Package 'Ravages'

March 28, 2023

Type Package

Title Rare Variant Analysis and Genetic Simulations

Version 1.1.3

Date 2023-03-28

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 (Bocher et al. 2021 <doi:10.1038/s41431-020-00792-8>) in the whole genome; 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, parallel, bedr, curl, data.table

NeedsCompilation yes

Suggests knitr, rmarkdown

VignetteBuilder knitr

LazyLoad yes

LazyData yes

LazyDataCompression xz

Repository CRAN

Date/Publication 2023-03-28 13:00:02 UTC

R topics documented:

adjustedCADD.annotation

SNVs and Indels annotation with adjusted CADD scores

Description

Annotate SNVs and Indels with the adjusted CADD scores (CADD PHRED scores for coding, regulatory and intergenic regions)

Usage

```
adjustedCADD.annotation(x, SNVs.scores = NULL, indels.scores = NULL,
cores = 10, verbose = T, path.data)
```

Arguments

x	A bed.matrix annotated with CADD regions using set.CADDregions
SNVs.scores	A dataframe containing the ADJUSTED CADD scores of the SNVs (Optional, useful to gain in computation time if the adjusted CADD scores of variants in the study are available)
indels.scores	A dataframe containing the CADD PHREDv1.4 scores of the indels - Compulsory if indels are present in x
cores	How many cores to use, set at 10 by default
verbose	Whether to display information about the function actions
path.data	The repository where data for RAVA-FIRST are or will be downloaded from https://lysine.univ-brest.fr/RAVA-FIRST/

Details

This function calls adjustedCADD.annotation.SNVs and adjustedCADD.annotation.indels. See the help of those two functions for more details.

Value

The bed matrix x with adjusted CADD scores in adjCADD.

Source

https://lysine.univ-brest.fr/RAVA-FIRST/

See Also

adjustedCADD.annotation.SNVs, adjustedCADD.annotation.indels, RAVA.FIRST, filter.adjustedCADD

Examples

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

#Annotate variants with adjusted CADD score
#x <- adjustedCADD.annotation(x)
```

---

adjustedCADD.annotation.indels

Indels annotation with adjusted CADD scores

Description

Annotate Indels with the adjusted CADD scores (CADD PHRED scores for coding, regulatory and intergenic regions)

Usage

```
adjustedCADD.annotation.indels(x, variant.scores = NULL,
       cores = 10, verbose = T, path.data)
```

Arguments

x	A bed.matrix annotated with CADD regions using `set.CADDregions`
variant.scores	A dataframe containing the CADD PHREDv1.4 scores of the indels
cores	How many cores to use, set at 10 by default
verbose	Whether to display information about the function actions
path.data	The repository where data for RAVA-FIRST are or will be downloaded from https://lysine.univ-brest.fr/RAVA-FIRST/

Details

Indels are directly annotated with the adjusted CADD scores in the function using the file "AdjustedCADD_v1.4_202204_indels.tsv.gz" downloaded from https://lysine.univ-brest.fr/RAVA-FIRST/ in the repository of the package Ravages.

The adjusted CADD scores in "AdjustedCADD_v1.4_202204_indels.tsv.gz" have been computed using a set of 48M indels already annotated in the CADD website. If indels not present in this set are to be annotated, they will be given the same adjusted score as the indel with the nearest PHRED score v1.4 provided in `variant.scores` which should contain the chromosome ('chr'), position ('pos'), reference allele ('A1'), alternative allele ('A2') and PHRED CADD scores v1.4 ('PHRED_1.4').

Those adjusted scores are used in the `RAVA.FIRST()` pipeline to filter rare variants.

As this function can take time when a large number of SNVs are present, it is recommended to use this function chromosome by chromosome for large datasets or to fitler the bed matrix before the annotation.

Value

The bed matrix x with adjusted CADD scores in `adjCADD`.

Source

https://lysine.univ-brest.fr/RAVA-FIRST/

See Also

`adjustedCADD.annotation`, `adjustedCADD.annotation.SNVs`, `RAVA.FIRST`, `filter.adjustedCADD`

`adjustedCADD.annotation.SNVs`

SNVs annotation with adjusted CADD scores

Description

Annotate SNVs with the adjusted CADD scores (CADD PHRED scores for coding, regulatory and intergenic regions)

Usage

