Package ‘TVTB’

December 22, 2016

Type Package

Title TVTB: The VCF Tool Box

Version 1.0.2

Date 2016-10-23

Description The package provides S4 classes and methods to filter, summarise and visualise genetic variation data stored in VCF files. In particular, the package extends the FilterRules class (S4Vectors package) to define new classes of filter rules applicable to the various slots of VCF objects. functionalities are integrated and demonstrated in a Shiny web-application, the Shiny Variant Explorer (tSVE).

License Artistic-2.0

Depends R (>= 3.3), methods, utils, stats

Imports BiocGenerics (>= 0.19.1), BiocParallel, Biostrings, ensembldb, ensemblVEP, GenomeInfoDb, GenomicRanges, ggplot2, IRanges (>= 2.7.1), reshape2, Rsamtools, S4Vectors (>= 0.11.11), SummarizedExperiment, VariantAnnotation (>= 1.19.9)

Suggests EnsDb.Hsapiens.v75 (>= 0.99.7), shiny (>= 0.13.2.9005), DT (>= 0.1.67), rtracklayer, BiocStyle (>= 2.1.23), knitr (>= 1.12), rmarkdown, testthat, covr

biocViews Software, Genetics, GeneticVariability, GenomicVariation, DataRepresentation, GUI, Genetics, DNASEq, WholeGenome, Visualization, MultipleComparison, DataImport, VariantAnnotation, Sequencing, Coverage, Alignment, SequenceMatching


VignetteBuilder knitr

URL https://github.com/kevinrue/TVTB

BugReports https://github.com/kevinrue/TVTB/issues
NeedsCompilation: no

Author: Kevin Rue-Albrecht [aut, cre]
Maintainer: Kevin Rue-Albrecht <kevinrue67@gmail.com>

R topics documented:

TVTB-package .......................................................... 2
addCountGenos-methods ............................................. 3
addFrequencies-methods ........................................... 5
addOverallFrequencies-methods ................................. 7
addPhenoLevelFrequencies-methods ......................... 8
countGenos-methods ............................................... 9
dropInfo-methods ................................................ 11
Genotypes-class .................................................... 12
readVcf-methods .................................................. 14
tSVE ................................................................. 15
TVTBparam-class ..................................................... 16
variantsInSamples-methods ..................................... 18
VcfBasicRules-class ............................................... 19
VcfFilterRules-class .............................................. 22
vepInPhenoLevel-methods ....................................... 25

Index ................................................................. 28

TVTB-package  

TVTB: The VCF Tool Box

Description

The package provides S4 classes and methods to filter, summarise and visualise genetic variation data stored in VCF files. In particular, the package extends the FilterRules class (S4Vectors package) to define new classes of filter rules applicable to the various slots of VCF objects. Functionalities are integrated and demonstrated in a Shiny web-application, the Shiny Variant Explorer (tSVE).

Details

Package: TVTB
Type: Package
Title: TVTB: The VCF Tool Box
Version: 1.0.2
Date: 2016-10-23
Authors@R: person("Kevin", "Rue-Albrecht", role = c("aut", "cre"), email = "kevinrue67@gmail.com")
Description: The package provides S4 classes and methods to filter, summarise and visualise genetic variation data stored in VCF files. In particular, the package extends the FilterRules class (S4Vectors package) to define new classes of filter rules applicable to the various slots of VCF objects. Functionalities are integrated and demonstrated in a Shiny web-application, the Shiny Variant Explorer (tSVE).
License: Artistic-2.0
Depends: R (>= 3.3), methods, utils, stats
Imports: BiocGenerics (>= 0.19.1), BiocParallel, Biostrings, ensemblpdb, ensemblVEP, GenomeInfoDb, GenomicRanges, ggplot2, IRanges (>= 2.7.1), reshape2, Rsamtools, S4Vectors (>= 0.11.11), SummarizedExperiment, VariantAnnotation (>= 1.19.9)
Suggests: EnsDb.Hsapiens.v75 (>= 0.99.7), shiny (>= 0.13.2.9005), rtracklayer, BiocStyle (>= 2.1.23), knitr (>= 1.12), rmarkdown, testthat, covr
biocViews: Software, Genetics, GeneticVariability, GenomicVariation, DataRepresentation, GUI, Genetics, DNASeq, WholeGenome, Visualization, MultipleComparison, DataImport, VariantAnnotation, Sequencing, Coverage, Alignment, SequenceMatching
Collate: tSVE.R AllClasses.R AllGenerics.R Genotypes-class.R TVTBparam-class.R VcfFilterRules-class.R ...
**AddCountGenos-methods**

