Package 'VariantFiltering'

April 10, 2015

Type Package

Title Filtering of coding and non-coding genetic variants

Description Filter genetic variants using different criteria such as inheritance model, amino acid change consequence, minimum allele frequencies across human populations, splice site strength, conservation, etc.

Version 1.2.14

Author Dei M. Elurbe, Robert Castelo

Maintainer Robert Castelo <robert.castelo@upf.edu>

Depends R (>= 3.0.0), methods, BiocGenerics (>= 0.11.3)

Imports DBI, RSQLite (>= 1.0.0), Biobase, S4Vectors, IRanges (>= 1.99.17), AnnotationDbi, BiocParallel, Biostrings (>= 2.33.11), GenomeInfoDb (>= 1.2.2), GenomicRanges (>= 1.17.19), GenomicFeatures, Rsamtools (>= 1.17.8), BSgenome, BSgenome.Hsapiens.UCSC.hg19, VariantAnnotation (>= 1.12.2), shiny

Suggests BiocStyle, org.Hs.eg.db, TxDb.Hsapiens.UCSC.hg19.knownGene, SNPlocs.Hsapiens.dbSNP.20120608, MafDb.ALL.wgs.phase1.release.v3.20101123, MafDb.ESP6500SI.V2.SSA137.dbSNP138, phastCons100way.UCSC.hg19, PolyPhen.Hsapiens.dbSNP131, SIFT.Hsapiens.dbSNP137

LinkingTo S4Vectors, IRanges, XVector, Biostrings

License Artistic-2.0

LazyData yes

URL https://github.com/rcastelo/VariantFiltering/issues

biocViews Genetics, Homo_sapiens, Annotation, SNP, Sequencing, HighThroughputSequencing

R topics documented:

`VariantFiltering-package`

Filtering of coding and non-coding genetic variants

Description

The `VariantFiltering` package filters coding and non-coding genetic variants using different criteria such as an inheritance model (autosomal recessive -both, homozygous and heterozygous-, autosomal dominant, X-linked and *de novo*), amino acid change consequence, minimum allele frequencies, cryptic splice site potential, conservation, etc.

Functions

- `autosomalRecessiveHomozygous` identify homozygous variants in the affected individual(s) while the unaffected ones present these same variants but in heterozygous state. Autosomal recessive inheritance pattern.
- `autosomalRecessiveHeterozygous` identify variants grouped by genes with two (or more) heterogeneous alleles (at least one on each allele, i.e. coming from each parent). Autosomal recessive inheritance pattern.
- `autosomalDominant` identify variants present in all the affected individual(s) discarding the ones that also occur in at least one of the unaffected subjects. Autosomal dominant inheritance pattern.
- `xLinked` identify variants that appear only in the X chromosome of the unaffected females as heterozygous, don't appear in the unaffected males analyzed and finally are present (as homozygous) in the affected male(s). X-linked inheritance pattern.
- `deNovo` identify variants in the affected individual that have not been inherited.
- `allInheritanceModels` annotates all the variants present in the VCF file allowing to select the filtering according to a certain inheritance model interactively through a shiny app.
- `unrelatedIndividuals`

Author(s)

Dei M. Elurbe and Robert Castelo.

Maintainer: Robert Castelo <robert.castelo@upf.edu>

References

Elurbe D.M., Mila, M., Castelo, R. VariantFiltering: filtering of coding and non-coding genetic variants, in preparation.

---

allInheritanceModels *Analysis designed to be applied over a group of related individuals*

Description

This method filters variants from a group of related individuals annotating compatible inheritance models of segregation.

Usage

```
## S4 method for signature VariantFilteringParam
allInheritanceModels(param,
                                                    BPPARAM=bpparam())
```

Arguments

`param`	An VariantFilteringParam object containing a VCF file.
`BPPARAM`	An object of class 'BiocParallelParam' specifiying parameters related to the parallel execution of some of the tasks and calculations within this function. See function `bpparam()` from the `BiocParallel` package.

Details

This function requires as an input a VariantFilteringParam class object which contains the vcf file ready for the analysis.

Value

An object of class `VariantFilteringResults` including functional annotations on all variants.

Author(s)

Dei M. Elurbe and R. Castelo

References

Elurbe D.M., Mila, M., Castelo, R. The VariantFiltering package, in preparation.

