Package 'VariantFiltering'

April 16, 2019

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, minor allele frequencies across human populations, splice site strength, conservation, etc.

Version 1.18.0

License Artistic-2.0

Depends R (>= 3.0.0), methods, BiocGenerics (>= 0.25.1), VariantAnnotation (>= 1.13.29)

Imports utils, stats, Biobase, S4Vectors (>= 0.9.25), IRanges (>= 2.3.23), RBGL, graph, AnnotationDbi, BiocParallel, Biostrings (>= 2.33.11), GenomeInfoDb (>= 1.3.6), GenomicRanges (>= 1.19.13), SummarizedExperiment, GenomicFeatures, Rsamtools (>= 1.17.8), BSgenome, GenomicScores (>= 1.0.0), Gviz, shiny, shinythemes, shinyjs, DT, shinyTree

Suggests RUnit, BiocStyle, org.Hs.eg.db, BSgenome.Hsapiens.1000genomes.hs37d5, TxDb.Hsapiens.UCSC.hg19.knownGene, SNPlocs.Hsapiens.dbSNP144.GRCh37, MafDb.1Kgenomes.phase1.hs37d5, phastCons100way.UCSC.hg19, PolyPhen.Hsapiens.dbSNP131, SIFT.Hsapiens.dbSNP137

LinkingTo S4Vectors, IRanges, XVector, Biostrings

LazyData yes

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

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

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

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

git_branch RELEASE_3_8

git_last_commit e217023

git_last_commit_date 2018-10-30

Date/Publication 2019-04-15

Author Robert Castelo [aut, cre],
Dei Martinez Elurbe [ctb],
Pau Puigdevall [ctb],
Joan Fernandez [ctb]

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

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, minor 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.
- `unrelatedIndividuals` annotate variants without filtering by any inheritance pattern.

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.

`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,
                                                    svparam=ScanVcfParam(),
                                          use=c("everything", "complete.obs", "all.obs"),
                                                BPPARAM=bpparam("SerialParam"))
```

Arguments

`param`	A `VariantFilteringParam` object built from a multisample VCF file with at least one affected individual and zero or more unaffected ones, and from a PED file specyfing the family relationships among individuals as well as their gender and phenotype status (affected or unaffected).
`svparam`	An instance of a `ScanVcfParam` object to enable analyzing a subset of variants and samples. This object is passed internally to a call to the `readVcf()` function in the `VariantAnnotation` package, see its help page for a complete description of this functionality.
`use`	character string specifying the policy to apply on missing genotypes when comparing them. This policy can be either '"everything"' (default), '"complete.obs"' or '"all.obs"'. The default policy ('"everything"') will propagate NA truth values using the behavior of the R logical operators, with the exception that when the final truth value associated with a variant is `NA`, the variant is ultimately discarded.
`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 built from an input multisample VCF file, along with a PED file.

Value

An object of class VariantFilteringResults including functional annotations on variants segregating according to an autosomal dominant inheritance model.

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 unrelatedIndividuals VariantFilteringResults

Examples

```
## Not run:

CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.vcf.bgz")
CEUped <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.ped")
param <- VariantFilteringParam(vcfFilename=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,
                                                    svparam=ScanVcfParam(),
                                                BPPARAM=bpparam("SerialParam"))
```

Arguments

param	A VariantFilteringParam object built from a multisample VCF file with at least one affected individual and two or more unaffected ones, and from a PED file specyfing the family relationships among individuals as well as their gender and phenotype status (affected or unaffected).
svparam	An instance of a ScanVcfParam object to enable analyzing a subset of variants and samples. This object is passed internally to a call to the readVcf() function in the VariantAnnotation package, see its help page for a complete description of this functionality.
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 built from an input multisample VCF file, along with a PED file.

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 VariantFilteringResults

Examples

```
## Not run:

CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.vcf.bgz")
CEUped <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.ped")
param <- VariantFilteringParam(vcfFilename=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,
                                                    svparam=ScanVcfParam(),
                                  use=c("everything", "complete.obs", "all.obs"),
                                                  includeHomRef=FALSE,
                                                     age.of.onset=999,
                                                      phenocopies=0,
                                  BPPARAM=bpparam("SerialParam"))
```

Arguments

`param`	A `VariantFilteringParam` object built from a multisample VCF file with at least one affected individual and zero or more unaffected ones, and from a PED file specyfing the family relationships among individuals as well as their gender and phenotype status (affected or unaffected).
`svparam`	An instance of a `ScanVcfParam` object to enable analyzing a subset of variants and samples. This object is passed internally to a call to the `readVcf()` function in the `VariantAnnotation` package, see its help page for a complete description of this functionality.
`use`	character string specifying the policy to apply on missing genotypes when comparing them. This policy can be either '"everything"' (default), '"complete.obs"' or '"all.obs"'. The default policy ('"everything"') will propagate NA truth values using the behavior of the R logical operators, with the exception that when the final truth value associated with a variant is `NA`, the variant is ultimately discarded.
`includeHomRef`	logical value specifying whether the homozygous reference genotype determines the affected phenotype (`TRUE`) or not (`FALSE`, default).
`age.of.onset`	numerical value specifying the age of onset at which individuals below that age are set their phenotype to unknown. This argument is currently experimental.
`phenocopies`	numerical value specifying the maximum fraction of affected individuals not sharing the homozygous genotype.
`BPPARAM`	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.

Details

This function requires as an input a VariantFilteringParam class object built from an input multisample VCF file, along with a PED file.

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 unrelatedIndividuals VariantFilteringResults

Examples

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

CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"),
                    "CEUtrio.vcf.bgz")
CEUped <- file.path(system.file("extdata", package="VariantFiltering"),
                    "CEUtrio.ped")
param <- VariantFilteringParam(vcfFilename=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 and annotate *de novo* variants in one individual, discarding the ones shared with his/her parents.

Usage

```
## S4 method for signature 'VariantFilteringParam'
deNovo(param,
                                        svparam=ScanVcfParam(),
                                  use=c("everything", "complete.obs", "all.obs"),
                                        BPPARAM=bpparam("SerialParam"))
```

Arguments

param	A VariantFilteringParam object built from a multisample VCF file with at least one affected individual and zero or more unaffected ones, and from a PED file specyfing the family relationships among individuals as well as their gender and phenotype status (affected or unaffected).
svparam	An instance of a ScanVcfParam object to enable analyzing a subset of variants and samples. This object is passed internally to a call to the readVcf() function in the VariantAnnotation package, see its help page for a complete description of this functionality.
use	character string specifying the policy to apply on missing genotypes when comparing them. This policy can be either '"everything"' (default), '"complete.obs"' or '"all.obs"'. The default policy ('"everything"') will propagate NA truth values using the behavior of the R logical operators, with the exception that when the final truth value associated with a variant is NA, the variant is ultimately discarded.
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 built from an input multisample VCF file, along with a PED file.

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.bgz")
CEUped <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.ped")
param <- VariantFilteringParam(vcfFilename=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'
genePhylostratum(object, ids)
## S4 method for signature 'GenePhylostrataDb'
organism(object)
```

Arguments

object	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.

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
```

`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,
                                                    svparam=ScanVcfParam(),
                                          BPPARAM=bpparam("SerialParam"))
```

Arguments

param	A VariantFilteringParam object built from a multisample VCF file.
svparam	An instance of a ScanVcfParam object to enable analyzing a subset of variants and samples. This object is passed internally to a call to the readVcf() function in the VariantAnnotation package, see its help page for a complete description of this functionality.
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 built from an input multisample VCF file.

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 VariantFilteringResults

Examples

```
## Not run:

CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.vcf.bgz")
param <- VariantFilteringParam(vcfFilename=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()`, `autosomalRecessiveHomozygo` etc).

Usage

```
VariantFilteringParam(vcfFilename, pedFilename=NA_character_,
                      bsgenome="BSgenome.Hsapiens.1000genomes.hs37d5",
                      orgdb="org.Hs.eg.db",
                      txdb="TxDb.Hsapiens.UCSC.hg19.knownGene",
                      snpdb="SNPlocs.Hsapiens.dbSNP144.GRCh37",
                      weightMatricesFilenames=NA,
                  weightMatricesLocations=rep(list(variantLocations()), length(weightMatricesFile
                  weightMatricesStrictLocations=rep(list(FALSE), length(weightMatricesFilenames))
                      radicalAAchangeFilename=file.path(system.file("extdata",
                                                        package="VariantFiltering"),
                                        "AA_chemical_properties_HanadaGojoboriLi2006.tsv"),
                      codonusageFilename=file.path(system.file("extdata",
                                                   package="VariantFiltering"),
                                                   "humanCodonUsage.txt"),
                      geneticCode=getGeneticCode("SGC0"),
                      allTranscripts=FALSE,
                     regionAnnotations=list(CodingVariants(), IntronVariants(),
                                  FiveSpliceSiteVariants(), ThreeSpliceSiteVariants(),
                                  PromoterVariants(), FiveUTRVariants(), ThreeUTRVariants()),
                      intergenic=FALSE,
                      otherAnnotations=c("MafDb.1Kgenomes.phase1.hs37d5",
                                         "PolyPhen.Hsapiens.dbSNP131",
                                         "SIFT.Hsapiens.dbSNP137",
                                         "phastCons100way.UCSC.hg19",
                                         "humanGenesPhylostrata"),
                      geneKeytype=NA_character_,
                      yieldSize=NA_integer_)
## S4 method for signature 'VariantFilteringParam'
show(object)
```

