants having weights greater than 0.

A liability model is built on haplotypes' burden computed on sampled causal variants using each variant's weights, and adjusted on the desired h2. Thresholds from this liability are then chosen to respect the given prev (from a standard normal distribution if normal.approx=TRUE, or using a distribution from 1e6 sampled burdens if normal.approx=FALSE). Please be carreful when using the normal approximation with high h2 values or low prev values. Haplotypes' probabilities in each group of individuals are then computed and two haplotypes are then sampled for each individual based on these probabilities.

To simulate a group of controls, prev needs to be set at 1, regardless of the other arguments.

N replicates will be performed, and to gain in computation time, the same causal variants can be used for multiple replicates as different haplotypes will be sampled for each individual. rep.by.causal indicates the number of replicates to perform for each set of causal variants. To ensure a variability in the simulations, we yet recommend to resample causal variants a few times when many replicates are to be performed. For example, if 1000 replicates are to be performed, we recommend to resample causal variants 20 times.

The phenotype will be stored in @ped$pheno, and the simulation number is @snps$genomic.region.

Value

x	A bed matrix with simulated genotypes

Examples

```
#Load LCT dataset for haplotype matrix
data(LCT.haplotypes)
#LCT gene in the EUR population
LCT.gene.hap <- LCT.hap[which(LCT.sample$super.population=="EUR"),
                        which(LCT.snps$pos>=136545410 & LCT.snps$pos<=136594750)]

#Simulation of 100 controls, and two groups of 50 cases with 30 causal variants
#and with the second group having half h2 and twice the prevalence
#compared to the first one
#5 replicates are performed and causal variants are sampled once
x <- rbm.haplos.thresholds(haplos=LCT.gene.hap, maf.threshold = 0.01, nb.causal=30,
                           p.protect=0, h2=c(0.01, 0.01, 0.02), prev=c(1, 0.01, 0.005),
                            size=c(100, 50, 50), replicates = 5, rep.by.causal = 5)
```

---

set.genomic.region *Variants annotation based on gene positions*

Description

Attributes genes to variants based on known genes positions

Usage

```
set.genomic.region(x, genes = genes.b37, flank.width = 0L)
```

Arguments

x	A bed.matrix
genes	A dataframe containing the fields : Chr (the chromosome of the gene), Start (the start position of the gene), End (the end position of the gene), and Gene_Name (the name of the gene - a factor),
flank.width	An integer: width of the flanking regions in base pairs downstream and upstream the genes.

Details

Warnings: genes$Gene_Name should be a factor containing the names of the genes, ordered in the genome order.

If x@snps$chr is not a vector of integers, it should be a factor with same levels as genes$Chr. We provide two data sets of autosomal humain genes, genes.b37 and genes.b38.

If `flank.width` is null, only the variants having their position between the `genes$Start` and the `genes$End` of a gene will be attributed to the corresponding gene. When two genes overlap, variants in the overlapping zone will be assigned to those two genes, separated by a comma.

If `flank.width` is a positive number, variants `flank.width` downstream or upstream a gene will be annotated annotated to this gene. You can use `flank.width = Inf` to have each variant attributed to the nearest gene.

Value

The same bed matrix as x with an additional column `x@snps$genomic.region` containing the annotation of each variant (character).

See Also

`genes.b37`, `genes.b38`

Examples

```
#Import 1000Genome data from region around LCT gene
x <- as.bed.matrix(LCT.gen, LCT.fam, LCT.bim)

#Group variants within known genes
x <- set.genomic.region(x)

#Group variants within know genes +/- 500bp
x <- set.genomic.region(x, flank.width=500)
```

SKAT *SKAT test*

Description

Peforms SKAT on categorial or binary phenotypes

Usage

```
SKAT(x, NullObject, genomic.region = x@snps$genomic.region,
     weights = (1 - x@snps$maf)**24, maf.threshold = 0.5,
     get.moments = "size.based", estimation.pvalue = "kurtosis",
     params.sampling, cores = 10, debug = FALSE)
```

Arguments

`x`	A bed.matrix
`NullObject`	A list returned from `NullObject.parameters`
`genomic.region`	A factor defining the genomic region of each variant

