

Package ‘Ravages’

March 5, 2021

Type Package

Title Rare Variant Analysis and Genetic Simulations

Version 0.1.2

Date 2021-03-05

Encoding UTF-8

Author Ozvan Bocher and Hervé Perdry

Maintainer Ozvan Bocher <bocherozvan@gmail.com>

Description Rare variant association tests: burden tests (Bocher et al. 2019 <doi:10.1002/gepi.22210>) and the Sequence Kernel Association Test; 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, ddfx

NeedsCompilation yes

Suggests knitr, rmarkdown

VignetteBuilder knitr

LazyLoad yes

LazyData yes

R topics documented:

bed.matrix.split.genomic.region	2
burden	3
burden.continuous	5
burden.mlogit	7
burden.weighted.matrix	9
CAST	10
filter.rare.variants	11
genes.positions	12

genotypic.freq	13
GnomADgenes	14
GRR.matrix	15
Jaccard	16
Kryukov	17
LCT.haplotypes	18
LCT.matrix	19
NullObject.parameters	20
random.bed.matrix	22
rbm.haplos.freqs	24
rbm.haplos.thresholds	25
set.genomic.region	26
SKAT	28
SKAT.bootstrap	31
SKAT.continuous	34
SKAT.permutations	36
SKAT.theoretical	38
WSS	40
Index	42

bed.matrix.split.genomic.region
<i>Bed matrix for variants associated to multiple genomic regions</i>

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=factor(letters[1:4]))

#Attribute genomic regions
x.ex <- set.genomic.region(x.ex, regions = 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 categorical 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,
  verbose = TRUE)
```

Arguments

x	A bed matrix, only needed if burden="CAST" or burden="WSS"
NullObject	A list returned from NullObject.parameters
genomic.region	A factor containing 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. Set at 0.5 by default
get.OR.value	TRUE/FALSE: whether to return OR values

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 <code>NullObject\$pheno.type = "categorical"</code>
verbose	Whether to display information about the function actions

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 = "categorical"` 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. DOI: 10.1002/gepi.22210. *Rare variant association testing for multicategory phenotype*. Genet.Epidemiol. 2019;43:646–656.

See Also

[NullObject.parameters](#), [burden.continuous](#), [burden.mlogit](#), [CAST](#), [WSS](#), [burden.weighted.matrix](#)

Examples

```
#Import 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 = "categorical")

#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	<i>Linear regression on a genetic score</i>
-------------------	---

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

<code>x</code>	A bed matrix, only needed if <code>burden="CAST"</code> or <code>burden="WSS"</code>
<code>NullObject</code>	A list returned from <code>NullObject.parameters</code>
<code>genomic.region</code>	A factor containing the genomic region of each SNP, <code>x@snp\$genomic.region</code> by default, only needed if <code>burden="CAST"</code> or <code>burden="WSS"</code>
<code>burden</code>	"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 <code>genomic.region</code> if another genetic score is wanted.
<code>maf.threshold</code>	The MAF threshold to use for the definition of a rare variant in the CAST score. Set at 0.5 by default

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:

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

See Also

[CAST](#), [WSS](#), [burden.weighted.matrix](#)

Examples

```
#Import 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

<code>x</code>	A bed matrix, only needed if <code>burden="CAST"</code> or <code>burden="WSS"</code>
<code>NullObject</code>	A list returned from <code>NullObject.parameters</code>
<code>genomic.region</code>	A factor containing the genomic region of each SNP, <code>x@snps\$genomic.region</code> by default, only needed if <code>burden="CAST"</code> or <code>burden="WSS"</code>
<code>burden</code>	"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 <code>genomic.region</code> if another genetic score is wanted.
<code>maf.threshold</code>	The MAF threshold to use for the definition of a rare variant in the CAST score. Set at 0.5 by default
<code>get.OR.value</code>	TRUE/FALSE: whether to return OR values
<code>alpha</code>	The alpha threshold to use for the OR confidence interval
<code>cores</code>	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. DOI: 10.1002/gepi.22210. *Rare variant association testing for multicategory phenotype*. Genet.Epidemiol. 2019;43:646–656.

See Also

[NullObject.parameters](#), [CAST](#), [WSS](#), [burden.weighted.matrix](#)

Examples

```
#Import 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 = "categorical",
                                     data = covar, formula = ~ sex)

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

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

<code>x</code>	A bed.matrix
<code>weights</code>	A vector containing the weight of each variant
<code>genomic.region</code>	A factor 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

```
#Import 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

<code>x</code>	A bed.matrix
<code>genomic.region</code>	A factor defining the genomic region of each variant
<code>maf.threshold</code>	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