```
adjustedCADD.annotation.SNVs(x, variant.scores = NULL,
     cores = 10, verbose = T, path.data)
```

Arguments

`x`	A bed.matrix annotated with CADD regions using `set.CADDregions`
`variant.scores`	A dataframe containing the ADJUSTED CADD scores of the SNVs (Optional, useful to gain in computation time if the adjusted CADD scores of variants in the study are available)
`cores`	How many cores to use, set at 10 by default
`verbose`	Whether to display information about the function actions
`path.data`	The repository where data for RAVA-FIRST are or will be downloaded from https://lysine.univ-brest.fr/RAVA-FIRST/

Details

SNVs are directly annotated with the adjusted CADD scores in the function using the file "AdjustedCADD_v1.4_202108.tsv.gz" downloaded from https://lysine.univ-brest.fr/RAVA-FIRST/ in the repository of the package Ravages or the scores of variants can be provided to `variant.scores` to gain in computation time (this file should contain 5 columns: the chromosome ('chr'), position ('pos'), reference allele ('A1'), alternative allele ('A2') and adjusted CADD scores ('adjCADD').

Those adjusted scores are used in the `RAVA.FIRST()` pipeline to filter rare variants.

As this function can take time when a large number of SNVs are present, it is recommended to use this function chromosome by chromosome for large datasets or to fitler the bed matrix before the annotation.

Value

The bed matrix x with adjusted CADD scores in `adjCADD`.

Source

https://lysine.univ-brest.fr/RAVA-FIRST/

See Also

`adjustedCADD.annotation`, `adjustedCADD.annotation.indels`, `RAVA.FIRST`, `filter.adjustedCADD`

Examples

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

#Annotate variants with adjusted CADD score
#x <- adjustedCADD.annotation.SNVs(x)
```

---

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=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.effect.size = 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.effect.size`
: TRUE/FALSE: whether to return effect sizes of the tested `genomic.region` (OR for categorical phenotypes, betas for continuous phenotypes)

`alpha`
: The alpha threshold to use for the OR confidence interval

`cores`
: How many cores to use, set at 10 by default.

`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/beta`
: The OR/beta value(s) associated to the regression. For categorical phenotypes, 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/beta

`l.upper`
: The upper bound of the confidence interval of each OR/beta

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.effect.size=TRUE, cores = 1)
```

---

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, get.effect.size = F,
                  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 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.effect.size	TRUE/FALSE: whether to return the beta value
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 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
beta	The beta coefficient associated to the tested genomic region
l.lower	The lower bound of the confidence interval of beta
l.upper	The upper bound of the confidence interval of beta

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")
#Get the beta value
burden.continuous(x1, NullObject = x1.H0, burden = "WSS",
                  get.effect.size = TRUE, cores = 1)
```

`burden.continuous.subscores`

Linear regression on a multiple genetic scores within a genomic region

---

Description

Performs burden tests with subscores in the regression on continuous phenotypes

Usage

```
burden.continuous.subscores(x, NullObject, genomic.region = x@snps$genomic.region,
                            SubRegion = x@snps$SubRegion, burden.function = WSS,
                            maf.threshold = 0.5, get.effect.size = 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 containing the genomic region of each SNP, `x@snps$genomic.region` by default, for example the CADD regions
`SubRegion`	A vector containing subregions within each `genomic.region`, `x@snps$SubRegion` by default, for example genomic categories
`burden.function`	A function to compute the genetic score, `WSS` by default.
`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.effect.size`	TRUE/FALSE: whether to return effect sizes of the tested `genomic.region` (OR for categorical phenotypes, betas for continuous phenotypes)
`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 = "categorical"`

Details

This function will return results from the regression of the phenotype on the genetic score(s) for each genomic region. Within each genomic region, a subscore will be computed for each SubRegion and one test will be performed for each genomic.region.

Value

A dataframe with one row per genomic region and 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.effect.size=TRUE`, a list is returned with the previous dataframe in `$Asso` and with `effect`, a list containing matrices with three columns:

`beta`	The beta value(s) associated to the subscores in the regression
`l.lower`	The lower bound of the confidence interval of each beta
`l.upper`	The upper bound of the confidence interval of each beta

See Also

`NullObject.parameters`, `burden.subscores`, `CAST`, `WSS`

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 CADD regions and genomic categories
#x <- set.CADDregions(x)

#Filter of rare variants: only non-monomorphic variants with
#a MAF lower than 2.5%
#and with a adjusted CADD score greater than the median
#x1 <- filter.adjustedCADD(x, filter = "whole", maf.threshold = 0.025)

#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
#res.subscores <-burden.continuous.subscores(x1, NullObject = x1.H0.covar,
#                                            burden = WSS, get.effect.size=TRUE, cores = 1)
#res.subscores$Asso # p-values
#res.subscores$effect #beta values
```

`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.effect.size = 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 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.effect.size`	TRUE/FALSE: whether to return OR values
`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.effect.size=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 associationtesting 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 200 SNP
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.025, min.nb.snps = 200)

#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.effect.size=TRUE, cores = 1)
```

`burden.mlogit.subscores` *Logistic or multinomial regression on a multiple genetic scores within a genomic region*

Description

Performs burden tests with subscores in the regression on categorical phenotypes

Usage