`weights` A vector with the weight of each variant. By default, the weight of each variant is inversely proportionnal to its MAF, as it was computed in the original SKAT method

`maf.threshold` The MAF above which variants are removed (default is to keep all variants)

`get.moments` How to estimate the moments to compute the p-values among "size.based", "bootstrap", "permutations", or "theoretical" for categorial phenotypes (2 or more groups of individuals). By default "size.based" that will choose the method depending on sample size (see `details`)

`estimation.pvalue`
Whether to use the skewness ("skewness") or the kurtosis ("kurtosis") for the chi-square approximation

`params.sampling`
A list containing the elements "perm.target", "perm.max", "debug". Only needed if `get.moments = "boostrap"` or `get.moments = "permutations"`

`cores` How many cores to use for moments computation, set at 10 by default

`debug` Whether to return the mean, standard deviation, skewness and kurtosis of the statistics

Details

The p-value is calculated using a chi-square approximation based on the statistics' moments. The user has to choose how to compute these moments (argument `get.moments`), and which moments to use for the chi-square approximation (argument `estimation.pvalue`): The moments can be computed either using a sampling procedure (`"permutations"` if there are no covariates, or `"bootstrap"` otherwise), or using theoretical moments computed as in Liu et al. 2008 (`"theoretical"`). If `get.moments = "size.based"`, the sampling procedure will be used for sample sizes lower than 2000, and the theoretical calculations otherwise.

To estimate the p-values, etiher the first three moments are used (`estimation.pvalue = "skewness"`), or the moments 1, 2 and 4 are used (`estimation.pvalue = "kurtosis"`).

If `get.moments = "theoretical"` and `estimation.pvalue = "skewness"`, it corresponds to `method = "liu"` in the SKAT package. If `get.moments = "theoretical"` and `estimation.pvalue = "kurtosis"`, it corresponds to `method = "liu.mod"` in the SKAT package.

For small samples, p-values estimation is based on sampling and a sequential procedure: permutated statistics are computed and each one is compared to the observed statistics. This method requires `perm.target` and `perm.max` that should be given as a list to `params.bootstrap`. If `params.bootstrap` is not specified, perm.target will be set at 100, perm.max at 5e4. The boostrap progam stops when either `perm.target` or `perm.max` is reached. P-values are then computed using a mixed procedure:

if `perm.target` is reached, the p-value is computed as : `perm.target` divided by the number of permutations used to reach `perm.target`;

if `perm.max` is reached, the SKAT small sample procedure is used, and p-values are approximated using a chi-square distributions based on statistics' moments 1, 2 and 4 computed from the permutated values.

If `debug=TRUE`, more informations about the estimated statistics moments are given.

If SKAT.NullObject.continuous has been used to generate NullObject, the method from Liu et al. will be used to compute the p-value for the continuous phenotype, but estimation.pvalue can be set at "skewness" or "kurtosis".

Value

A data frame containing for each genomic region:

stat	The observed statistics
p.value	The p-value of the test

If get.moments = "bootstrap" or get.moments = "permutations", additional fields are present:

p.perm	The p-value computed by permutations: number of times permutated is greater than observed statistics divided by the total number of permutations performed
p.chi2	The p-value computed by the chi-square approximation using the SKAT small sample procedure

If debug = TRUE, the mean, standard deviation, skewness and kurtosis are also returned, as well as for the sampling procedure:

nb.gep	The number of times a permutated statistics is equal or greater than the observed statistics stat
nb.eq	The number of times a permutated statistics is equal to the observed statistics stat
nb.perms	The total number of simulations performed

References

Wu et al. 2011, *Rare-variant association testing for sequencing data with the sequence kernel association test*, American Journal of Human Genetics **82-93** doi:10.1016/j.ajhg.2011.05.029;

Lee et al. 2012, *Optimal Unified Approach for Rare-Variant Association Testing with Application to Small-Sample Case-Control Whole-Exome Sequencing Studies*, American Journal of Human Genetics, doi:10.1016/j.ajhg.2012.06.007;