See Also

autosomalRecessiveHomozygous autosomalRecessiveHeterozygous autosomalDominant deNovo xLinked allInheritanceModels VariantFilteringResults

Examples

```
## Not run:

CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.vcf.gz")
param <- VariantFilteringParam(vcfFileNames=CEUvcf)
aim <- allInheritanceModels(param)
aim

## End(Not run)
```

---

autosomalDominant *Autosomal dominant inheritance analysis*

Description

This function identifies variants present in all the affected individual(s) discarding the ones that also occur in at least one of the unaffected subjects.

Usage

```
## S4 method for signature VariantFilteringParam
autosomalDominant(param,
                                                    BPPARAM=bpparam())
```

Arguments

param	An VariantFilteringParam object containing VCF file(s). From 1 to 5 independent files for affected individuals and 0 to 5 more for unaffected ones (up to 10 individuals). If the VCF is a multi-sample, it can contain the same amount of individuals divided likewise.
BPPARAM	An object of class 'BiocParallelParam' specifiying parameters related to the parallel execution of some of the tasks and calculations within this function. See function bpparam() from the BiocParallel package.

Details

This function requires as an input a VariantFilteringParam class object which contains the vcf files ready for the analysis, along with a ped file which specifies the characteristics of each individual present in the analysis.

Value

An object of class VariantFilteringResults including functional annotations on all selected variants.

Author(s)

Dei M. Elurbe and R. Castelo

References

Elurbe D.M., Mila, M., Castelo, R. The VariantFiltering package, in preparation.

See Also

autosomalRecessiveHomozygous autosomalRecessiveHeterozygous xLinked deNovo allInheritanceModels unrelatedIndividuals VariantFilteringResults

Examples

```
## Not run:

CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.vcf.gz")
CEUped <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.ped")
param <- VariantFilteringParam(vcfFileNames=CEUvcf,
                    pedFileName=CEUped)
aDo <- autosomalDominant(param)
aDo

## End(Not run)
```

---

autosomalRecessiveHeterozygous

Autosomal recessive inheritance analysis: Heterozygous

Description

This function aims to analyze the variants of the unaffected individuals, storing and grouping the heterozygous ones by gene. The affected individuals ought present two or more different heterozygous changes in the gene, and at least one of them shall come from each parent.

Usage

```
## S4 method for signature VariantFilteringParam
autosomalRecessiveHeterozygous(param,
                                                BPPARAM=bpparam())
```

Arguments

param	An VariantFilteringParam object containing VCF file(s). From 1 to 5 independent files for affected individuals and 2 more for the carriers of each allele (required). If the VCF is a multi-sample, it can contain the same amount of individuals divided likewise.

BPPARAM — An object of class 'BiocParallelParam' specifiying parameters related to the parallel execution of some of the tasks and calculations within this function. See function bpparam() from the BiocParallel package.

Details

This function requires as an input a VariantFilteringParam class object which contains the vcf files ready for the analysis, along with a ped file which specifies the characteristics of each individual present in the analysis.

For this function, it is not possible to work with all the transcripts and it is limited to variants located into coding regions.

Value

An object of class VariantFilteringResults including functional annotations on all selected variants.

Author(s)

Dei M. Elurbe and R. Castelo

References

Elurbe D.M., Mila, M., Castelo, R. The VariantFiltering package, in preparation.

See Also

autosomalRecessiveHomozygous autosomalDominant xLinked deNovo allInheritanceModels VariantFilteringResults

Examples

```
## Not run:

CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.vcf.gz")
CEUped <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.ped")
param <- VariantFilteringParam(vcfFileNames=CEUvcf,
                    pedFileName=CEUped)
reHet <- autosomalRecessiveHeterozygous(param)
reHet

## End(Not run)
```

`autosomalRecessiveHomozygous` *Autosomal recessive inheritance analysis: Homozygous*

Description

This function works analyzing the variants of the unaffected individuals storing the common heterozygous ones and comparing them with the common homozygous variants between the affected subjects.

Usage

```
## S4 method for signature VariantFilteringParam
autosomalRecessiveHomozygous(param,
                                                BPPARAM=bpparam())
```

Arguments

`param`	An VariantFilteringParam object containing VCF file(s). From 1 to 5 independent files for affected individuals and 0 to 5 more for unaffected ones (up to 10 individuals). If the VCF is a multi-sample, it can contain the same amount of individuals divided likewise.
`BPPARAM`	An object of class 'BiocParallelParam' specifiying parameters related to the parallel execution of some of the tasks and calculations within this function. See function `bpparam()` from the `BiocParallel` package.