```
## S4 method for signature 'VariantFilteringParam'
x$name
## S4 method for signature 'VariantFilteringParam'
names(x)
```

Arguments

vcfFilename	Character string of the input VCF file name.
pedFilename	Character string of the pedigree file name in PED format.
bsgenome	Character string of a genome annotation package (BSgenome.Hsapiens.1000genomes.hs37d5 by default).
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).
weightMatricesFilenames	Character string of filenames of position weight matrices for binding site recognition. The default NA value indicates that no binding sites will be scored. To use this feature to score, for instance, splice sites in human, assign to this argument the function spliceSiteMatricesHuman(). See the files (hsap.donors.hcmc10_15_1.ibn and hsap.acceptors.hcmc10_15_1.ibn) returned by this function for details on their format.
weightMatricesLocations	Keywords of the annotated locations to variants under which a weight matrix will be used for scoring binding sites. This argument is only used when weightMatricesFilenames!=NA and, in such case, then more than one matrix is provided, this argument should be a list of character vectors with as many elements as matrices given in weightMatricesFilenames. The possible values can be obtained by typing variantLocations().
weightMatricesStrictLocations	Logical vector flagging whether a weight matrix should be scoring binding sites strictly within the boundaries of the given locations. This argument is only used when weightMatricesFilenames!=NA and, in such case, then more than one matrix is provided, this argument should be a list of logical vectors with as many elements as matrices given in weightMatricesFilenames.
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.
codonusageFilename	Name of a text file containing the codon usage.
geneticCode	Name character vector of length 64 describing the genetic code. The default value is getGeneticCode("SGC0"), the standard genetic code. An alternative

genetic code, for instance, is `getGeneticCode("SGC1")`, the vertebrate mitochondrial genetic code. See `getGeneticCode` in the Biostrings package for further details.

`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.

`regionAnnotations`
List of `VariantType-class` objects defining what regions to annotate.

`intergenic` Logical. When `TRUE`, the intergenic variants are also annotated.

`otherAnnotations`
Character vector of names of annotation packages or annotation objects.

`geneKeytype` Character vector of the type of key gene identifier provided by the transcript-centric (TxDb) annotation package to interrogate the organism-centric (OrgDb) annotation package. The default value (`NA_character_` indicates that it will be assumed to be an Entrez identifier unless the values in the `GENEID` column returned by the TxDb package start with `ENSG` and then it will be assumed that they are Ensembl gene identifiers, or with one of `NM_, NP_, NR_, XM_, XP_, XR_ or YP_` and then it will be assumed that they are RefSeq gene identifiers.

`yieldSize` Number of variants to yield each time the input VCF file is read. This argument is passed to the `TabixFile` function when opening the input VCF file and it allows to iterate through the variants in chunks of the given size to limit the memory requirements. Its default value (`NA_integer_`) implies that the whole input VCF file will be read into main memory.

`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, P. Puigdevall and R. Castelo

Examples