Liu et al. 2008, *A new chi-square approximation to the distribution of non-negative definite quadratic forms in non-central normal variables*, Computational Statistics & Data Analysis, doi:10.1016/j.csda.2008.11.025

See Also

NullObject.parameters, SKAT.NullObject.continuous, SKAT.NullObject, SKAT.theoretical, SKAT.bootstrap, SKAT.permutations

Examples

```
#Importation of data in a bed matrix
x <- as.bed.matrix(x=LCT.matrix.bed, fam=LCT.matrix.fam, bim=LCT.snps)
#Add population
x@ped[,c("pop", "superpop")] <- LCT.matrix.pop1000G[,c("population", "super.population")]
```

```
#Select EUR superpopulation
x <- select.inds(x, superpop=="EUR")
x@ped$pop <- droplevels(x@ped$pop)

#Group variants within known genes
x <- set.genomic.region(x)

#Filter of rare variants: only non-monomorphic variants with
#a MAF lower than 2.5%
#keeping only genomic regions with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.025, min.nb.snps = 10)

#Simulation of a covariate + Sex as a covariate
sex <- x1@ped$sex
set.seed(1) ; u <- runif(nrow(x1))
covar <- cbind(sex, u)

#run SKAT using the 1000 genome EUR populations as "outcome"
#The maximum number of permutations used is 500, please increase
#this number for a more accurate estimation of the p-values
#Fit Null model
x1.H0 <- NullObject.parameters(x1@ped$pop, RVAT = "SKAT", pheno.type = "categorial")
#Parameters for the sampling procedure
params.sampling = list(perm.target = 100, perm.max = 500)

#Run SKAT using the method based on sample size, here permutations
SKAT(x1, x1.H0, params.sampling = params.sampling)

#Run SKAT with the covariates: use boostrap as n<2000
x1.H0.covar <- NullObject.parameters(x1@ped$pop, RVAT = "SKAT", pheno.type = "categorial",
                                     data = covar, formula = ~ sex)
SKAT(x1, x1.H0.covar, params.sampling = params.sampling, get.moments = "bootstrap")

## Not run:
#Run SKAT using theoretical moments (discourage here as n<2000) and 1 core
SKAT(x1, x1.H0, get.moments = "theoretical", cores = 1)

## End(Not run)

#Run SKAT with a random continuous phenotype
#Null Model
x1.H0.c <- NullObject.parameters(rnorm(nrow(x1)), RVAT = "SKAT", pheno.type = "continuous")
SKAT(x1, x1.H0.c)
```

SKAT.bootstrap *Multi group SKAT test using bootstrap sampling*

Description

Peforms SKAT on two or more groups of individuals using bootstrap sampling

Usage

```
SKAT.bootstrap(x, NullObject, genomic.region = x@snps$genomic.region,
               weights = (1-x@snps$maf)**24, maf.threshold = 0.5,
               perm.target = 100, perm.max = 5e4, debug = FALSE,
               estimation.pvalue = "kurtosis")
```

Arguments

x	A bed.matrix
NullObject	A list returned from NullObject.parameters
genomic.region	A factor defining the genomic region of each variant
weights	A vector with the weight of each variant. By default, the weight of each variant is inversely proportionnal to its MAF, as it was computed in the original SKAT method
maf.threshold	The MAF above which variants are removed (default is to keep all variants)
perm.target	The number of times to exceed the observed statistics. If not reached, perm.max permutations will be used
perm.max	The maximum number of permutations to perform to estimate the p-value, will be used if perm.target is not reached
debug	Whether to print details about the permutations (mean, standard deviation, skewness, kurtosis), FALSE by default
estimation.pvalue	Whether to use the skewness ("skewness") or the kurtosis ("kurtosis") for the chi-square approximation

Details

P-values estimation is based on bootstrap sampling and a sequential procedure: permutated statistics are computed and each one is compared to the observed statistics. The boostrap progam stops when either perm.target or perm.max is reached. P-values are then computed using a mixed procedure:

if perm.target is reached, the p-value is computed as : perm.target divided by the number of permutations used to reach perm.target;

if perm.max is reached, p-values are approximated using a chi-square distributions based on the first three moments if estimation.pvalue = "skewness", or on statistics' moments 1, 2 and 4 if estimation.pvalue = "kurtosis".