Details

This function requires as an input an `VariantFilteringParam` class object which contains the VCF files ready for the analysis, along with a ped file which specifies the characteristics of each individual present in the analysis.

Value

An object of class `VariantFilteringResults` including functional annotations on all selected variants.

Author(s)

Dei M. Elurbe and R. Castelo

References

Elurbe D.M., Mila, M., Castelo, R. The VariantFiltering package, in preparation.

See Also

autosomalRecessiveHeterozygous autosomalDominant xLinked deNovo allInheritanceModels unrelatedIndividuals VariantFilteringResults

Examples

```
## Not run:
library(VariantFiltering)

CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"),
                    "CEUtrio.vcf.gz")
CEUped <- file.path(system.file("extdata", package="VariantFiltering"),
                    "CEUtrio.ped")
param <- VariantFilteringParam(vcfFileNames=CEUvcf, pedFileName=CEUped)
reHo <- autosomalRecessiveHomozygous(param)
reHo

## End(Not run)
```

---

deNovo *De Novo variants analysis*

Description

This function has been created in order to search for *de novo* variants in one individual, discarding the ones shared with his/her parents.

Usage

```
## S4 method for signature VariantFilteringParam
deNovo(param,
                                        BPPARAM=bpparam())
```

Arguments

param	An VariantFilteringParam object containing VCF file(s). 1 independent files for affected individuals and 2 more for unaffected ones (both parents). If the VCF is a multi-sample, it can contain the same amount of individuals divided likewise.
BPPARAM	An object of class 'BiocParallelParam' specifiying parameters related to the parallel execution of some of the tasks and calculations within this function. See function bpparam() from the BiocParallel package.

Details

This function requires as an input a VariantFilteringParam class object which contains the vcf files ready for the analysis, along with a ped file which specifies the characteristics of each individual present in the analysis.

Vcf from both parents are required and only one child at time can be analyzed.

Value

An object of class VariantFilteringResults including functional annotations on all selected variants.

Author(s)

Dei M. Elurbe and R. Castelo

References

Elurbe D.M., Mila, M., Castelo, R. The VariantFiltering package, in preparation.

See Also

autosomalRecessiveHomozygous autosomalDominant autosomalRecessiveHeterozygous xLinked VariantFilteringResults

Examples

```
## Not run:

CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.vcf.gz")
CEUped <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.ped")
param <- VariantFilteringParam(vcfFileNames=CEUvcf,
                    pedFileName=CEUped)
deNo <- deNovo(param)
deNo

## End(Not run)
```

GenePhylostrataDb-class

PhastConsDb class

Description

Class for storing gene-level conservation information in the form of levels of phylogenetic strata; see Neme and Tautz (2013).

Usage

```
## S4 method for signature GenePhylostrataDb
genePhylostrata(object)
## S4 method for signature GenePhylostrataDb
organism(x)
## S4 method for signature GenePhylostrataDb
genePhylostratum(object, ids)
## S4 method for signature GenePhylostrataDb
```

```
annotateVariants(annObj, variantsGR, BPPARAM)
## S4 method for signature GenePhylostrataDb
organism(x)
```

Arguments

object	A GenePhylostrataDb object.
x	A GenePhylostrataDb object.
ids	A string character vector with the gene identifiers to fetch their phylostrata. These identifiers can be only either Ensembl Gene Identifiers (ENSGXXXXX) or Entrez Gene Identifiers.
annObj	A GenePhylostrataDb object.
variantsGR	A GRanges object with the variants to annotate.
BPPARAM	An object of class BiocParallelParam specifying parameters related to the parallel execution of this function. See function bpparam() from the BiocParallel package.

Details

The GenePhylostrataDb class and associated methods serve the purpose of storing and manipulating gene-level conservation information in the form of levels of phylogenetic strata (Neme and Tautz, 2013). One such objects is created at loading time by the VariantFiltering package with the constructor function GenePhylostrataDb(), and it is called humanGenesPhylostrata.

Value

.

Author(s)

R. Castelo

Source

http://genomebiology.com/content/supplementary/1471-2164-14-117-s1.xlsx

References

Neme, R. and Tautz, D. Phylogenetic patterns of emergence of new genes support a model of frequent de novo evolution. BMC Genomics, 14:117, 2013

See Also

phastCons100way.UCSC.hg19

Examples

```
humanGenesPhylostrata
```

MafDb-class *MafDb class*

Description

Class for annotation packages storing minimum allele frequency data.