```
vfpar <- VariantFilteringParam(system.file("extdata", "CEUtrio.vcf.bgz", package="VariantFiltering"),
                          system.file("extdata", "CEUtrio.ped", package="VariantFiltering"),
                                snpdb=character(0), otherAnnotations=character(0))
vfpar
names(vfpar)
vfpar$vcfFiles
```

VariantFilteringResults-class

VariantFilteringResults class

Description

The VariantFilteringResults class is used to store the kind of object obtained as a result of an analysis using the functions unrelatedIndividuals(), autosomalRecessiveHomozygous(), autosomalRecessiveHeterozygous(), autosomalDominant(), deNovo() and xLinked(). Its purpose is to ease the task of filtering and prioritizing the variants annotated by those functions.

Details

Variants are stored within a VariantFilteringResults object using a VRanges object, which also holds the variant annotations in its metadata columns. VariantFiltering adds the following core set of annotations.

LOCATION Region where the variant is located (coding, intronic, splice site, promoter, ...) as given by the function locateVariants() from the VariantAnnotation package.

LOCSTART Start position of the variant within the region defined by the LOCATION annotation.

GENEID Gene identifier derived with the transcript-centric annotation package given in the txdb argument of the VariantFilteringParam() function, typically an Entrez Gene identifier.

GENE Gene name given by HGNC derived with the gene-centric annotation package given in the orgdb argument of the VariantFilteringParam() function.

TYPE Type of variant, either a single nucleotide variant (SNV), an insertion, a deletion, a multinucleotide variant (MNV) or a deletion followed by an insertion (Delins). These types are determined using functions isSNV(), isInsertion(), isDeletion(), isSubstitution() and isDelins() from the VariantAnnotation package.

dbSNP dbSNP identifier derived by position from the annotation packages given in the snpdb argument of the VariantFilteringParam() function.

cDNALOC Location of the variant along the processed transcript, when the variant belongs to an exonic region.

CONSEQUENCE Consequence of the variant when located in the coding region (synonymous, nonsynonymous, missense, nonsense o frameshift) as given by the function predictCoding() from the VariantAnnotation package.

TXNAME Transcript name extracted from the TxDb annotation package given by the txdb argument of the VariantFilteringParam() function.

HGVSg HGVS description of the variant at genomic level.

HGVSc HGVS description of the variant at coding level.

HGVSp HGVS description of the variant at protein level.

OMIM OMIM identifier of the gene associated to the variant derived with the gene-centric annotation package given in the orgdb argument of the VariantFilteringParam() function.

AAchangeType In the case of coding variants, whether the amino acid change is conservative or radical according to the matrix of amino acid biochemical properties given in the argument radicalAAchangeFilename of the VariantFilteringParam() function.

SCORE5ssREF Score for the cryptic 5'ss for the REF allele respect to the ALT allele.

SCORE5ssALT Maximum score for a potential cryptic 5'ss created by the ALT allele.

SCORE5ssPOS Position of the allele respect to the position of the dinucleotide GT, considering those as positions 1 and 2.

SCORE3ssREF Score for the cryptic 3'ss for the REF allele respect to the ALT allele.

SCORE3ssALT Maximum score for a potential cryptic 3'ss created by the ALT allele.

SCORE3ssPOS Position of the allele respect to the position of the dinucleotide AG, considering those as positions 1 and 2.

Accessors

A VariantFilteringResults has the following set of accessor methods.

length(x): total number of variants stored internatlly within the VRanges object. Note that this number will be typically larger than the number of variantes in the input VCF object because each of them is copied for each combination of alternate allele, annotated region and sample.

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

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

samples(object): active samples from which the current filtered variants were derived. If the x was obtained with unrelatedIndividuals(), then the replace method samples(object)<- can be used to restrict the subset of active samples. In every other case (autosomalDominant(), etc.) active samples cannot be changed.

resetSamples(object): set back as active samples the initial set of samples specified in the input parameter object.

sog(x): Sequence Ontology (SO) graph (actually, an acyclic digraph) returned as a graphNEL object, whose vertices are SO terms, edges represent ontology relationships and vertex attributes vcfIdx and varIdx contain what variants are annotated to each SO term. These annotations can be directly retrieved from the SO graph with the nodeData() function from the graph package. The summary() function described in this manual page allows one to tally the number of variants in each SO term throughout the entire SO hierarchy.

bamFiles(x): access and update the BamViews object containing references to BAM files from which the input VCF files were derived. Initially this is empty.

allVariants(x, groupBy="sample"): returns a VRangesList object with all variants grouped by default by sample. Using the argument groupBy we can specify any metadata column to be used to group variants. If the value given to groupBy does not correspond to any such columns, a VRanges object with all variants together is returned.

filteredVariants(x, groupBy="sample"): it works like allVariants(x) but instead of returning all variants, it returns only those who pass the active filters; see filters() and cutoffs() below.

Filters and cutoffs

The variants contained in a VariantFilteringResults object can be filtered using the FilterRules mechanism, defined in the S4Vectors package, by using the functions filters() and cutoffs() described below. There are additional functions, also described in this section, to facilitate this task on the set of core annotations provided by VariantFiltering.

filters(x): get the current FilterRules object that defines the available set of filter criteria that one can use to filter the variants contained in x. This can also be used as a replacement function filters(x)<- to update this set of filters. The actual filtering is done when calling the function filteredVariants().

filtersMetadata(x): metadata about the available filters.

cutoffs(x): get cutoffs from the available filters.

change(x, cutoff)<-: change cutoffs from the available filters. Here, argument x is a CutoffsList object given by the method cutoffs(), and argument cutoff is a character string with the name of the cutoff.

softFilterMatrix(x): get and update the variant by filter matrix; see softFilterMatrix() in the VariantAnnotation package.

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 ("SNV", "Insertion", "Deletion", "MNV", "Delins").

variantLocation(x): filter by variant location ("coding", "intron", "threeUTR", "fiveUTR", "intergenic", "spliceSite", "promoter").

variantConsequence(x): filter by variant consequence ("snynonymous", "nonsynonymous", "frameshift", "nonsense", "not translated").

aaChangeType(x): filter by type of change of amino acid ("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.

minScore5ss(x): minimum weight matrix score on a cryptic 5'ss. NA indicates this filter is not applied.

minScore3ss(x): minimum weight matrix score on a cryptic 3'ss. NA indicates this filter is not applied.

minCUFC(x): minimum absolute codon-usage log2 fold-change.

Summarization, visualization and reporting

The following functions help in summarizing, visualizing and reporting the fiiltered variants.

summary(object, method=c("SO", "SOfull", "bioc")): tally the current filtered set of variants to features. By default, features are Sequence Ontology (SO) terms to which variants are annotated by VariantFiltering. The method argument allows the user to change this default setting to tallying throughout the entire SO hierarchy. Both options, SO and

SOfull can be used in combination with the cutoff SOterms; see the vignette. The option method="bioc" considers as features the regions and consequences annotated by functions locateVariants() and predictCoding() from the VariantAnnotation package. The result is returned as a data.frame object.

plot(x, what, sampleName, flankingNt=20, showAlnNtCutoff=200, isPaired=FALSE, ...): Plot variants using the Gviz package. The argument what can be either a character vector specifying gene or variant identifiers or a chromosome name, or a GRanges object specifying a genomic region. The argument sampleName is optional and allows the user to plot the aligned reads and coverage from a specific sample, located in the plotted region, when the corresponding BAM file has been linked to the object with bamFiles(). The argument flankingNt is a number of nucleotides to extend the plotting region derived from the argument what. The argument showAlnNtCutoff is the region size cutoff below which it will be attempted to plot the aligned reads. The argument isPaired is passed directly to the Gviz function AlignmentsTrack() which streams over the BAM file to plot the reads and sets whether the BAM file contains single (default) or paired-end reads. Further arguments in ... are passed to the Gviz function plotTracks() and can be used to fine-tune the final plot; see the vignette of Gviz to find out what these arguments are.

reportVariants(x, type=c("shiny", "csv", "tsv"), file=NULL): Builds a report from the VariantFilteringResult object x. 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)

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)
```