```
burden.mlogit.subscores(x, NullObject, genomic.region = x@snps$genomic.region,
                        SubRegion = x@snps$SubRegion, burden.function = WSS,
                        maf.threshold = 0.5, get.effect.size = 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 containing the genomic region of each SNP, `x@snps$genomic.region` by default, for example the CADD regions
`SubRegion`	A vector containing subregions within each `genomic.region`, `x@snps$SubRegion` by default, for example genomic categories
`burden.function`	A function to compute the genetic score, `WSS` by default.
`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.effect.size`	TRUE/FALSE: whether to return effect sizes of the tested `genomic.region` (OR for categorical phenotypes, betas for continuous phenotypes)
`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 = "categorical"`

Details

This function will return results from the regression of the phenotype on the genetic score(s) for each genomic region. Within each genomic region, a subscore will be computed for each SubRegion and one test will be performed 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`.

Value

A dataframe with one row per genomic region and 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.effect.size=TRUE`, a list is returned with the previous dataframe in `$Asso` and with `effect`, a list containing matrices with three columns:

`OR`	The OR value(s) associated to the subscores in 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

See Also

`NullObject.parameters`, `burden.subscores`, `CAST`, `WSS`

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 CADD regions and genomic categories
#x <- set.CADDregions(x)

#Filter of rare variants: only non-monomorphic variants with
#a MAF lower than 2.5%
#and with a adjusted CADD score greater than the median
#x1 <- filter.adjustedCADD(x, filter = "whole", maf.threshold = 0.025)
```

```
#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
#res.subscores <- burden.subscores(x1, NullObject = x1.H0, burden = WSS,
#                                  get.effect.size=TRUE, cores = 1)
#res.subscores$Asso # p-values
#res.subscores$effect #OR values
```

`burden.subscores` *Linear, logistic or multinomial regression on a multiple genetic scores within a genomic region*

Description

Performs burden tests with subscores in the regression on categorical or continuous phenotypes

Usage

```
burden.subscores(x, NullObject, genomic.region = x@snps$genomic.region,
                 SubRegion = x@snps$SubRegion, burden.function = WSS,
                 maf.threshold = 0.5, get.effect.size = FALSE,
                 alpha = 0.05, cores = 10, verbose = TRUE)
```

Arguments

`x`	A bed matrix
`NullObject`	A list returned from `NullObject.parameters`
`genomic.region`	A factor containing the genomic region of each SNP, `x@snps$genomic.region` by default, for example the CADD regions
`SubRegion`	A vector containing subregions within each `genomic.region`, `x@snps$SubRegion` by default, for example genomic categories
`burden.function`	A function to compute the genetic score, `WSS` by default.
`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.effect.size`	TRUE/FALSE: whether to return effect sizes of the tested `genomic.region` (OR for categorical phenotypes, betas for continuous phenotypes)
`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 = "categorical"`
`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(s) for each genomic region. Within each genomic region, a subscore will be computed for each SubRegion and one test will be performed for each genomic.region.

When used after `set.CADDregions`, it will perform a test by CADD region with one subscore by genomic category (coding, regulatory, intergenic) as in the `RAVA.FIRST()` strategy.

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`.

Value

A dataframe with one row per genomic region and 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.effect.size=TRUE`, a list is returned with the previous dataframe in `$Asso` and with `effect`, a list containing matrices with three columns:

`OR/beta`	The OR/beta value(s) associated to the subscores in the regression. For categorical phenotypes, 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/beta
`l.upper`	The upper bound of the confidence interval of each OR/beta

See Also

`RAVA.FIRST`, `NullObject.parameters`, `burden.continuous.subscores`, `burden.mlogit.subscores`, `CAST`, `WSS`

Examples

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

#Group variants within known genes and
#Within coding and regulatory regions
x <- set.genomic.region.subregion(x, regions = genes.b37,
                                  subregions = subregions.LCT)

#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)

#Keep only variants with a MAF lower than 1%
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.01)

#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 functionally-informed burden test WSS in LCT
burden.subscores(select.snps(x1, genomic.region == "LCT"),
                 NullObject = x1.H0, burden.function = WSS,
                 get.effect.size=FALSE, cores = 1)


####Using the RAVA-FIRST approach with CDD regions
#Group variants within CADD regions and genomic categories
#x <- set.CADDregions(x)

#Filter of rare variants: only non-monomorphic variants with
#a MAF lower than 2.5%
#and with a adjusted CADD score greater than the median
#x1 <- filter.adjustedCADD(x, filter = "whole", maf.threshold = 0.025)

#run functionally-informed burden test WSS
#burden.subscores(x1, NullObject = x1.H0, burden.function = WSS,
#                 get.effect.size=FALSE, 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

`x`	A bed.matrix
`weights`	A vector containing the weight of each variant
`genomic.region`	A factorcontaining 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*. This function returns a weighted score of rare alleles in the genomic region: if the reference allele is rare, it will be counted in the score instead of the atlernative allele.

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,
     flip.rare.alleles = T)
```

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
`flip.rare.alleles`	Whether to flip the A1/A2 alleles if the A1 allele is rare, set at T by default

Details

By default, CAST counts if an individual carries at least one rare allele in the genomic region. If `flip.rare.alleles = F` and the reference allele A1 is rare, the alles A1 and A2 won't be flipped and CAST will count the number of alternative alleles A2.