Usage

```
## S4 method for signature MafDb
fetchKnownVariantsByID(mafdb, varID)
## S4 method for signature MafDb
knownVariantsMAFcols(mafdb)
## S4 method for signature MafDb
keytypes(x)
## S4 method for signature MafDb
keys(x, keytype)
## S4 method for signature MafDb
columns(x)
## S4 method for signature MafDb
select(x, keys, columns, keytype)
## S4 method for signature MafDb
annotateVariants(annObj, variantsGR, BPPARAM)
```

Arguments

mafdb	A MafDb object.
x	A MafDb object.
varID	A variant identifier, typically a rsxxxx dbSNP identifier.
keytype	the keytype that matches the keys used. For MafDb objects there is at the moment only one type of key which is the variant identifier provided by the original data manufacturer.
keys	the keys to select records from the database. All possible keys are turned by using the keys method.
columns	the columns or kinds of things that can be retrieved from the database. As with keys, all possible columns are returned by using the columns method.
annObj	A MafDb object.
variantsGR	A GRanges object with the variants to annotate.
BPPARAM	An object of class BiocParallelParam specifying parameters related to the parallel execution of this function. See function bpparam() from the BiocParallel package.

Details

The MafDb class and associated methods serve the purpose of creating annotation packages that store minimum allele frequency data. Two such annotation packages are:

MafDb.ALL.wgs.phase1.release.v3.20101123	MAF values from the 1000 Genomes project downloaded in November 20
MafDb.ESP6500SI.V2.SSA137.dbSNP138	MAF values from 6500 ESP exomes downloaded in November 2013 from

This object class tries to reduce the disk space required to store allele frequencies (AFs) for millions of SNPs by coding AF float values, which range between 0 and 1, into a single-byte `raw` object type. To achieve this, the original AF values are rounded and coded as follows:

- AF < 0.001 values are rounded to 4 digits, where values 0, 0.0001, ..., 0.0009 are coded as raw byte values 1 to 10.
- AF < 0.01 values are rouneded to 3 digits, where values 0.001, ... 0.009, are coded as raw byte values 11 to 19.
- AF >= 0.01 values are rounded to 2 digits and coded as raw byte values 20 to 119.
- AF `NA` values are coded to raw byte value of 255. Note that by default `NA` values are coded by the raw byte value 0 but this corresponds by default to the null string when raw byte values are coerced into char. This precludes using this original coding of `NA` values.

A further compression of these data is performed in the cases of variants with mutiple alternative alleles. In those cases, instead of storing the AF of each alternate allele only the maximum AF value is stored and as alternate allele the string concatenation of all alleles, separated by a slash character, is stored.

Value

.

Author(s)

R. Castelo

Source

ftp://ftp.1000genomes.ebi.ac.uk

http://evs.gs.washington.edu/EVS

See Also

MafDb.ALL.wgs.phase1.release.v3.20101123 MafDb.ESP6500SI.V2.SSA137.dbSNP138

Examples

```
if (require(MafDb.ESP6500SI.V2.SSA137.dbSNP138)) {
  MafDb.ESP6500SI.V2.SSA137.dbSNP138

  ## specialized interface
  knownVariantsMAFcols(MafDb.ESP6500SI.V2.SSA137.dbSNP138)
  fetchKnownVariantsByID(MafDb.ESP6500SI.V2.SSA137.dbSNP138, "rs199529001")

  ## standard AnnotationDbi interface
```

```
  keytypes(MafDb.ESP6500SI.V2.SSA137.dbSNP138)
  columns(MafDb.ESP6500SI.V2.SSA137.dbSNP138)
  select(MafDb.ESP6500SI.V2.SSA137.dbSNP138,
         keys="rs199529001", columns=c("varID", "chrom", "AF"))
}
```

PhastConsDb-class *PhastConsDb class*

Description

Class for annotation packages storing UCSC phastCons conservation scores.

Usage

```
## S4 method for signature PhastConsDb
annotateVariants(annObj, variantsGR, BPPARAM)
## S4 method for signature PhastConsDb,GRanges
scores(object, gpos,
                                        summaryFun="mean",
                                        coercionFun="as.numeric",
                                        caching=TRUE)
```

Arguments

annObj	A PhastConsDb object.
variantsGR	A GRanges object with the variants to annotate.
BPPARAM	An object of class BiocParallelParam specifying parameters related to the parallel execution of this function. See function bpparam() from the BiocParallel package.
object	A PhastConsDb object.
gpos	A GRanges object with positions from where to retrieve phastCons scores.
summaryFun	Function to summarize phastCons scores when more than one position is retrieved. By default is set to the arithmetic mean.
coercionFun	Function to coerce the stored phastCons scores, before the summary function is applied. By default phastCons scores are coerced to real values.
caching	Flag setting whether phastCons scores per chromosome should be kept cached in memory (TRUE, default) or not (FALSE). The latter option minimizes the memory footprint but slows down the performance when the scores() method is called multiple times.