VariantType-class *FiveSpliceSiteVariants and ThreeSpliceSiteVariants subclasses of VariantType*

Description

VariantType subclasses specify the type of variant to be located with the function locateVariants() from the VariantAnnotation package. Here we describe two such subclasses defined within the VariantFiltering package: FiveSpliceSiteVariants and ThreeSpliceSiteVariants. For further details on the VariantType-class please consult the corresponding manual page in the VariantAnnotation package.

Usage

```
FiveSpliceSiteVariants(minIntronLength=20L, upstream = 3L, downstream = 4L)
ThreeSpliceSiteVariants(minIntronLength=20L, upstream = 18L, downstream = 3L)
```

Arguments

minIntronLength	Minimum intron length in nucleotides, by default minIntronLength=20L. Splice sites annotated within introns smaller than this length will not be used to annotate variants.
upstream, downstream	Single integer values representing the number of nucleotides upstream and downstream of the 5' and 3' dinucleotide ends of introns. By default, the constructor function FiveSpliceSiteVariants has values upstream=3L and downstream=4L, corresponding to 3 nucleotides upstream and 4 nucleotides downstream of the first dinucleotide of the intron, while ThreeSpliceSiteVariants has values upstream=18L and downstream=3L, corresponding to 18 nucleotides upstream and 4 nucleotides downstream of the last dinucleotide of the intron.

Details

The subclasses FiveSpliceSiteVariants and ThreeSpliceSiteVariants can be used as in the region argument to the locateVariants() function from the VariantAnnotation package or as values in the regionAnnotations argument to the VariantFilteringParam() function that constructs a VariantFilteringParam-class object. These two subclasses allow the package VariantFiltering and the function locateVariants() in the VariantAnnotation package, to annotate variants located within a splice site region defined by the user. The boundaries of this region as well as the minimum intron length can be defined with the following arguments to the constructor functions:

Accessors

In the following code, x is a PromoterVariants or a AllVariants object.