If debug=TRUE, more informations about the estimated statistics moments are given.

This function is used by SKAT when the sample size is smaller than 2000 and covariates are present.

Value

A data frame containing for each genomic:

stat	The observed statistics
p.value	p.perm if perm.target is reached, p.chi2 if perm.max is reached.

p.perm	The p-value computed by permutations: number of times permutated is greater than observed statistics divided by the total number of permutations performed
p.chi2	The p-value computed by the chi-square approximation using the SKAT small sample procedure

If debug=TRUE, other informations are given about the moments estimation:

nb.gep	The number of times a permutated statistics is equal or greater than the observed statistics stat
nb.eq	The number of times a permutated statistics is equal to the observed statistics stat
nb.perms	The total number of simulations performed
mean	The mean of the permutated statistics
sigma	The standard deviation of the permutated statistics
skewness	The skweness of the permutated statistics
kurtosis	The kurtosis of the permutated statistics

References

Wu et al. 2011, *Rare-variant association testing for sequencing data with the sequence kernel association test*, American Journal of Human Genetics **82-93** doi:10.1016/j.ajhg.2011.05.029;

Lee et al. 2012, *Optimal Unified Approach for Rare-Variant Association Testing with Application to Small-Sample Case-Control Whole-Exome Sequencing Studies*, American Journal of Human Genetics, doi:10.1016/j.ajhg.2012.06.007;

See Also

NullObject.parameters, SKAT.NullObject, SKAT

Examples

```
#Importation of data in a bed matrix
x <- as.bed.matrix(x=LCT.matrix.bed, fam=LCT.matrix.fam, bim=LCT.snps)
#Add population
x@ped[,c("pop", "superpop")] <- LCT.matrix.pop1000G[,c("population", "super.population")]

#Select EUR superpopulation
x <- select.inds(x, superpop=="EUR")
x@ped$pop <- droplevels(x@ped$pop)

#Group variants within known genes
x <- set.genomic.region(x)

#Filter of rare variants: only non-monomorphic variants with
#a MAF lower than 1%
#keeping only genomic regions with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.01, min.nb.snps = 10)
```

```
#Simulation of a covariate + Sex as a covariate
sex <- x1@ped$sex
set.seed(1) ; u <- runif(nrow(x1))
covar <- cbind(sex, u)

#run SKAT using the 1000 genome EUR populations as "outcome"
#The maximum number of permutations used is 100,
#and the target number is 10, please increase
#both values for a more accurate estimation of the p-values
#Fit Null model with covariates
x1.H0 <- NullObject.parameters(x1@ped$pop, data = covar, RVAT = "SKAT", pheno.type = "categorial")

SKAT.bootstrap(x1, x1.H0, perm.target = 10, perm.max = 100)
```

`SKAT.continuous` *Multi group SKAT test using Liu et al. approximation*

Description

Peforms SKAT on a continuous phenotype using Liu et al. approximation

Usage

```
SKAT.continuous(x, NullObject, genomic.region = x@snps$genomic.region,
                weights = (1 - x@snps$maf)**24, maf.threshold = 0.5,
                estimation.pvalue = "kurtosis", debug = FALSE )
```

Arguments

`x`	A bed.matrix
`NullObject`	A list returned from `NullObject.parameters`
`genomic.region`	A factor defining the genomic region of each variant
`weights`	A vector with the weight of each variant. By default, the weight of each variant is inversely proportionnal to its MAF, as it was computed in the original SKAT method
`maf.threshold`	The MAF above which variants are removed (default is to keep all variants)
`estimation.pvalue`	Whether to use the skewness ("skewness") or the kurtosis ("kurtosis") for the chi-square approximation
`debug`	Whether to return the mean, standard deviation, skewness and kurtosis of the statistics. Set at FALSE by default

Details

The method from Liu et al. 2008 is used where p-values are estimated using a chi-square approximation from moment's

If `estimation.pvalue = "kurtosis"`, the kurtosis is used instead of skewness in the chi-square approximation. This is equivalent to "liu.mod" in SKAT package.