```
#Import 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	<i>Rare variants filtering</i>
----------------------	--------------------------------

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 = 2, group, genomic.region = NULL)
```

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, set at 0.01 by default
min.nb.snps	The minimum number of snps needed to keep a genomic region, set at 2 by default
group	A factor indicating the group of each individual, only needed if filter = "controls" or filter = "any". If missing, x@ped\$pheno is taken
genomic.region	An optional factor containing the genomic region of each variant, only needed if min.nb.snps is specified and if x@snps\$genomic.region doesn't exist

Details

To use this function, a factor 'genomic.region' should be present in the slot x@snps.

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)

#Group variants within known genes
x <- set.genomic.region(x)
table(x@snp$genomic.region, useNA="ifany")

#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@snp$genomic.region, useNA="ifany")
```

genes.positions	<i>Genes positions</i>
-----------------	------------------------

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.

References

- RJ Kinsella et al, 2011, *Ensembl BioMart: a hub for data retrieval across taxonomic space*, Database. doi:10.1093/database/bar030;
- AD Yates et al, 2020, *Ensembl 2020*, Nucleic Acide Research. doi:10.1093/nar/gkz966

See Also

[set.genomic.region](#)

genotypic.freq	<i>Genotypic frequencies calculation for data simulations</i>
----------------	---

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 multiplicative 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 in build version 37.

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 functional 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	<i>Jaccard index</i>
---------	----------------------

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 recherches 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 selection 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.

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.matrixLCT 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.

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

```
#Import 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 performs burden tests or SKAT

Usage

```
NullObject.parameters(pheno, RVAT, pheno.type = c("categorical", "continuous"),
  ref.level, data, formula)
```

Arguments

pheno	The phenotype of each individual: a factor if <code>pheno.type = "categorical"</code> , and a numeric vector if <code>pheno.type = "continuous"</code>
RVAT	The type of Rare Variant Association Test (RVAT) to perform: should be "burden" or "SKAT"
pheno.type	The type of phenotype: "categorical" for binary or multinomial traits, or "continuous"
ref.level	The reference group of individuals for the regression, only needed if <code>RVAT = "burden"</code> and <code>pheno.type = "categorical"</code>
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

Warning: individuals in `pheno` and `data` should be in the same order.

This function gets 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 perform the Likelihood Ratio Test.

For SKAT, it computes the probabilities 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`.

- if `RVAT = "burden"` and `pheno.type = "categorical"`:

`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
- if RVAT = "burden" and pheno.type = "continuous":	
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
- if RVAT = "SKAT" and pheno.type = "categorical":	
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 categorical phenotype in SKAT.
P1	The vairance-covariance matrix of (Y - Pi_hat)
- if RVAT = "SKAT" and pheno.type = "continuous":	
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

[SKAT](#), [burden](#)

Examples

```
#Random phenotype of 100 individuals
random.multi.pheno <- sample(1:3, 100, replace = TRUE)
#Random continuous phenotype
random.continuous.pheno <- rnorm(100)
#Random sex covariate
random.covar <- matrix( sample(1:2, prob = c(0.4, 0.6), size = 100, replace = TRUE),
                        ncol = 1 )

#Null Model for burden with a multinomi-category phenotype
```

```
#Controls as reference group, no covariates
H0.burden.multi <- NullObject.parameters(pheno = as.factor(random.multi.pheno),
                                         RVAT = "burden", pheno.type = "categorical", ref.level = 1)
#Null Model for SKAT with a continuous phenotype and a covariate
H0.SKAT.continuous <- NullObject.parameters(pheno = random.continuous.pheno,
                                             RVAT = "SKAT", pheno.type = "continuous",
                                             data = random.covar)
```

random.bed.matrix	<i>Simulation of genetic data using GRR values</i>
-------------------	--

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` contains 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 liability model*

Description

Simulates genetic data with respect to allele frequency spectrum and linkage disequilibrium pattern observed on given haplotype data under a liability 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, verbose = TRUE)
```

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, weights=-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
verbose	Whether to display information about the function actions

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 careful when using

the normal approximation with high h^2 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	<i>Variants annotation based on gene positions</i>
--------------------	--

Description

Attributes regions to variants based on given region positions

Usage

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

Arguments

<code>x</code>	A bed.matrix
<code>regions</code>	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),
<code>flank.width</code>	An integer: width of the flanking regions in base pairs downstream and upstream the regions.

Details

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

We provide two data sets of autosomal humain genes, `genes.b37` and `genes.b38`.

If `x@snps$chr` is not a vector of integers, it should be a factor with same levels as `regions$Chr`.