**Description**

Adds the total occurrences of a set of genotypes as an INFO field for each variant. All given genotypes are counted toward a single total (*e.g.* grand total of \(<0/0\), \(<0/1\>)\)), while other genotypes are silently ignored.
addCountGenos-methods

Usage

## S4 method for signature 'ExpandedVCF'
addCountGenos(
  vcf, genos, key, description,
  samples = 1:ncol(vcf), force = FALSE)

Arguments

vcf ExpandedVCF object.
genos character vector of genotypes to count (toward a common unique total).
key Name of the INFO field to create or update (character vector of length 1). See Details below.
description character description of the INFO field to create or overwrite (character vector of length 1).
samples integer, numeric or character vector indicating samples to consider in VariantAnnotation::geno(vcf).
If not specified, all samples are considered.
force If TRUE, the field header and data will be overwritten if present; If FALSE, an error is thrown if the field already exists.

Details

In all cases, the new INFO field is inserted after the last existing field. In other words, overwriting an existing INFO field is achieved by dropping it from the data and header of the info slot, and subsequently inserting the new data after the last remaining INFO field.

Value

ExpandedVCF object including an additional INFO field stating the count of genotypes.

Author(s)

Kevin Rue-Albrecht

See Also

countGenos, ExpandedVCF-method and geno, VCF-method

Examples

# Example data ----
# VCF file
vcfFile <- system.file("extdata", "moderate.vcf", package = "TVTB")

# TVTB parameters
tparam <- TVTBparam(Genotypes(ref = "0|0", het = c("0|1", "1|0"), alt = "1|1"))

# Pre-process variants
vcf <- VariantAnnotation::readVcf(vcfFile, param = tparam)
vcf <- VariantAnnotation::expand(vcf, row.names = TRUE)

# Example usage ----
vcf <- addCountGenos(
    vcf, het(tparam),
    suffix(tparam)["het"],
    "Number of heterozygous genotypes")

### addFrequencies-methods

**Group-level genotypes counts and allele frequencies**

**Description**

Adds genotypes counts (reference homozygote, heterozygote, and alternate homozygote) and allele frequencies (alternate and minor) as INFO fields in an ExpandedVCF object. Counts and frequencies may be calculated overall (i.e. across all samples), or within groups of samples (i.e. within phenotype levels). Multiple genotypes can be counted toward a single frequency (e.g. combined `c("0/0", "0|0")` for homozygote reference genotypes).

**Usage**

```r
## S4 method for signature 'ExpandedVCF,list'
addFrequencies(vcf, phenos, force = FALSE)
## S4 method for signature 'ExpandedVCF,character'
addFrequencies(vcf, phenos, force = FALSE)
## S4 method for signature 'ExpandedVCF,missing'
addFrequencies(vcf, force = FALSE)
```

**Arguments**

- **vcf**
  - ExpandedVCF object.
  - `metadata(vcf)["TVTBparam"]` must contain a `TVTBparam` object.

- **phenos**
  - If `NULL`, counts and frequencies are calculated across all samples.
  - Otherwise, either a character vector of phenotypes in `colnames(colData(vcf))`, or a named list in which names are phenotypes in `colnames(colData(vcf))` and values are character vectors of phenotype levels in `colData(vcf)[,phenotype]`. See Details below.

- **force**
  - If `TRUE`, INFO fields header and data are overwritten with a message, if present.
  - If `FALSE`, an error is thrown if any field already exists.

**Details**

The `phenos` argument is central to control the behaviour of this method.

If `phenos=NULL`, genotypes and frequencies are calculated across all the samples in the ExpandedVCF object, and stored in INFO fields named according to settings stored in the `TVTBparam` object (see below).

If `phenos` is a character vector of phenotypes present in `colnames(colData(vcf))`, counts and frequencies are calculated for each level of those phenotypes, and stored in INFO fields prefixed with "<phenotype>_<level>" and suffixed with the settings stored in the `param` object (see below).
Finally, if `phenos` is a named list, names must be phenotypes present in `colnames(colData(vcf))`, and values must be levels of those phenotypes. In this case, counts and frequencies are calculated for the given levels of the given phenotypes, and stored in INFO fields as described above.

The `param` object controls the key (suffix) of INFO fields as follows:

- `names(ref(param))` Count of reference homozygote genotypes.
- `names(het(param))` Count of heterozygote genotypes.
- `names(alt(param))` Count of alternate homozygote genotypes.
- `aaf(param)` Alternate allele frequency.
- `maf(param)` Minor allele frequency

**Value**

ExpandedVCF object including additional INFO fields for genotype counts and allele frequencies. See Details.