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.adjustedCADD *Variant filtering based on frequency and median adjusted CADD by CADD regions*

Description

Filter rare variants based on a MAF threshold, a given number of SNP or a given cumulative MAF per genomic region and the median of adjusted CADD score for each CADD region

Usage

```
filter.adjustedCADD(x, SNVs.scores = NULL, indels.scores = NULL,
                    ref.level = NULL,
                    filter=c("whole", "controls", "any"),
                    maf.threshold=0.01, min.nb.snps = 2,
                    min.cumulative.maf = NULL,
                    group = NULL, cores = 10, path.data, verbose = T)
```

Arguments

x	A bed.matrix annotated with CADD regions using `set.CADDregions`
SNVs.scores	A dataframe containing the ADJUSTED CADD scores of the SNVs (Optional, useful to gain in computation time if the adjusted CADD scores of variants in the study are available)
indels.scores	A dataframe containing the CADD PHREDv1.4 scores of the indels - Compulsory if indels are present in `x`
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 variants needed to keep a CADD region, set at 2 by default
min.cumulative.maf	The minimum cumulative maf of variants needed to keep a CADD region
group	A factor indicating the group of each individual, only needed if `filter = "controls"` or `filter = "any"`. If missing, `x@ped$pheno` is taken
cores	How many cores to use, set at 10 by default
path.data	The repository where data for RAVA-FIRST are or will be downloaded from https://lysine.univ-brest.fr/RAVA-FIRST/
verbose	Whether to display information about the function actions

Details

Variants are directly annotated with the adjusted CADD scores in the function using the file "AdjustedCADD_v1.4_202108.tsv.gz" downloaded from https://lysine.univ-brest.fr/RAVA-FIRST/ in the repository of the package Ravages or the scores of variants can be provided to `variant.scores` to gain in computation time (this file should contain 5 columns: the chromosome ('chr'), position ('pos'), reference allele ('A1'), alternative allele ('A2') and adjusted CADD scores ('adjCADD'). As CADD scores are only available for SNVs, only those ones will be kept in the analysis.

If a column 'adjCADD' is already present in `x@snps`, no annotation will be performed and filtering will be directly on this column.

To use this function, a factor 'genomic.region' corresponding to the CADD regions and a vector 'adjCADD.Median' should be present in the slot `x@snps`. To obtain those two, use the function `set.CADDregions`.

Only variants with an adjusted CADD score upper than the median value are kept in the analysis. It is the filtering strategy applied in the `RAVA.FIRST()` pipeline.

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.

It is recommended to use this function chromosome by chromosome for large datasets.

Value

A bed.matrix with filtered variants

Source

https://lysine.univ-brest.fr/RAVA-FIRST/

See Also

`RAVA.FIRST`, `set.CADDregions`, `burden.subscores`, `filter.rare.variants`

Examples

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

#Group variants within CADD regions and genomic categories
#x <- set.CADDregions(x)

#Annotate variants with adjusted CADD score
#and filter on frequency and median
#x.median <- filter.adjustedCADD(x, maf.threshold = 0.025,
#                                min.nb.snps = 2)
```

---

filter.rare.variants *Rare variants filtering*

Description

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

Usage

```
filter.rare.variants(x, ref.level = NULL, filter=c("whole", "controls", "any"),
                 maf.threshold=0.01, min.nb.snps = 2, min.cumulative.maf = NULL,
                     group = NULL, 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 variants needed to keep a genomic region, set at 2 by default
min.cumulative.maf	The minimum cumulative maf of variants needed to keep a genomic region
group	A factor indicating the group of each individual, only needed if filter = "controls" or filter = "any". If missing, x@ped$pheno is taken
genomic.region	An optional factor containing the genomic region of each variant, only needed if min.nb.snps or min.cumulative.maf is specified and if x@snps$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@snps$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@snps$genomic.region, useNA="ifany")

#Keep only variants with a MAF<2%
#and regions with a cumulative MAF>10%
```

```
filter.rare.variants(x, filter = "whole", maf.threshold = 0.02, min.nb.snps = 1,
                     min.cumulative.maf=0.2)
```

`genes.positions` *Genes positions*

Description

Positions of human genes in bed format (Start is 0-based and End is 1-based). 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 `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 (0-based)

`End` The end position of the gene (1-based)

`Name` The name of the gene

Source

The data were obtained from the Ensembl website.

References

RJ Kinsella et al, 2011, *Ensembl BioMarts: 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 *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,
               selected.controls = T)
```

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.
selected.controls	Whether controls are selected controls (by default) or controls from the general population

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.