Value

A data frame containing for each genomic region:

stat	The observed statistics
p.value	The p-value of the test

If debug = TRUE, the mean, standard deviation, skewness and kurtosis used to compute the p-value are returned

References

Wu et al. 2011, *Rare-variant association testing for sequencing data with the sequence kernel association test*, American Journal of Human Genetics **82-93** doi:10.1016/j.ajhg.2011.05.029;

Liu et al. 2008, *A new chi-square approximation to the distribution of non-negative definite quadratic forms in non-central normal variables*, Computational Statistics & Data Analysis, doi:10.1016/j.csda.2008.11.025

See Also

NullObject.parameters, SKAT.NullObject.continuous, SKAT

Examples

```
#Importation of data in a bed matrix
x <- as.bed.matrix(x=LCT.matrix.bed, fam=LCT.matrix.fam, bim=LCT.snps)
#Add population
x@ped[,c("pop", "superpop")] <- LCT.matrix.pop1000G[,c("population", "super.population")]

#Select EUR superpopulation
x <- select.inds(x, superpop=="EUR")
x@ped$pop <- droplevels(x@ped$pop)

#Group variants within known genes
x <- set.genomic.region(x)

#Filter of rare variants: only non-monomorphic variants with
#a MAF lower than 2.5%
#keeping only genomic regions with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.025, min.nb.snps = 10)

#run SKAT using a random continuous phenotype
#Fit Null model
x1.H0 <- NullObject.parameters(rnorm(nrow(x1)), RVAT = "SKAT", pheno.type = "continuous")

SKAT.continuous(x1, x1.H0)
```

SKAT.NullObject *Null Model for multi group SKAT test*

Description

Get the parameters under the null model to peforms SKAT

Usage

```
SKAT.NullObject(group, data = NULL, formula)
```

Arguments

group	A factor containing the group of each individual
data	Optional, a matrix containing the covariates with one column per covariate and one row per individual
formula	Optional, an R formula corresponding to the regression model indicating which covariates from data to include in the model if only some of them are to be included

Details

This function computes the probabilites for each individual of belonging to each group based on the group sizes and the potential covariates.

If formula is missing, all columns from data will be included as covariates.

Value

A list containing:

Pi.data	A matrix n.individuals x n.groups containing the probabilities that each individual belong to each group
X	A matrix containing 1 in the first column for the intercept, and covariates from data and formula
group	A factor containing the group of each individual as given
get.moments	How to compute moments based on sample size for p-value calculations (only used if get.moments = "size.based" for a categorial phenotype in SKAT.
P1	The vairance-covariance matrix of (Y - Pi_hat)

See Also

NullObject.parameters, SKAT, SKAT.theoretical, SKAT.bootstrap, SKAT.permutations

SKAT.NullObject.continuous

Null Model for multi group SKAT test

Description

Get the parameters under the null model to peforms SKAT

Usage

```
SKAT.NullObject.continuous(pheno, data = NULL, formula)
```

Arguments

pheno	A numeric vector containing the phenotype value for each individual
data	Optional, a matrix containing the covariates with one column per covariate and one row per individual
formula	Optional, an R formula corresponding to the regression model indicating which covariates from data to include in the model if only some of them are to be included

Details

This function computes the probabilites for each individual of belonging to each group based on the group sizes and the potential covariates.

If formula is missing, all columns from data will be included as covariates.

Value

A list containing:

ymp	A matrix n.individuals x 1 containing the (y - pi_hat) values, i.e. the residuals from the regression of the phenotype on the potential covariates
X	A matrix containing 1 in the first column for the intercept, and covariates from data and formula
pheno	The phenotype of each individual as given
P1	The variance matrix of ymp

See Also

NullObject.parameters, SKAT, SKAT.continuous

`SKAT.permutations` *Multi group SKAT test using bootstrap sampling*

Description

Peforms SKAT on two or more groups of individuals using bootstrap sampling

Usage

```
SKAT.permutations(x, NullObject, genomic.region = x@snps$genomic.region,
                  weights = (1-x@snps$maf)**24, maf.threshold = 0.5,
                  perm.target = 100, perm.max = 5e4, debug = FALSE,
                  estimation.pvalue = "kurtosis")
```