**Author(s)**

Kevin Rue-Albrecht

**See Also**

`addOverallFrequencies`, `ExpandedVCF-method`, `addPhenoLevelFrequencies`, `ExpandedVCF-method`, `VCF`, and `TVTBparam`.

**Examples**

```r
# Example data ----
# VCF file
vcfFile <- system.file("extdata", "moderate.vcf", package = "TVTB")

# Phenotype file
phenoFile <- system.file("extdata", "moderate_pheno.txt", package = "TVTB")
phenotypes <- S4Vectors::DataFrame(read.table(phenoFile, TRUE, row.names = 1))

# TVTB parameters
tparam <- TVTBparam(Genotypes("0|0", c("0|1", "1|0"), "1|1"))

# Pre-process variants
vcf <- VariantAnnotation::readVcf(
  vcfFile, param = tparam, colData = phenotypes)
vcf <- VariantAnnotation::expand(vcf, row.names = TRUE)

# Example usage ----
vcf <- addFrequencies(vcf, list(super_pop = "AFR"))
```
Description

Adds dataset-wide genotypes counts (reference homozygote, heterozygote, and alternate homozygote) and allele frequencies (alternate and minor) as INFO fields in an ExpandedVCF object. Counts and frequencies may be calculated across all samples. Multiple genotypes can be counted toward a single frequency (e.g. combined c("0/0", "0|0") for homozygote reference genotypes).

Usage

```r
## S4 method for signature 'ExpandedVCF'
addOverallFrequencies(vcf, force = FALSE)
```

Arguments

- `vcf` ExpandedVCF object.
  - metadata(vcf)[["TVTparam"]]) must contain a TVTParam object.
- `force` If TRUE, INFO fields header and data are overwritten. If FALSE, an error is thrown if any field already exists.

Details

Genotypes and frequencies are calculated across all the samples in the ExpandedVCF object, and stored in INFO fields named according to settings stored in the TVTParam object (see below).

The `param` object controls the key of INFO fields as follows:

- `names(ref(param))` Count of reference homozygote genotypes.
- `names(het(param))` Count of heterozygote genotypes.
- `names(alt(param))` Count of alternate homozygote genotypes.
- `aaf(param)` Alternate allele frequency.
- `maf(param)` Minor allele frequency

Value

ExpandedVCF object including additional INFO fields for genotype counts and allele frequencies. See Details.

Author(s)

Kevin Rue-Albrecht

See Also

`addFrequencies, ExpandedVCF, list-method, addPhenoLevelFrequencies, ExpandedVCF-method`, and `VCF`.
Examples

```r
# Example data ----

# VCF file
vcfFile <- system.file("extdata", "moderate.vcf", package = "TVTB")

# TVTB parameters
tparam <- TVTBparam(Genotypes("0|0", c("0|1", "1|0"), "1|1"))

# Pre-process variants
vcf <- VariantAnnotation::readVcf(vcfFile, param = tparam)
vcf <- VariantAnnotation::expand(vcf, row.names = TRUE)

# Example usage ----
vcf <- addOverallFrequencies(vcf, tparam)
```

Description

Adds genotypes counts (reference homozygote, heterozygote, and alternate homozygote) and allele frequencies (alternate and minor) calculated in a group of samples associated with a given level of a given phenotype as INFO fields in an ExpandedVCF object. Multiple genotypes can be counted toward a single frequency (e.g. combined c("0/0", "0|0") for homozygote reference genotypes).

Usage

```r
## S4 method for signature 'ExpandedVCF'
addPhenoLevelFrequencies(
  vcf, pheno, level, force = FALSE)
```

Arguments

- **vcf**: ExpandedVCF object. `metadata(vcf)["TVTBparam"]` must contain a `TVTBparam` object.
- **pheno**: Phenotype in `colnames(colData(vcf))`.
- **level**: Phenotype level in `colData(vcf)[,pheno]`.
- **force**: If TRUE, INFO fields header and data are overwritten. If FALSE, an error is thrown if any field already exists.

Details

Genotypes and frequencies are calculated within the groups of samples associated with the given level of the given phenotype, and stored in INFO fields named according to settings stored in `metadata(vcf)["TVTBparam"]` (see below).

The `TVTBparam` object controls the key suffix of INFO fields as follows:
countGenos-methods

Names and counts of genotypes:

- `names(ref(param))` Count of reference homozygote genotypes.
- `names(het(param))` Count of heterozygote genotypes.
- `names(alt(param))` Count of alternate homozygote genotypes.
- `aaf(param)` Alternate allele frequency.
- `maf(param)` Minor allele frequency

Value

ExpandedVCF object including additional INFO fields for genotype counts and allele frequencies. See Details.