If selected.controls = T, genotypic frequencies in the control group are computed from genotypic frequencies in the cases groups and the prevalence of the disease. If FALSE, genotypic frequencies in the control group are computed from allelic frequencies under Hardy-Weinberg equilibrium.

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, rbm.GRR, 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 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 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, 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 rbm.GRR.

Value

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

See Also

rbm.GRR, 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 <- rbm.GRR(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 (5008 haplotypes) of the 2004 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.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.

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")]
```

`multinomial.asso.freq` *Single variant association test with categorical phenotype*

Description

Performs an association test between categorical phenotypes and single variants

Usage

```
multinomial.asso.freq(x, pheno = x@ped$pheno, ref.level,
                      test = c("Genotypic", "Allelic"), get.effect.size = F,
                      min.maf.threshold = 0.05)
```

Arguments

`x`	A bed matrix, only needed if `burden="CAST"` or `burden="WSS"`
`pheno`	The phenotype of each individual: a factor if `pheno.type = "categorical"`, and a numeric vector if `pheno.type = "continuous"`
`ref.level`	The reference group of individuals for the estimation of the effect size, only needed if `get.effect.size = T`
`test`	Whether to perform the test on the three genotypes ("Genotypic") or on the two alleles ("Allelic")
`get.effect.size`	TRUE/FALSE: whether to return effect sizes of the variants (OR)
`min.maf.threshold`	MAF threshold used to define a frequent variant to apply single-variant test

Details

This association test is based on a chi-square with the following number of df: If `test = "Genotypic"`, (number of groups of individuals - 1)* 2 If `test = "Allelic"`, (number of groups of individuals - 1)

Value

A dataframe with one row per variant and three columns: the chromosome, position and p-value of each variant. If `get.effect.size = T`, a list with `Asso` containing the previous dataframe and `OR` containing the OR in each group for each variant.

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)

#Perform association test
x.freq.asso <- multinomial.asso.freq(x, test = "Genotypic",
                                     pheno = x@ped$pop)
```

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 = c("categorical", "continuous"),
                      ref.level, data, formula)
```

Arguments

`pheno`	The phenotype of each individual: a factor if `pheno.type = "categorical"`, 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: "categorical" for binary or multinomial traits, or "continuous"
`ref.level`	The reference group of individuals for the regression, only needed if `RVAT = "burden"` and `pheno.type = "categorical"`
`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 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.

- 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)
```

RAVA.FIRST *RAVA-FIRST: RAre Variant Association using Functionally-InfoRmed STeps*

Description

Analyse rare variants using the RAVA-FIRST approach based on CADD scores to group and filter rare variants

Usage

```
RAVA.FIRST(x, SNVs.scores = NULL, indels.scores = NULL, ref.level,
           filter=c("whole", "controls", "any"),
           maf.threshold=0.01, min.nb.snps = 2,
           min.cumulative.maf = NULL, group = NULL,
           cores = 10, burden = TRUE, H0.burden, burden.parameters,
           SKAT = TRUE, H0.SKAT, SKAT.parameters, verbose = TRUE, path.data)
```

Arguments

x	A bed.matrix
SNVs.scores	A dataframe containing the ADJUSTED CADD scores of the SNVs (Optional, useful to gain in computation time if the adjusted CADD scores of variants in the study are available)
indels.scores	A dataframe containing the CADD PHREDv1.4 scores of the indels - Compulsory if indels are present in x

ref.level	The level corresponding to the controls group, only needed if filter=="controls"
filter	On which group the MAF 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 variants needed to keep a CADD region, set at 2 by default
min.cumulative.maf	The minimum cumulative maf of variants needed to keep a CADD region
group	A factor indicating the group of each individual, only needed if filter = "controls" or filter = "any". If missing, x@ped$pheno is taken
cores	How many cores to use, set at 10 by default
burden	Whether to compute the burden test
H0.burden	A list returned from NullObject.parameters with RVAT="burden"
burden.parameters	A list containing the parameters to use by burden.subscores for the burden analysis ('burden.function' and 'get.effect.size')
SKAT	Whether to compute SKAT
H0.SKAT	A list returned from NullObject.parameters with RVAT="SKAT"
SKAT.parameters	A list containing the parameters to use by SKAT ('get.moments', 'estimation.pvalue', 'params.sampling', 'debug')
verbose	Whether to display information about the function actions
path.data	The repository where data for RAVA-FIRST are or will be downloaded from https://lysine.univ-brest.fr/RAVA-FIRST/

Details

Rare variants are analysed using the 'RAVA-FIRST' strategy composed of three steps: - Rare variants are grouped in 'CADD regions' defined from the CADD scores of variants observed in GnomAD. - Rare variant are selected within each CADD region based on an adjusted CADD score using a region-specific threshold corresponding to the median of scores observed in GnomAD in each region. - Burden analysis is performed by integrating sub-scores for the coding, regulatory and intergenic categories within each CADD region. For SKAT analysis, a test for each CADD region is performed.