Details

The PhastConsDb class and associated methods serve the purpose of creating annotation packages that store phastCons nucleotide-level conservation scores from the UCSC Genome Browser. One such annotation package is:

phastCons100way.UCSC.hg19 Nucleotide-level phastCons conservation scores from the UCSC Genome Browser download

Value

.

Author(s)

R. Castelo

Source

http://genome.ucsc.edu

See Also

phastCons100way.UCSC.hg19

Examples

```
if (require(phastCons100way.UCSC.hg19)) {
  library(GenomicRanges)

  phastCons100way.UCSC.hg19
  scores(phastCons100way.UCSC.hg19,
         GRanges(seqnames="chr7", IRanges(start=117232380, width=5)))
}
```

---

readAAradicalChangeMatrix

Read matrix of amino acid radical changes

Description

Function to read and parse a tab-separated file of amino acid properties into a matrix of logical values indicating whether the change of one amino acid by another can be considered radical or conservative according to the chemical properties specified in the input file.

Usage

```
readAAradicalChangeMatrix(file)
```

Arguments

file	A file containing a classification of amino acids with respect to one or more chemical properties. Its particular format should match the one from the file called AA_chemical_properties_HanadaGojoboriLi2006.tsv found in the extdata folder of this package. This file is based on Table 1 from Hanada et al. (2006).

Details

The input file should contain one or more columns each of them forming a logical mask specifing sets of amino acids sharing some chemical property.

Value

An squared symmetric matrix with as many rows and columns as amino acids and whose cells contain logical values. These values are set to TRUE whenever the amino acid change implied by row and column is considered radical and FALSE when considered conservative. Amino acid changes within a chemical property are defined as conservative and radical otherwise.

Author(s)

R. Castelo

References

Hanada, K., Gojobori, T. and Li, W. Radical amino acid change versus positive selection in the evolution of viral envelope proteins. Gene, 385:83-88, 2006.

See Also

VariantFilteringParam

Examples

```
aamat <- readAAradicalChangeMatrix(file.path(system.file("extdata", package="VariantFiltering"),
                                   "AA_chemical_properties_HanadaGojoboriLi2006.tsv"))
aamat[1:5, 1:5]
```

---

unrelatedIndividuals *Analysis designed to be applied over a group of unrelated individuals*

Description

This function is designed to create an object to deepen into the variants presented by a group of unrelated individuals

Usage

```
## S4 method for signature VariantFilteringParam
unrelatedIndividuals(param,
                                                    BPPARAM=bpparam())
```

Arguments

param	An VariantFilteringParam object containing a VCF file.
BPPARAM	An object of class 'BiocParallelParam' specifiying parameters related to the parallel execution of some of the tasks and calculations within this function. See function bpparam() from the BiocParallel package.

Details

This function requires as an input a VariantFilteringParam class object which contains the vcf file ready for the analysis.

Value

An object of class VariantFilteringResults including functional annotations on all variants.

Author(s)

Dei M. Elurbe and R. Castelo

References

Elurbe D.M., Mila, M., Castelo, R. The VariantFiltering package, in preparation.

See Also

allInheritanceModels VariantFilteringResults

Examples

```
## Not run:

CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.vcf.gz")
param <- VariantFilteringParam(vcfFileNames=CEUvcf)
uInd <- unrelatedIndividuals(param)
uInd

## End(Not run)
```

VariantFilteringParam-class

VariantFiltering parameter class

Description

The class VariantFilteringParam is defined to ease configuring the call to the functions that filter input genetic variants according to a desired segregating inheritance model (xLinked(), autosomalRecessiveHomozygous() etc).