Author(s)

Kevin Rue-Albrecht

See Also

- `addFrequencies`, `ExpandedVCF`, `list-method`, `addOverallFrequencies`, `ExpandedVCF-method`, `VCF`, and `TVTBparam`.

Examples

```r
# Example data ----

# VCF file
cvfFile <- system.file("extdata", "moderate.vcf", package = "TVTB")

# Phenotype file
dataFile <- system.file("extdata", "moderate_pheno.txt", package = "TVTB")
phenotypes <- S4Vectors::DataFrame(read.table(dataFile, TRUE, row.names = 1))

# TVTB parameters
tparam <- TVTBparam(Genotypes("0|0", c("0|1", "1|0"), "1|1"))

# Pre-process variants
cvf <- VariantAnnotation::readVcf(
  vcffile, param = tparam, colData = phenotypes)
cvf <- VariantAnnotation::expand(vcf, row.names = TRUE)

# Example usage ----

cvf <- addPhenoLevelFrequencies(vcf, "super_pop", "AFR")
```

**Description**

Counts the total occurrences of a set of genotypes by row in a matrix of genotype. All given genotypes are counted toward a single total (e.g. grand total of `c("0/0", "0|0")`), while other genotypes are silently ignored.
countGenos-methods

Usage

## S4 method for signature 'ExpandedVCF'
countGenos(
  x, genos, pheno = NULL, level = NULL
)

Arguments

x ExpandedVCF object.

genos character vector of genotypes to count (toward a common unique total).

pheno If x is an ExpandedVCF object, phenotype in colnames(colData(x)).

level If x is an ExpandedVCF object, phenotype level in colData(x)[, pheno].

Value

An integer vector representing the aggregated count of the given genotypes in each row.

Author(s)

Kevin Rue-Albrecht

See Also

VCF

Examples

# Example data ----

# VCF file
vcfFile <- system.file("extdata", "moderate.vcf", package = "TVTB")

# Phenotype file
phenoFile <- system.file("extdata", "moderate_pheno.txt", package = "TVTB")
phenotypes <- S4Vectors::DataFrame(read.table(phenoFile, TRUE, row.names = 1))

# TVTB parameters
tparam <- TVTBparam(Genotypes("0|0", c("0|1", "1|0"), "1|1"))

# Pre-process variants
vcf <- VariantAnnotation::readVcf(
  vcfFile, param = tparam, colData = phenotypes)
vcf <- VariantAnnotation::expand(vcf, row.names = TRUE)

# Example usage ----

countGenos(vcf, het(tparam), "super_pop", "AFR")
dropInfo-methods

Remove INFO keys from VCF objects

Description

Given a character vector of INFO keys, removes either the associated header, data, or both from a VCF object. If no INFO key is given (the default), all INFO keys are checked and removed from the given slot if they do not have a matching entry in the other slot.

Usage

```r
## S4 method for signature 'VCF'
dropInfo(
  vcf, key = NULL, slot = "both")
```

Arguments

- `vcf`: VCF object.
- `key`: character vector of INFO keys to remove.
  If NULL (the default), all keys are checked, and removed from the given slot if they do not have a matching entry in the other slot.
- `slot`: Should the INFO keys be removed from the "header", the "data", or "both" (the default)?

Value

An integer vector representing the aggregated count of the given genotypes in each row.

Note

In the future, \( x \) should also support genotype quality (GQ) to consider only genotypes above a given quality cut-off.

Author(s)

Kevin Rue-Albrecht

See Also

VCF

Examples

```r
# Example data ----

# VCF file
vcfFile <- system.file("extdata", "moderate.vcf", package = "TVTB")

# TVTB parameters
tparam <- TVTBparam(Genotypes("0|0", c("0|1", "1|0"), "1|1"))

# Pre-process variants
```
vcf <- VariantAnnotation::readVcf(
  vcfFile, param = tparam)
vcf <- VariantAnnotation::expand(vcf, row.names = TRUE)

# Example usage ----
dropInfo(vcf)
dropInfo(vcf, "CSQ")

---

Genotypes-class

Genotypes class objects

Description

The Genotypes class stores genotype definitions in a convenient format.