RAVA.FIRST() is based on the functions set.CADDregions, filter.adjustedCADD, burden.subscores and SKAT. Please refer to these functions for more information. Especially, refer to the functions burden.subscores and SKAT to get more information about what is need in burden.parameters and SKAT.parameters.

It is recommended to use this function chromosome by chromosome for large datasets.

Value

A list containing the results for the burden analysis ('burden') and the results for the SKAT analysis ('SKAT'), along with information about CADD regions (positions, type of genomic categories overlapped by each region and median of adjusted CADD scores).

Source

https://lysine.univ-brest.fr/RAVA-FIRST/

See Also

set.CADDregions, filter.adjustedCADD, burden.subscores, SKAT

Examples

```
#Import 1000Genome data from region around LCT gene
#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)

#Remove indels from the bed matrix
#x <- select.snps(x, nchar(A1)==1 & nchar(A2)==1)

#Perform RAVA-FIRST with burden analysis
#H0.burden <- NullObject.parameters(pheno = x@ped$pop, ref.level = "CEU",
#                                   RVAT = "burden", pheno.type = "categorical")
#res.burden <- RAVA.FIRST(x, maf.threshold = 0.05,
#                         H0.burden = H0.burden, SKAT = F)
```

---

rbm.GRR *Simulation of genetic data using GRR values*

Description

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

Usage

```
rbm.GRR(genes.maf = Kryukov, size, prev, replicates,
        GRR.matrix.del, GRR.matrix.pro = NULL,
        p.causal = 0.5, p.protect = 0, same.variant = FALSE,
        genetic.model=c("general", "multiplicative", "dominant", "recessive"),
        select.gene, selected.controls = T, max.maf.causal = 0.01)
```

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.
selected.controls	Whether controls are selected controls (by default) or controls from the general population
max.maf.causal	Only variants with a MAF lower than this threshold can be sampled as causal variants.

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.

If selected.controls = T, genotypic frequencies in the control group are computed from genotypic frequencies in the cases groups and the prevalence of the disease. If FALSE, genotypic frequencies in the control group are computed from allelic frequencies under Hardy-Weinberg equilibrium.

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.

Only non-monomorphic variants are kept for the simulations.

Causal variants that have been sampled in each group of individuals are indicated in `x@ped$Causal`.

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`, `rbm.GRR.power`

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 <- rbm.GRR(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.GRR.power *Power of RVAT based on simulations and theoretical calculations (CAST) with GRR*

Description

Computes the power of the tests CAST, WSS and SKAT based on simulations with GRR and based on theoretical calculations for CAST

Usage

```
rbm.GRR.power(genes.maf = Kryukov, size = c(500, 500), prev = 0.01,
              GRR.matrix.del, GRR.matrix.pro = NULL,
              p.causal = 0.5, p.protect = 0, same.variant = FALSE,
          genetic.model=c("multiplicative", "general", "dominant", "recessive"),
              select.gene, alpha = 2.5e-6, selected.controls = TRUE,
              power.type = c("simulations", "theoretical"), verbose = TRUE,
              RVAT = c("CAST", "WSS", "SKAT"),
              SKAT.method = c("permutations", "theoretical"),
              max.maf.causal = 0.01, maf.filter = max.maf.causal,
              replicates = 1000, cores = 10)
```

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
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.
alpha	The significance level to compute the power
selected.controls	Whether controls are selected controls (by default) or controls from the general population
power.type	Whether to compute the power based on 'simulations' (by default) or 'theoretical' calculations (only for CAST)
verbose	Whether to print details about the running function
RVAT	On which RVAT among 'CAST', 'WSS' and 'SKAT' to compute power (only needed if power.type="simulations"
SKAT.method	Which method to use to compute SKAT ppower, i.e. permutations or theoretical moments (cf SKAT documentation)
max.maf.causal	The maf threshold to consider a causal variant (set at 0.01 by default)
maf.filter	The MAF filter to apply after the simulations to select rare variants to keep for RVAT power analysis. By default corresponds to max.maf.causal
replicates	On how many replicates the power should be computed
cores	How many cores to use for moments computation, set at 10 by default

Details

Simulations are performed in the same was as in rbm.GRR. Please refer to the documentation of this function.