Usage

```
## S4 method for signature character
VariantFilteringParam(vcfFilenames, pedFilename=character(),
                                          orgdb="org.Hs.eg.db",
                                      txdb="TxDb.Hsapiens.UCSC.hg19.knownGene",
                                       snpdb="SNPlocs.Hsapiens.dbSNP.20120608",
                                radicalAAchangeFilename=file.path(system.file("extdata", package="Vari
                                                        "AA_chemical_properties_HanadaGojoboriLi2006.ts
                                                    allTranscripts=FALSE,
                                otherAnnotations=c("MafDb.ESP6500SI.V2.SSA137.dbSNP138",
                                                   "MafDb.ALL.wgs.phase1.release.v3.20101123",
                                                          "PolyPhen.Hsapiens.dbSNP131",
                                                              "SIFT.Hsapiens.dbSNP137",
                                                           "phastCons100way.UCSC.hg19",
                                                              "humanGenesPhylostrata"),
                                                    filterTag=NA_character_)
## S4 method for signature VariantFilteringParam
show(object)
## S4 method for signature VariantFilteringParam
x$name
## S4 method for signature VariantFilteringParam
names(x)
```

Arguments

vcfFilenames	Character vector of paths to VCF files.
pedFilename	Character string of the pedigree file name in PED format.
orgdb	Character string of a gene-centric annotation package (org.Hs.eg.db by default).
txdb	Character string of a transcript-centric annotation package (TxDb.Hsapiens.UCSC.hg19.knownGene by default). The package GenomicFeatures provides infraestructure to build such annotation packages from different sources such as online UCSC tracks, Biomart tables, or GFF files.
snpdb	Character string of a SNP-centric annotation package (SNPlocs.Hsapiens.dbSNP.20120608 by default).
radicalAAchangeFilename	Name of a tab-separated text file containing chemical properties of amino acids. These properties are interpreted such that amino acid changes within a property are considered "conservative" and between properties are considered "radical". See the default file (AA_chemical_properties_HanadaGojoboriLi2006.tsv) for details on its format.
allTranscripts	Logical. This option allows the user to choose between working with all the transcripts affected by the variant (allTranscripts=TRUE) or with only one transcript per variant.
otherAnnotations	Character vector of names of annotation packages or annotation objects.

filterTag	Character vector of tags used to select only those variants which present the corresponding string (or strings) within the VCF's FILTER column. NA_character_ (default) means that it will consider all the variants present in the VCF file to perform the analysis. An example of possible value is PASS, a commonly used flag to indicate that the call fullfils all the quality parameters, so only variants flagged this way will be used in the analysis.
object	A VariantFilteringParam object created through VariantFilteringParam().
x	A VariantFilteringParam object created through VariantFilteringParam().
name	Slot name of a VariantFilteringParam object. Use names() to find out what these slots are.

Details

The class VariantFilteringParam serves as a purpose of simplifying the call to the inheritance model function and its subsequent annotation and filtering steps. It also groups all the parameters that the user can customize (i.e newer versions of the annotation packages, when available).

The method VariantFilteringParam() creates an VariantFilteringParam object used as an input argument to other functions such as autosomalRecessiveHomozygous(), etc.

The method names() allows one to see the names of the slots from a VariantFilteringParam object. Using the $ operator, one can retrieve the values of these slots in an analogous way to a list.

Value

An VariantFilteringParam object is returned by the method VariantFilteringParam.

Author(s)

D.M. Elurbe and R. Castelo

Examples

```
p <- VariantFilteringParam(list.files(system.file("extdata", package="VariantFiltering"), "CEUtrio.vcf.gz$", ful
p
names(p)
p$vcfFiles
```

VariantFilteringResults-class

VariantFiltering results class

Description

The class `VariantFilteringResults` is defined to ease manipulating the results from the calls to functions that filter input genetic variants according to a desired segregating inheritance model (xLinked(), recessiveHomozygous(), etc).

The classes `VariantFilteringResultsAIM` and `VariantFilteringResultsUI` are both inherited from the `VariantFilteringResults` one but created through the `allInheritanceModels()` and `unrelatedIndividuals()` functions respectively.

Details

The class `VariantFilteringResults` serves as a purpose of manipulating the results and applying filters on functional annotations of the variants. This manipulation takes place by means of accessor functions enumerated below.

Variant Annotation data

An `VariantFilteringResults` object contains the following annotation data for each variant:

GENE Gene name by the HGNC

CHR Chromosome number, following the UCSC standards

POS Location of the variant inside de chromosome

LOCATION Region where the variant is located (coding, splice site, promoter...)