Usage

```r
## S4 method for signature 'character,character,character'
Genotypes(
  ref, het, alt, suffix = c(ref="REF", het="HET", alt="ALT"))
```

Arguments

- **ref**: A character vector declaring the encoding of homozygote reference genotypes.
- **het**: A character vector declaring the encoding of heterozygote genotypes.
- **alt**: A character vector declaring the encoding of homozygote alternate genotypes.
- **suffix**: Set the individual INFO key suffixes used to store the statistics of homozygote reference, heterozygote, and homozygote alternate genotypes, in this order. See Details section.

Details

For each `suffix` stored in the Genotypes object, TTB may store data in the VCF object under the INFO keys defined as follows:

- **suffix**: Statistics across all samples in the ExpandedVCF (e.g. "MAF").
- **phenotype_level_suffix**: Statistics across samples associated with a given level of a given phenotype (e.g. "gender_male_MAF").

Users are recommended to avoid using those INFO keys for other purposes.

Value

A Genotypes object that contains genotype definitions.
Accessor methods

In the following code snippets x is a Genotypes object.

ref(x), ref(x) <- value  Gets or sets the character vector that declares homozygote reference genotypes.
het(x), het(x) <- value  Gets or sets the character vector that declares heterozygote genotypes.
alt(x), alt(x) <- value  Gets or sets the character vector that declares homozygote alternate genotypes.

genos(x)  Gets a character vector of concatenated homozygote reference, heterozygote, and homozygote alternate genotypes. See also ref, het, alt, and carrier accessors.
carrier(x)  Gets a character vectors of concatenated heterozygote and homozygote alternate genotypes. See also het and alt accessors.
suffix(x)  Gets a named character vector that declares individual suffixes used to store the data for each set of genotypes in the INFO field of the VCF object. Names of this vector are ref, het, and alt.

Author(s)

Kevin Rue-Albrecht

See Also

VCF, TVTbparam, and addCountGenos-methods.

Examples

# Constructors ----
genotypes <- Genotypes("0|0", c("0|1", "1|0"), "1|1")

# Accessors ----

## Concatenated homozygote reference, heterozygote, and alternate heterozygote genotypes stored in the Genotypes object returned by the genos() accessor.
genos(genotypes)

## Individual genotypes can be extracted with ref(), het(), alt() accessors.
ref(genotypes)
het(genotypes)
altn(genotypes)

## Their individual INFO key suffixes can be extracted with suffix() accessor
## and the relevant name
suffix(genotypes)
suffix(genotypes)["ref"]
suffix(genotypes)["het"]
suffix(genotypes)["alt"]

## Concatenated heterozygote, and alternate heterozygote genotypes are returned by the carrier() accessor.
carrier(genotypes)
names(carrier(genotypes))
Description

Read Variant Call Format (VCF) files and attaches optional phenotype information.

Usage

```r
## S4 method for signature 'character,TVTParam'
readVcf(
  file, genome, param, ..., colData = DataFrame())
## S4 method for signature 'TabixFile,TVTParam'
readVcf(
  file, genome, param, ..., colData = DataFrame())
```

Arguments

- `file`, `genome`: See `readVcf`
- `param`: TVTParam object that contains recurrent parameters. The vep slot of `param` is checked for presence among the INFO keys of the VCF file. The TVTParam object is coerced to ScanVcfParam using the ranges slot only. All fixed, info, and geno fields are imported (see argument colData to declare samples to import).
- `...`: Additional arguments, passed to methods.
- `colData`: Phenotype information in a DataFrame. If supplied, only samples identifiers present in `rownames(colData)` are imported from the VCF file. An error is thrown if any of the samples is absent from the VCF file.

Value

VCF object. See `?VCF` for complete details of the class structure.

Author(s)

Kevin Rue-Albrecht

See Also

`readVcf`, `TabixFile`, `ScanVcfParam-method`, and `VCF`.

Examples

```r
# Example data ----

# VCF file
vcfFile <- system.file("extdata", "moderate.vcf.gz", package = "TVTB")

# Phenotype file
phenoFile <- system.file("extdata", "moderate.pheno.txt", package = "TVTB")
```
The Shiny Variant Explorer (tSVE) web-application

**Description**

Currently unsupported — Package undergoing major updates.

This function starts the interactive tSVE shiny web-application that allows to interactively load and visualise genetic variants and their Ensembl Variant Effect Predictor (VEP) predictions using the package methods.