Arguments

`x`	A bed.matrix
`NullObject`	A list returned from `NullObject.parameters`
`genomic.region`	A factor defining the genomic region of each variant
`weights`	A vector with the weight of each variant. By default, the weight of each variant is inversely proportionnal to its MAF, as it was computed in the original SKAT method
`maf.threshold`	The MAF above which variants are removed (default is to keep all variants)
`perm.target`	The number of times to exceed the observed statistics. If not reached, `perm.max` permutations will be used
`perm.max`	The maximum number of permutations to perform to estimate the p-value, will be used if `perm.target` is not reached
`debug`	Whether to print details about the permutations (mean, standard deviation, skewness, kurtosis), FALSE by default
`estimation.pvalue`	Whether to use the skewness ("skewness") or the kurtosis ("kurtosis") for the chi-square approximation

Details

P-values estimation is based on permutations sampling and a sequential procedure: permutated statistics are computed and each one is compared to the observed statistics. The boostrap progam stops when either `perm.target` or `perm.max` is reached. P-values are then computed using a mixed procedure:

if `perm.target` is reached, the p-value is computed as : `perm.target` divided by the number of permutations used to reach `perm.target`;

if `perm.max` is reached, p-values are approximated using a chi-square distributions based on the first three moments if `estimation.pvalue = "skewness"`, or on statistics' moments 1, 2 and 4 if `estimation.pvalue = "kurtosis"`.

If debug=TRUE, more informations about the estimated statistics moments are given.

This function is used by SKAT when the sample size is smaller than 2000 and no covariates are present.

Value

A data frame containing for each genomic:

stat	The observed statistics
p.value	p.perm if perm.target is reached, p.chi2 if perm.max is reached.
p.perm	The p-value computed by permutations: number of times permutated is greater than observed statistics divided by the total number of permutations performed
p.chi2	The p-value computed by the chi-square approximation using the SKAT small sample procedure

If debug=TRUE, other informations are given about the moments estimation:

nb.gep	The number of times a permutated statistics is equal or greater than the observed statistics stat
nb.eq	The number of times a permutated statistics is equal to the observed statistics stat
nb.perms	The total number of simulations performed
mean	The mean of the permutated statistics
sigma	The standard deviation of the permutated statistics
skewness	The skweness of the permutated statistics
kurtosis	The kurtosis of the permutated statistics

References

Wu et al. 2011, *Rare-variant association testing for sequencing data with the sequence kernel association test*, American Journal of Human Genetics **82-93** doi:10.1016/j.ajhg.2011.05.029;

Lee et al. 2012, *Optimal Unified Approach for Rare-Variant Association Testing with Application to Small-Sample Case-Control Whole-Exome Sequencing Studies*, American Journal of Human Genetics, doi:10.1016/j.ajhg.2012.06.007;

See Also

NullObject.parameters, SKAT.NullObject, SKAT

Examples

```
#Importation of data in a bed matrix
x <- as.bed.matrix(x=LCT.matrix.bed, fam=LCT.matrix.fam, bim=LCT.snps)
#Add population
x@ped[,c("pop", "superpop")] <- LCT.matrix.pop1000G[,c("population", "super.population")]

#Select EUR superpopulation
```

```
x <- select.inds(x, superpop=="EUR")
x@ped$pop <- droplevels(x@ped$pop)

#Group variants within known genes
x <- set.genomic.region(x)

#Filter of rare variants: only non-monomorphic variants with
#a MAF lower than 1%
#keeping only genomic regions with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.01, min.nb.snps = 10)

#run SKAT using the 1000 genome EUR populations as "outcome"
#The maximum number of permutations used is 100,
#and the target number is 10, please increase
#both values for a more accurate estimation of the p-values
#Fit Null model
x1.H0 <- NullObject.parameters(x1@ped$pop, RVAT = "SKAT", pheno.type = "categorial")
SKAT.permutations(x1, x1.H0, perm.target = 10, perm.max=100)
```