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

Performs SKAT on categorical 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, verbose = TRUE)
```

Arguments

<code>x</code>	A bed.matrix
<code>NullObject</code>	A list returned from <code>NullObject.parameters</code>
<code>genomic.region</code>	A factor defining the genomic region of each variant
<code>weights</code>	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
<code>maf.threshold</code>	The MAF above which variants are removed (default is to keep all variants)
<code>get.moments</code>	How to estimate the moments to compute the p-values among "size.based", "bootstrap", "permutations", or "theoretical" for categorical phenotypes (2 or more groups of individuals). By default "size.based" that will choose the method depending on sample size (see details)
<code>estimation.pvalue</code>	Whether to use the skewness ("skewness") or the kurtosis ("kurtosis") for the chi-square approximation
<code>params.sampling</code>	A list containing the elements "perm.target", "perm.max", "debug". Only needed if <code>get.moments = "bootstrap"</code> or <code>get.moments = "permutations"</code>
<code>cores</code>	How many cores to use for moments computation, set at 10 by default. Only needed if <code>get.moments = "theoretical"</code>
<code>debug</code>	Whether to return the mean, standard deviation, skewness and kurtosis of the statistics
<code>verbose</code>	Whether to display information about the function actions

Details

For categorical phenotypes, 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, either 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: permuted 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 bootstrap program 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 permuted values.

If `NullObject$pheno.type = "continuous"`, 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".

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

All missing genotypes are imputed by the mean genotype.

Value

A data frame containing for each genomic region:

<code>stat</code>	The observed statistics
<code>p.value</code>	The p-value of the test

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

<code>p.perm</code>	The p-value computed by permutations: number of times permuted is greater than observed statistics divided by the total number of permutations performed
<code>p.chi2</code>	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:

<code>nb.gep</code>	The number of times a permuted statistics is equal or greater than the observed statistics <code>stat</code>
---------------------	--

nb.eq	The number of times a permuted 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.theoretical](#), [SKAT.bootstrap](#), [SKAT.permutations](#)

Examples

```
#Example on simulated data from Ravages with
#One group of 50 controls and
#two groups of 25 cases, each one with a prevalence of 0.01
#with 50% of causal variants, 5 genomic regions are simulated
GRR.del <- GRR.matrix(GRR = "SKAT", genes.maf = Kryukov,
                      n.case.groups = 2, select.gene = "R1",
                      GRR.multiplicative.factor=2)

x.sim <- random.bed.matrix(genes.maf = Kryukov, size = c(50, 25, 25),
                          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)

#Null Model
x.sim.H0 <- NullObject.parameters(x.sim@ped$pheno, RVAT = "SKAT", pheno.type = "categorical")

#Run SKAT (here permutations as n<2000 and no covariates)
#Parameters for the sampling procedure: target = 5, max = 100
#Please increase the number of permutations for a more accurate estimation of the p-values
params.sampling = list(perm.target = 5, perm.max = 100)
SKAT(x.sim, x.sim.H0, params.sampling = params.sampling)

#Run SKAT with a random continuous phenotype
#Null Model
x.sim.H0.c <- NullObject.parameters(rnorm(100), RVAT = "SKAT", pheno.type = "continuous")
SKAT(x.sim, x.sim.H0.c)

#Example on 1000Genome data
```

```

#Import 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"
#with very few permutations
#Please increase the permutations for a more accurate estimation of the p-values
#Fit Null model with covariate sex
x1.H0.covar <- NullObject.parameters(x1@ped$pop, RVAT = "SKAT", pheno.type = "categorical",
                                   data = covar, formula = ~ sex)

#Run SKAT with the covariates: use bootstrap as n<2000
SKAT(x1, x1.H0.covar, params.sampling = params.sampling, get.moments = "bootstrap")