Theoretical power is only available for CAST for which a non-central Chi-squared is used.

Variants are filtered after the simulations to keep only the rare ones, defined by maf.filter. By defaut, it corresponds to max.maf.causal is used. To disable this filter, set maf.filter at 0.5.

Value

A single value giving the power of CAST if `power.type="theoretical"` or the power of RVAT if `power.type="simulations"`.

See Also

`GRR.matrix`, `Kryukov`, `GnomADgenes`, `rbm.GRR`

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.
rbm.GRR.power(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",
              power.type="theoretical", cores = 1, alpha = c(0.001,2.5e-6))
```

---

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.power *Power of RVAT based on simulations with haplotypes*

Description

Computes the power of the tests CAST, WSS and SKAT based on simulations with haplotypes

Usage

```
rbm.haplos.power(haplos, freqs, weights = "SKAT",
            max.maf.causal = 0.01, maf.filter = max.maf.causal, p.causal = 0.5,
                 p.protect = 0, h2 = c(0.01, 0.01), prev = c(1, 0.01),
                 normal.approx = TRUE, size = c(500, 500), verbose = TRUE,
                 alpha = 2.5e-6, RVAT = c("CAST", "WSS", "SKAT"),
                 SKAT.method = c("permutations", "theoretical"),
                 simus.haplos = c("freqs", "liability"),
                 replicates = 1000, rep.by.causal = 50, cores = 10)
```

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, only needed if simus.haplos = "freqs"

weights	How to weight rare variants (if "constant", all variants have the same weight, if "SKAT", the rarest variants have the highest weights: weights = -0.4*log10(MAF))
max.maf.causal	The maf threshold to consider a rare variant (set at 0.01 by default). Only variants with a MAF upper than this threshold will be kept to compute RVAT power. If simus.haplos="liability", variants with a MAF upper this threshold will have a weight of 0
maf.filter	The MAF filter to apply after the simulations to select rare variants to keep for RVAT power analysis. By default corresponds to max.maf.causal
p.causal	The percentage of causal variants, only needed if simus.haplos = "liability"
p.protect	The proportion of protective variants among causal variants, only needed if simus.haplos = "liability"
h2	The variance explained by the gene, only needed if simus.haplos = "liability"
prev	A vector with the prevalence in each group of individuals, only needed if simus.haplos = "liability"
normal.approx	TRUE/FALSE: whether to use the normal approximation to compute thresholds. Set at TRUE by default, only needed if simus.haplos = "liability"
size	The sizes of each group of individuals
verbose	Whether to display information about the function actions
alpha	The significance level to compute the power
RVAT	On which RVAT among 'CAST', 'WSS' and 'SKAT' to compute power (only needed if power.type="simulations"
SKAT.method	Which method to use to compute SKAT ppower, i.e. permutations or theoretical moments (cf SKAT documentation)
simus.haplos	Which method to simulate the data, if simus.haplos="freqs", rbm.haplos.freqs() is used, otherwise rbm.haplos.thresholds() is used.
replicates	The number of simulations to perform to estimate the power
rep.by.causal	The number of time causal variants will be sampled
cores	How many cores to use for moments computation, set at 10 by default

Details

Simulations are perfromed accordingly to rbm.haplos.thresholds() or rbm.haplos.freqs(). Please refer to the corresponding manuals for more details on the simulation procedures. Variants are filtered after the simulations to keep only the rare ones, defined by maf.filter. By defaut, it corresponds to max.maf.causal is used. To disable this filter, set maf.filter at 0.5.

Value

Power values of RVAT

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
rbm.haplos.power(haplos=LCT.gene.hap, freqs=LCT.freqs, size=rep(50,5),
                 replicates=5, rep.by.causal = 5,  RVAT = "CAST",
                 alpha = c(0.001,2.5e-6), cores = 1)
```

`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 = c("SKAT", "constant"),
                      max.maf.causal = 0.01, p.causal = 0.5, 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`	How to weight rare variants (if "constant", all variants have the same weight, if "SKAT", the rarest variants have the highest weights as in the SKAT paper: weights = -0.4*log10(MAF))
`max.maf.causal`	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
`p.causal`	The proportion 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 max.maf.causal will have a weight of 0. Causal variants are sampled among variants having weights greater than 0. Causal variants in each group of individuals are indicated in x@ped$Causal.