**Usage**

```r
tSVE(...)```

**Arguments**

... Additional arguments passed to the `runApp` function from the `shiny` package.

**Value**

Not applicable (yet).

**Author(s)**

Kevin Rue-Albrecht

**References**

Interface to EnsDb adapted from ensembldb.

**See Also**

`runEnsDbApp`.

**Examples**

```r
if (interactive()){
  runEnsDbApp()
}
```
TVTBparam-class

TVTBparam class objects

Description

The TVTBparam class stores recurrent parameters of the TVTB package in a convenient format.

Usage

```r
## S4 method for signature 'Genotypes'
TVTBparam(
genos, ranges = GRangesList(),
aaf = "AAF", maf = "MAF", vep = "CSQ", bp = SerialParam(),
svp = ScanVcfParam(which = reduce(unlist(ranges))))
```

Arguments

- **genos**: A Genotypes object that declares the three sets of homozygote reference, heterozygote, and homozygote alternate genotypes, as well as the individual key suffix used to store data for each set of genotypes in the info slot of a VCF object. See also Details section.
- **ranges**: A GRangesList of genomic regions. See ssvp argument. In the future, may be used to facet statistics and figures.
- **aaf**: INFO key suffix used to store the alternate allele frequency (AAF).
- **maf**: INFO key suffix used to store the minor allele frequency (MAF).
- **vep**: INFO key suffix used to extract the VEP predictions. See ssvp argument.
- **bp**: A BiocParallelParam object.
- **svp**: A ScanVcfParam object. If none is supplied, the ScanVcfParam slot which is automatically set to reduce(unlist(ranges)).

Details

For each suffix stored in the TVTBparam object, TVTB may store data in the VCF object under the INFO keys defined as follows:

- **suffix**: Statistics across all samples in the ExpandedVCF (e.g. "MAF").
- **phenotype_level_suffix**: Statistics across samples associated with a given level of a given phenotype (e.g. "gender_male_MAF").

Users are recommended to avoid using those INFO keys for other purposes.

Value

A TVTBparam object that contains recurrent parameters.
Accessor methods

In the following code snippets, x is a TTVBparam object.

```r
genos(x), genos(x) <- value  # Gets or sets the Genotypes object stored in the genos slot.
ranges(x), ranges(x) <- value  # List of genomic ranges to group variants during analyses and plots.
ref(x), ref(x) <- value  # Gets or sets the character vector that declares homozygote reference genotypes.
het(x), het(x) <- value  # Gets or sets the character vector that declares heterozygote genotypes.
alt(x), alt(x) <- value  # Gets or sets the character vector that declares homozygote alternate genotypes.
carrier(x)  # Gets a character vector of concatenated heterozygote and homozygote alternate genotypes. See also het and alt accessors.
aaf(x), aaf(x) <- value  # Gets or sets the INFO key suffix used to store the alternate allele frequency (AAF).
maf(x), maf(x) <- value  # Gets or sets the INFO key suffix used to store the minor allele frequency (MAF).
vep(x), maf(x) <- value  # Gets or sets the INFO key suffix used to extract the VEP predictions.
bp(x), bp(x) <- value  # Gets or sets the BiocParallel parameters.
suffix(x)  # Gets a named character vector that declares individual suffixes used to store the data for each set of genotypes in the INFO field of the VCF object. Names of this vector are ref, het, alt, aaf, and maf.
svp(x), svp(x) <- value  # Gets or sets the ScanVcfParam parameters.
```

Author(s)

Kevin Rue-Albrecht

See Also

Genotypes, VCF, ExpandedVCF, addCountGenos-methods vepInPhenoLevel-methods, variantsInSamples-methods, and BiocParallelParam.

Examples

```r
# Constructors ----
grl <- GenomicRanges::GRangesList(GenomicRanges::GRanges("15", IRanges::IRanges(48413170, 48434757, names = "SLC24A5")))

tparam <- TTVBparam(Genotypes("0|0", c("0|1", "1|0"), "1|1"), ranges = grl)

# Accessors ----

## The Genotypes object stored in the genos slot of the TTVBparam object
## return by the genos() accessor.
genos(tparam)
```
## Genomic ranges stored in the TVTBparam object returned by the ranges() accessor.

ranges(tparam)

## Individual genotypes can be extracted with ref(), het(), alt() accessors.

ref(tparam)
het(tparam)
alt(tparam)

## Their individual INFO key suffixes can be extracted with suffix() applied to the above accessors.

suffix(tparam)
suffix(tparam)["ref"]
suffix(tparam)["het"]
suffix(tparam)["alt"]
 suffix(tparam)["aaf"]
suffix(tparam)["maf"]

## Heterozygote, and alternate heterozygote genotypes are returned by the carrier() accessor.

carrier(tparam)

## INFO key suffix of alternate/minor allele frequency returned by the aaf() and maf() accessors.

aaf(tparam)
maf(tparam)

## INFO key suffix of the VEP predictions returned by the vep() accessor.

vep(tparam)

## BiocParallel parameters

bp(tparam)

## ScanVcfParam parameters

svp(tparam)

---

variantsInSamples-methods

### Identify variants observed in samples

**Description**

Identifies variants observed (uniquely) in at least one sample of a given group.

**Usage**