#Run SKAT using theoretical moments (discourage here as n<2000) and 1 core
SKAT(x1, x1.H0.covar, get.moments = "theoretical", cores = 1)

```

SKAT.bootstrap

Multi group SKAT test using bootstrap sampling

Description

Performs 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

<code>x</code>	A bed.matrix
<code>NullObject</code>	A list returned from <code>NullObject.parameters</code>
<code>genomic.region</code>	A factor defining the genomic region of each variant
<code>weights</code>	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
<code>maf.threshold</code>	The MAF above which variants are removed (default is to keep all variants)
<code>perm.target</code>	The number of times to exceed the observed statistics. If not reached, <code>perm.max</code> permutations will be used
<code>perm.max</code>	The maximum number of permutations to perform to estimate the p-value, will be used if <code>perm.target</code> is not reached
<code>debug</code>	Whether to print details about the permutations (mean, standard deviation, skewness, kurtosis), FALSE by default
<code>estimation.pvalue</code>	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: permuted statistics are computed and each one is compared to the observed statistics. The bootstrap program 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.

All missing genotypes are imputed by the mean genotype.

Value

A data frame containing for each genomic:

<code>stat</code>	The observed statistics
<code>p.value</code>	<code>p.perm</code> if <code>perm.target</code> is reached, <code>p.chi2</code> if <code>perm.max</code> is reached.
<code>p.perm</code>	The p-value computed by permutations: number of times permuted is greater than observed statistics divided by the total number of permutations performed
<code>p.chi2</code>	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:

<code>nb.gep</code>	The number of times a permuted statistics is equal or greater than the observed statistics <code>stat</code>
<code>nb.eq</code>	The number of times a permuted statistics is equal to the observed statistics <code>stat</code>
<code>nb.perms</code>	The total number of simulations performed
<code>mean</code>	The mean of the permuted statistics
<code>sigma</code>	The standard deviation of the permuted statistics
<code>skewness</code>	The skewness of the permuted statistics
<code>kurtosis</code>	The kurtosis of the permuted 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](#)

Examples

```
#Import 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 = "categorical")

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

SKAT.continuous	<i>Multi group SKAT test using Liu et al. approximation</i>
-----------------	---

Description

Performs 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", 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

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](#)

Examples

```
#Import 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.permutations	<i>Multi group SKAT test using bootstrap sampling</i>
-------------------	---

Description

Performs 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: permuted statistics are computed and each one is compared to the observed statistics. The bootstrap program 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.

All missing genotypes are imputed by the mean genotype.

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 permuted 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 permuted statistics is equal or greater than the observed statistics stat
nb.eq	The number of times a permuted statistics is equal to the observed statistics stat
nb.perms	The total number of simulations performed
mean	The mean of the permuted statistics
sigma	The standard deviation of the permuted statistics
skewness	The skewness of the permuted statistics
kurtosis	The kurtosis of the permuted 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](#)

Examples

```
#Import 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 = "categorical")
SKAT.permutations(x1, x1.H0, perm.target = 10, perm.max=100)

```

SKAT.theoretical

Multi group SKAT test using Liu et al. approximation

Description

Performs 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

<code>x</code>	A bed.matrix
<code>NullObject</code>	A list returned from <code>NullObject.parameters</code>
<code>genomic.region</code>	A factor defining the genomic region of each variant
<code>weights</code>	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
<code>maf.threshold</code>	The MAF above which variants are removed (default is to keep all variants)
<code>estimation.pvalue</code>	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.

All missing genotypes are imputed by the mean genotype.

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

Examples

```
#Import 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 = "categorical")

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

WSS

WSS genetic score

Description

Calculates the WSS genetic score

Usage

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

Arguments

`x` A bed.matrix
`genomic.region` A factor 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

```
#Import 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

bed.matrix.split.genomic.region, [2](#)
burden, [3](#), [21](#)
burden.continuous, [4](#), [5](#)
burden.mlogit, [4](#), [7](#), [9](#), [10](#), [40](#)
burden.weighted.matrix, [4](#), [6](#), [8](#), [9](#), [10](#), [40](#)

CAST, [4](#), [6](#), [8](#), [9](#), [10](#), [40](#)

filter.rare.variants, [11](#)

genes.b37, [27](#)
genes.b37 (genes.positions), [12](#)
genes.b38, [27](#)
genes.b38 (genes.positions), [12](#)
genes.positions, [12](#)
genotypic.freq, [13](#)
GnomADgenes, [14](#), [14](#), [16](#), [23](#)
GRR.matrix, [14](#), [15](#), [23](#)

Jaccard, [16](#)

Kryukov, [14](#), [16](#), [17](#), [23](#)

LCT.hap (LCT.haplotypes), [18](#)
LCT.haplotypes, [18](#), [19](#)
LCT.matrix, [18](#), [19](#)
LCT.sample (LCT.haplotypes), [18](#)
LCT.snps (LCT.haplotypes), [18](#)

NullObject.parameters, [4](#), [8](#), [20](#), [30](#), [33](#), [35](#),
[37](#), [39](#)

random.bed.matrix, [14](#), [16](#), [22](#)
rbm.haplos.freqs, [24](#)
rbm.haplos.thresholds, [25](#)

set.genomic.region, [13](#), [26](#)
SKAT, [21](#), [28](#), [33](#), [35](#), [37](#)
SKAT.bootstrap, [30](#), [31](#)
SKAT.continuous, [34](#)
SKAT.permutations, [30](#), [36](#)

SKAT.theoretical, [30](#), [38](#)

WSS, [4](#), [6](#), [8–10](#), [40](#)