TYPE Type of variant (SNV, InDel...)

dbSNP dbSNP ID

CONSEQUENCE Consequence in the translation of the sequence

TXNAME Transcript name extracted from the `TxDb` object defined at `txdb`

CDS Reference and variant nucleotides in the cDNA

PROT Reference and variant amino acid in the amino acid chain

PolyPhen2 PolyPhen2 prediction for the variant

SIFT SIFT prediction for the variant

OMIM OMIM ID entry for the gene

AFKG Minor Allele Frequency from the 1000 Genomes project, all populations

AMR_AFKG Minor Allele Frequency from the 1000 Genomes project, Ad Mixed American

ASN_AFKG Minor Allele Frequency from the 1000 Genomes project, East Asian

AFR_AFKG Minor Allele Frequency from the 1000 Genomes project, African

EUR_AFKG Minor Allele Frequency from the 1000 Genomes project, European

AFESP Minor Allele Frequency from the NHLBI Exome Sequencing Project, all populations

EA_AFESP Minor Allele Frequency from the NHLBI Exome Sequencing Project, European American

AA_AFESP Minor Allele Frequency from the NHLBI Exome Sequencing Project, African American

CRYP5ssREF Score for the cryptic 5'ss for the REF allele respect to the ALT allele

CRYP5ssALT Maximum score for the cryptic 5'ss taking into account all the possible positions within the window

CRYP5ssPOS Position of the allele respect to the position of the dinucleotide GT, considering those as positions 1 and 2

CRYP3ssREF Score for the cryptic 3'ss for the REF allele respect to the ALT allele

CRYP3ssALT Maximum score for the cryptic 3'ss taking into account all the possible positions within the window

CRYP3ssPOS Position of the allele respect to the position of the dinucleotide AG, considering those as positions 1 and 2

Accessor methods

In the calls below, `x` is a `VariantFilteringResults` object.

`param(x)`: return the `VariantFilteringParam` input parameter object employed in the call that produced the `VariantFilteringResults` object `x`.

`inheritanceModel(x)`: return the model of inheritance employed in the call that produced the `VariantFilteringResults` object `x`.

`dbSNPpresent(x)`: flag whether to filter variants present or absent from dbSNP (`NA` -do not filter-, `"Yes"`, `"No"`).

`variantType(x)`: filter by type of variant (`"Any"`, `"SNV"`, `"Indel"`, `"MNV"`).

`aaChangeType(x)`: filter by type of change of amino acid (`"Any"`, `"Radical"`, `"Conservative"`).

`OMIMpresent(x)`: flag whether to filter variants whose associated genes are present or absent from OMIM (`NA` -do not filter-, `"Yes"`, `"No"`).

`naMAF(x)`: flag whether NA maximum MAF values should be included in the filtered variants.

`maxMAF(x)`: maximum MAF value that a variant may meet among the selected populations.

`minPhastCons(x)`: minimum phastCons score for nucleotide conservation (`NA` -do not filter-, [0-1]).

`minPhylostratum(x)`: minimum phylostratum for gene conservation (`NA` -do not filter-, [1-20]).

`MAFpop(x)`: selection of populations to use when filtering by maximum MAF value.

`minCRYP5ss(x)`: minimum weight matrix score on a cryptic 5'ss. `NA` indicates this filter is not applied.

`minCRYP3ss(x)`: minimum weight matrix score on a cryptic 3'ss. `NA` indicates this filter is not applied.

`allVariants(x)`: get a `GRanges` object with all variants without applying any filter.

`filteredVariants(x)`: get a `GRanges` object with the variants obtained after applying all the filters.

`selectIndividual(x)`: selection of individuals for further analysis. `NA` indicates that all individuals present in the VCF file are selected.

`reportVariants(x, type=c("shiny", "csv", "tsv"), file=NULL)`: Builds a report from an `VariantFilteringResult` object. Using the `type` argument, the report can take the form of a flat file in CSV or TSV format or a web `shiny` app (default) that enables applying functional annotation filters in an interactive manner.

When the `shiny` app is closed this method returns a `VariantFilteringResult` object with the corresponding filters switched on or off according to how the app has been interactively used.

Author(s)

D.M. Elurbe and R. Castelo

Examples

```
## Not run:
library(VariantFiltering)

CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"),
                    "CEUtrio.vcf.gz")
CEUped <- file.path(system.file("extdata", package="VariantFiltering"),
                    "CEUtrio.ped")
param <- VariantFilteringParam(vcfFileNames=CEUvcf, pedFileName=CEUped)
reHo <- autosomalRecessiveHomozygous(param)
naMAF(reHo) <- FALSE
maxMAF(reHo) <- 0.05
reHo
head(filteredVariants(reHo))
reportVariants(reHo, type="csv", file="reHo.csv")

## End(Not run)
```

WeightMatrix-class *Weight matrix class*

Description

Class for storing weight matrices that VariantFiltering uses to score potential cryptic splice sites.