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, max.maf.causal = 0.01, p.causal=0.3,
                           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.CADDregions` *Variants annotation based on 'CADD regions' and genomic categories*

Description

Attributes CADD regions and genomic categories to variants based on their positions

Usage

```
set.CADDregions(x, verbose = T, path.data)
```

Arguments

`x`	A bed.matrix
`verbose`	Whether to display information about the function actions
`path.data`	The repository where data for RAVA-FIRST are or will be downloaded from https://lysine.univ-brest.fr/RAVA-FIRST/

Details

To attribute variants to CADD regions and genomic categories, the files "CADDRegions.2021.hg19.bed.gz" and "FunctionalAreas.hg19.bed.gz" will be downloaded from https://lysine.univ-brest.fr/RAVA-FIRST/ in the repository of the package Ravages. CADD regions are non-overlapping regions that have been defined in the whole genome to perform rare variant association tests in the `RAVA.FIRST()` pipeline. It is recommended to use this function chromosome by chromosome for large datasets for time and memory managment.

Value

The same bed matrix as x with three additional columns :

`genomic.region`	The CADD region of each variant
`SubRegion`	The genomic category of each variant among 'Coding', 'Regulatory' or 'Intergenic'
`adjCADD.Median`	The median of adjusted CADD of variants observed at least to times in GnomAD genomes r2.0.1

Source

https://lysine.univ-brest.fr/RAVA-FIRST/

See Also

RAVA.FIRST, filter.adjustedCADD, burden.subscores

Examples

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

#Group variants within CADD regions and genomic categories
#x <- set.CADDregions(x)
#table(x@snps$genomic.region) #CADD regions
#table(x@snps$SubRegion) #Genomic categories
```

---

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

Description

Attributes regions to variants based on given region positions

Usage

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

Arguments

x	A bed.matrix
regions	A dataframe in bed format (start is 0-based and end is 1-based) containing the fields : Chr (the chromosome of the gene), Start (the start position of the gene, 0-based), End (the end position of the gene, 1-based), and Name (the name of the gene - a factor),
flank.width	An integer: width of the flanking regions in base pairs downstream and upstream the regions.
split	Whether to split variants attributed to multiple regions by duplicating this variants, set at TRUE by default

Details

Warnings: `regions$Name` should be a factor containing UNIQUE 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.

If a variant is attributed to multiple genomic regions, it will be duplicated in the bed matrix with one row per genomic region if `split = TRUE`. Variants will have new IDs being CHR:POS:A1:A2:genomic.region.

Value

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

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)
```

---

set.genomic.region.subregion

Variants annotation based on regions and subregions positions

---

Description

Attributes regions and subregions to variants based on given positions

Usage

```
set.genomic.region.subregion(x, regions, subregions, split = TRUE)
```

Arguments

x	A bed.matrix
regions	A dataframe in bed format (start is 0-based and end is 1-based) containing the regions with the fields : Chr (the chromosome of the gene), Start (the start position of the gene, 0-based), End (the end position of the gene, 1-based), and Name (the name of the gene - a factor),
subregions	A dataframe containing the subregions in the same format as regions
split	Whether to split variants attributed to multiple regions by duplicating this variants, set at TRUE by default

Details

Warnings: regions$Name and subregions$Name should be factors containing UNIQUE names of the regions, ORDERED in the genome order.

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

If a variant is attributed to multiple genomic regions, it will be duplicated in the bed matrix with one row per genomic region if split = TRUE.

This function can be applied before using burden.subscores to perform a functionally-informed burden tests with sub-scores for each SubRegion within each genomic.region.

Value

The same bed matrix as x with two additional columns: x@snps$genomic.region containing the annotation of the regions and x@snps$SubRegion containing the annotation of the subregions.

See Also

set.genomic.region, burden.subscores

Examples

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

#Group variants within known genes and
#Within coding and regulatory regions
x <- set.genomic.region.subregion(x,
 regions = genes.b37, subregions = subregions.LCT)
```

SKAT *SKAT test*

Description

Peforms 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

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 categorical 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. Only needed if get.moments = "theoretical"
debug	Whether to return the mean, standard deviation, skewness and kurtosis of the statistics
verbose	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, 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 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:

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.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 <- rbm.GRR(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, cores = 1)



#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 boostrap 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

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.

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 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

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` *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", 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, cores = 1)
```

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.

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 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

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

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.

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)
```

`subregions.LCT` *Exemple of functional categories*

Description

Example of arbitrary functional categories (coding or regulatory) in the LCT locus (bed format, GRCH37). "Coding" corresponds to coding parts of the exons and "Regulatory" corresponds to everything that falls outside these coding regions.

Data contain the `Chr`, the `Start` position, the `End` position and the `Name` of all functional regions in the LCT locus.

Format

The data contain one dataframe with four columns:

`Chr` The chromosome of the gene

`Start` The start position of the functional region (0-based)

`End` The end position of the functional region (1-based)

`Name` The name of the gene

See Also

`set.genomic.region.subregion`, `burden.subscores`

`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 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