`SKAT.theoretical` *Multi group SKAT test using Liu et al. approximation*

Description

Peforms SKAT on two or more groups of individuals using Liu et al. approximation

Usage

```
SKAT.theoretical(x, NullObject, genomic.region = x@snps$genomic.region,
                 weights = (1 - x@snps$maf)**24, maf.threshold = 0.5,
                 estimation.pvalue = "kurtosis", cores = 10, debug = FALSE )
```

Arguments

`x`	A bed.matrix
`NullObject`	A list returned from `NullObject.parameters`
`genomic.region`	A factor defining the genomic region of each variant
`weights`	A vector with the weight of each variant. By default, the weight of each variant is inversely proportionnal to its MAF, as it was computed in the original SKAT method
`maf.threshold`	The MAF above which variants are removed (default is to keep all variants)
`estimation.pvalue`	Whether to use the skewness ("skewness") or the kurtosis ("kurtosis") for the chi-square approximation
`cores`	How many cores to use for moments computation, set at 10 by default
`debug`	Whether to return the mean, standard deviation, skewness and kurtosis of the statistics. Set at FALSE by default

Details

The method from Liu et al. 2008 is used where p-values are estimated using a chi-square approximation from moment's statistics

If `estimation.pvalue = "kurtosis"`, the kurtosis is used instead of skewness in the chi-square approximation. This is equivalent to "liu.mod" in SKAT package.

This function is used by `SKAT` when the sample size is larger than 2000.

Value

A data frame containing for each genomic region:

`stat`	The observed statistics
`p.value`	The p-value of the test

If debug = `TRUE`, the mean, standard deviation, skewness and kurtosis used to compute the p-value are returned

References

Wu et al. 2011, *Rare-variant association testing for sequencing data with the sequence kernel association test*, American Journal of Human Genetics **82-93** doi:10.1016/j.ajhg.2011.05.029;

Liu et al. 2008, *A new chi-square approximation to the distribution of non-negative definite quadratic forms in non-central normal variables*, Computational Statistics & Data Analysis, doi:10.1016/j.csda.2008.11.025

See Also

`NullObject.parameters`, `SKAT.NullObject`, `SKAT.theoretical`, `SKAT.bootstrap`, `SKAT`

Examples

```
#Importation of data in a bed matrix
x <- as.bed.matrix(x=LCT.matrix.bed, fam=LCT.matrix.fam, bim=LCT.snps)
#Add population
x@ped[,c("pop", "superpop")] <- LCT.matrix.pop1000G[,c("population", "super.population")]

#Select EUR superpopulation
x <- select.inds(x, superpop=="EUR")
x@ped$pop <- droplevels(x@ped$pop)

#Group variants within known genes
x <- set.genomic.region(x)

#Filter of rare variants: only non-monomorphic variants with
#a MAF lower than 2.5%
#keeping only genomic regions with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.025, min.nb.snps = 10)

#run SKAT using the 1000 genome EUR populations as "outcome" using one core
#Fit Null model
```

```
x1.H0 <- NullObject.parameters(x1@ped$pop, RVAT = "SKAT", pheno.type = "categorial")

SKAT.theoretical(x1, x1.H0, cores = 1)
```

WSS *WSS genetic score*

Description

Caluclates the WSS genetic score

Usage

```
WSS(x, genomic.region = x@snps$genomic.region)
```

Arguments

x	A bed.matrix
genomic.region	A vector containing the genomic region of each variant

Value

A matrix containing the WSS genetic score with one row per individual and one column per genomic.region

References

Madsen E and Browning S. *A Groupwise Association Test for Rare Mutations Using a Weighted Sum Statistic.* PLoS Genet. 2009

See Also

CAST, burden.weighted.matrix, burden.mlogit

Examples

```
#Importation of data in a bed matrix
x <- as.bed.matrix(x=LCT.matrix.bed, fam=LCT.matrix.fam, bim=LCT.snps)

# Group variants within known genes
x <- set.genomic.region(x)

# Filter variants with maf (computed on whole sample) < 0.025
# keeping only genomic region with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.025, min.nb.snps = 10)

# Compute burden score WSS
score.WSS <- WSS(x1)
```

Index