```r
## S4 method for signature 'ExpandedVCF'
variantsInSamples(
  vcf, samples = 1:ncol(vcf), unique = FALSE)
```

**Arguments**

- `vcf`  
  ExpandedVCF object. `metadata(vcf)["TVTBparam"]` must contain a TVTBparam object.
VcfBasicRules-class

samples integer, numeric or character vector indicating samples to consider in VariantAnnotation::geno(vcf).
If not specified, all samples are considered.

unique If TRUE, consider only variants unique to the phenotype level (i.e. not seen in any other phenotype level).

Value
An integer vector of indices indicating which variants are observed in at least one non-reference genotype in the given group of samples.

Author(s)
Kevin Rue-Albrecht

See Also
VCF and TVTBparam.

Examples
# Example data ----
# VCF file
vcfFile <- system.file("extdata", "moderate.vcf", package = "TVTB")

# Phenotype file
phenoFile <- system.file("extdata", "moderate_pheno.txt", package = "TVTB")
phenotypes <- S4Vectors::DataFrame(
  read.table(file = phenoFile, header = TRUE, row.names = 1))

# TVTB parameters
tparam <- TVTBparam(Genotypes("0|0", c("0|1", "1|0"), "1|1"))

# Pre-process variants
vcf <- VariantAnnotation::readVcf(
  vcfFile, param = tparam, colData = phenotypes)
vcf <- VariantAnnotation::expand(vcf, row.names = TRUE)

# Example usage ----
variantsInSamples(
  vcf,
  which(SummarizedExperiment::colData(vcf)[,"super_pop"] == "EUR"))
Details

All arguments are first passed to S4Vectors::FilterRules before re-typing the resulting as a VcfFixedRules, VcfInfoRules, or VcfVepRules class.

Accessor methods

In the following code snippets x is an object from any of the classes described in this help page, except when specified otherwise.

active(x), active(x)<- Gets or sets the active state of each filter rule in x. Inherited from FilterRules

vep(x), vep(x)<- Gets or sets the INFO key where the Ensembl VEP predictions to use for filtering are stored. Returns NA_character_ for filters not applicable to VEP predictions.

type(x) Returns "filter" (linkS4class(FilterRules)), "fixed" (linkS4class(VcfFixedRules)), "info" (linkS4class(VcfInfoRules)), or "vep" (linkS4class(VcfVepRules)) as a character vector of length(x).

Constructors

VcfFixedRules(exprs = list(), ..., active = TRUE)
VcfInfoRules(exprs = list(), ..., active = TRUE)
VcfVepRules(exprs = list(), ..., active = TRUE, vep = "CSQ")

All methods construct an object of the corresponding class with the rules given in the list exprs or in .... The initial active state of the rules is given by active, which is recycled as necessary.

See the constructor of FilterRules for more details.

Subsetting and Replacement

In the following code snippets x and value are objects from any of the classes described in this help page.

x[i]: Subsets the filter rules using the same interface as for List.

x[[i]]: Extracts an expression or function via the same interface as for List.

x[i] <- value: Replaces a filter rule by one of the same class. The active state(s) and name(s) are transferred from value to x.

x[[i]] <- value: The same interface as for List. The default active state for new rules is TRUE.

Combining

In the following code snippets x, values, and ... are objects from any of the classes described in this help page, or VcfFilterRules.

append(x, values, after = length(x)): Appends the values onto x at the index given by after.

c(x, ...,): Concatenates the filters objects in ... onto the end of x.

Note that combining rules of different types (e.g. VcfFixedRules and VcfVepRules) produces a VcfFilterRules object.
Evaluating

As described in the S4Vectors documentation:

\[
eval(expr, envir, enclos): \text{Evaluates a rule instance (passed as the expr argument) in their respective context of a VCF object (passed as the envir argument). } i.e.: \\
\cdot \text{VcfFixedRules: fixed(envir)} \\
\cdot \text{VcfInfoRules: info(envir)} \\
\cdot \text{VcfVepRules: mcols(parseCSQToGRanges(envir, ...))} \\
\cdot \text{FilterRules: envir}
\]

\[
evalSeparately(expr, envir, enclos): \\
\text{subsetByFilter(x, filter)} \\
\text{summary(object)} \\
\text{See eval,FilterRules,ANY-method for details.}
\]

Author(s)