- minIntronLength(x), minIntronLength(x) <- value: Gets or sets the minimum number of nucleotides long of introns to be considered in the annotation of splice sites.
- upstream(x), upstream(x) <- value: Gets or sets the number of nucleotides upstream of the 5' and 3' end of the intron, excluding the first and last dinucleotide, respectively.
- downstream(x), downstream(x) <- value: Gets or sets the number of nucleotides downstream of the 5' and 3' end of the intron, excluding the first and last dinucleotide, respectively.

Author(s)

Robert Castelo <robert.castelo@upf.edu>

See Also

- The VariantType-class man page in the VariantAnnotation package.

Examples

```
FiveSpliceSiteVariants()
ThreeSpliceSiteVariants()
```

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'
wmName(x)
## S4 method for signature 'WeightMatrix'
wmFilename(x)
## S4 method for signature 'WeightMatrix'
wmLocations(x)
## S4 method for signature 'WeightMatrix'
wmStrictLocations(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)
## S4 method for signature 'character,DNAStringSet'
wmScore(object, dnaseqs, locations, strictLocations)
## S4 method for signature 'character,character'
wmScore(object, dnaseqs, locations, strictLocations)
```

Arguments

x	A WeightMatrix object.
object	A WeightMatrix object or the file name of a weight matrix.
dnaseqs	Either a vector of character strings a DNAStringSet object, both of which store nucleotide sequences to be scored using the input WeightMatrix object.
locations	Character vector of the annotated locations to variants under which the weight matrix will be used for scoring binding sites. The possible values can be obtained by typing variantLocations().
strictLocations	Logical vector flagging whether the weight matrix should be scoring binding sites strictly within the boundaries of the given locations.

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 that is specifically designed to enable the storage of weights that may depend on the occurrence of nucleotides at other positions on the matrix.

Next to this specific weight matrix format, the function readWm() can also read the MEME motif format specified at http://meme-suite.org/doc/meme-format.html. Under this format, this function reads only currently one matrix per file and when values correspond to probabilities (specified under the motif letter-probability matrix line) they are automatically converted to weights by either using the background frequencies specified in the background letter frequencies line or using an uniform distribution otherwise.

The method wmScore() scores one or more sequences of nucleotides using the input WeightMatrix object. When the input object is the file name of a weight matrix, the function readWm() is called to read first that weight matrix and internally create a WeightMatrix object. This way to call wmScore() is required when using it in parallel since currently WeightMatrix objects are not serializable.

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 list where each element is a vector with the calculated scores of the corresponding DNA sequence. 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"),
             locations="fiveSpliceSite", strictLocations=TRUE)
```

```
wm
wmFilename(wm)
width(wm)
wmName(wm)
wmLocations(wm)
wmStrictLocations(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,
                                        svparam=ScanVcfParam(),
                                use=c("everything", "complete.obs", "all.obs"),
                                        BPPARAM=bpparam("SerialParam"))
```

Arguments

`param`	A `VariantFilteringParam` object built from a multisample VCF file with at least one affected individual and zero or more unaffected ones, and from a PED file specyfing the family relationships among individuals as well as their gender and phenotype status (affected or unaffected).
`svparam`	An instance of a `ScanVcfParam` object to enable analyzing a subset of variants and samples. This object is passed internally to a call to the `readVcf()` function in the `VariantAnnotation` package, see its help page for a complete description of this functionality.
`use`	character string specifying the policy to apply on missing genotypes when comparing them. This policy can be either '"everything"' (default), '"complete.obs"' or '"all.obs"'. The default policy ('"everything"') will propagate NA truth values using the behavior of the R logical operators, with the exception that when the final truth value associated with a variant is `NA`, the variant is ultimately discarded.
`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 built from an input multisample VCF file, along with a PED file.

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 unrelatedIndividuals VariantFilteringResults

Examples

```
## Not run:

## This actually won't run b/c in this trio de descendant is a female
CEUvcf <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.vcf.bgz")
CEUped <- file.path(system.file("extdata", package="VariantFiltering"), "CEUtrio.ped")
param <- VariantFilteringParam(vcfFilename=CEUvcf,
                    pedFilename=CEUped)
xlid <- xLinked(param)
xlid

## End(Not run)
```

Index