Usage

```
## S4 method for signature WeightMatrix
width(x)
## S4 method for signature WeightMatrix
conservedPositions(x)
## S4 method for signature WeightMatrix
show(object)
## S4 method for signature WeightMatrix,DNAStringSet
wmScore(object, dnaseqs)
## S4 method for signature WeightMatrix,character
wmScore(object, dnaseqs)
```

Arguments

x	A `WeightMatrix` object.
object	A `WeightMatrix` object.
dnaseqs	Either a vector of character strings a `DNAStringSet` object, both of which store nucleotide sequences to be scored using the input `WeightMatrix` object.

Details

The WeightMatrix class and associated methods serve the purpose of enabling the VariantFiltering package to score synonymous and intronic genetic variants for potential cryptic splice sites. The class and the methods, however, are exposed to the end user since they could be useful for other analysis purposes.

The VariantFiltering package contains two weight matrices, one for 5'ss and another for 3'ss, which have been built by a statistical method that accounts for dependencies between the splice site positions, minimizing the rate of false positive predictions. The method concretely builds these models by inclusion-driven learning of Bayesian networks and further details can be found in the paper of Castelo and Guigo (2004).

The function readWm() reads a weight matrix stored in a text file in a particular format and returns a WeightMatrix object. See the .ibn files located in the extdata folder of the VariantFiltering package, as an example of this format.

The method wmScore() scores one or more sequences of nucleotides using the input WeightMatrix object. If the sequences are longer than the width of the weight matrix, this function will score every possible site within those sequences. It returns a vector of with the calculated scores. When the scores cannot be calculated because of a conserved position that does not occur in the sequence (i.e., absence of a GT dinucleotide with the 5'ss weight matrix), it returns NA as corresponding score value.

The method width() takes a WeightMatrix object as input and returns the number of positions of the weight matrix.

The method conservedPositions() takes a WeightMatrix object as input and returns the number of fully conserved positions in the weight matrix.

Value

.

Author(s)

R. Castelo

References

Castelo, R and Guigo, R. Splice site identification by idlBNs. Bioinformatics, 20(1):i69-i76, 2004.

Examples

```
wm <- readWm(file.path(system.file("extdata", package="VariantFiltering"), "hsap.donors.hcmc10_15_1.ibn"))
width(wm)
conservedPositions(wm)
wmScore(wm, "CAGGTAGGA")
wmScore(wm, "CAGGAAGGA")
wmScore(wm, "CAGGTCCTG")
wmScore(wm, "CAGGTCGTGGAG")
```

xLinked *X-Linked inheritance analysis*

Description

This function identifies variants that appear only in the X chromosome of the unaffected females as heterozygous, don't appear in the unaffected males analyzed and finally are present (as homozygous) in the affected male(s).

Usage

```
## S4 method for signature VariantFilteringParam
xLinked(param,
                                        BPPARAM=bpparam())
```

Arguments

param	An VariantFilteringParam object containing VCF file(s). From 1 to 5 independent files for affected individuals, 0 to 5 more for the carrier females and 0 to 5 additional individuals as unaffected males (up to 15 individuals). If the VCF is a multi-sample, it can contain the same amount of individuals divided likewise.
BPPARAM	An object of class 'BiocParallelParam' specifiying parameters related to the parallel execution of some of the tasks and calculations within this function. See function bpparam() from the BiocParallel package.

Details

This function requires as an input a VariantFilteringParam class object which contains the vcf files ready for the analysis, along with a ped file which specifies the characteristics of each individual present in the analysis.

Value

An object of class VariantFilteringResults including functional annotations on all selected variants.

Author(s)

Dei M. Elurbe and R. Castelo

References

Elurbe D.M., Mila, M., Castelo, R. The VariantFiltering package, in preparation.

See Also

autosomalRecessiveHomozygous autosomalRecessiveHeterozygous autosomalDominant deNovo allInheritanceModels unrelatedIndividuals VariantFilteringResults

Examples

```
## Not run:

## This actually wont run b/c in this trio de descendant is a female
CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.vcf.gz")
CEUped <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.ped")
param <- VariantFilteringParam(vcfFileNames=CEUvcf,
                    pedFileName=CEUped)
xlid <- xLinked(param)
xlid

## End(Not run)
```

Index