Kevin Rue-Albrecht

See Also

FilterRules, VcfFilterRules, and VCF.

Examples

# Constructors ----

fixedRules <- VcfFixedRules(list(
  pass = expression(FILTER == "PASS"),
  qual = expression(QUAL > 20)
))

fixedRules

infoRules <- VcfInfoRules(list(
  common = expression(MAF > 0.01), # minor allele frequency
  alt = expression(ALT > 0) # count of alternative homozygotes
))

infoRules

vepRules <- VcfVepRules(list(
  missense = expression(Consequence %in% c("missense_variant")),
  CADD = expression(CADD_PHRED > 15)
))

vepRules

filterRules <- S4Vectors::FilterRules(list(
  PASS = function(x) fixed(x)$FILTER == "PASS",
  COMMON = function(x) info(x)$MAF > 0.05
))

filterRules

# Accessors ----

## get/set the active state directly

S4Vectors::active(infoRules)
VcfFilterRules-class

VcfFilterRules-class objects
Description

The VcfFilterRules class can store multiple types of filters applicable to various slots of VCF objects.

Details

All arguments must be VcfFixedRules, VcfInfoRules, VcfVepRules, VcfFilterRules objects.

Accessor methods

In the following code snippets x is a VcfFilterRules object.

active(x), active(x)<- Get or set the active state of each filter rule in x. Inherited from FilterRules

vep(x), vep(x)<- Gets or sets the INFO key where the Ensembl VEP predictions to use for filtering are stored.

type(x) Gets the type of each filter stored in a VcfFilterRules object. Read-only.

Constructors

VcfFilterRules(...) constructs an VcfFilterRules object from VcfFixedRules, VcfInfoRules, VcfVepRules, and VcfFilterRules objects in ....

Subsetting and Replacement

In the code snippets below, x is a VcfFilterRules object.

x[i, drop = TRUE]: Subsets the filter rules using the same interface as for Vector. If all filter rules are of the same type and drop=TRUE (default), the resulting object is re-typed to the most specialised class, if possible. In other words, if all remaining filter rules are of type "vep", the object will be type as VcfVepRules.

x[[i]]: Extracts an expression or function via the same interface as for List.

x[i] <- value: Replaces a filter rule by one of any valid class (VcfFixedRules, VcfInfoRules, VcfVepRules, or VcfFilterRules). The active state(s), name(s), and type(s) (if applicable) are transferred from value.

x[[i]] <- value: The same interface as for List. The default active state for new rules is TRUE.

Combining

In the following code snippets x is an object of class VcfFilterRules, while values and ... are objects from any of the classes VcfFixedRules, VcfInfoRules, VcfVepRules, or VcfFilterRules:

append(x, values, after = length(x)): Appends the values onto x at the index given by after.

c(x, ...): Concatenates the filters objects in ... onto the end of x.
Evaluating

As described in the S4Vectors documentation:

\[
\text{eval}(\text{expr, envir, enclos}) \text{ Evaluates each active rule in a VcfFilterRules instance (passed as the expr argument) in their respective context of a VCF object (passed as the envir argument).}
\]

\[
\text{evalSeparately}(\text{expr, envir, enclos}): \\
\text{subsetByFilter}(x, \text{filter}) \\
\text{summary}(\text{object})
\]

See \text{eval,FilterRules,ANY-method} for details.

Author(s)

Kevin Rue-Albrecht

See Also

\text{FilterRules, VcfFixedRules, VcfInfoRules, VcfVepRules}, and \text{VCF}.

Examples

# Constructors ----

\[
\text{fixedR} <- \text{VcfFixedRules(\{list}(
\text{pass} = \text{expression(FILTER == "PASS"),}
\text{qual} = \text{expression(QUAL > 20)}
\})
\])
\]

\[
\text{fixedR}
\]

\[
\text{infoR} <- \text{VcfInfoRules(\{list}(
\text{common} = \text{expression(MAF > 0.1), # minor allele frequency}
\text{present} = \text{expression(ALT + HET > 0) # count of non-REF homozygotes}
\})
\])
\]

# ...is synonym to...

\[
\text{infoR} <- \text{VcfInfoRules(\{list}(
\text{common} = \text{expression(MAF > 0.1), # minor allele frequency}
\text{present} = \text{expression(ALT > 0 || HET > 0)}
\})
\])
\]

\[
\text{infoR}
\]

\[
\text{vepR} <- \text{VcfVepRules(\{list}(
\text{missense} = \text{expression(Consequence %in% c("missense_variant")),}
\text{CADD} = \text{expression(CADD_PHRED > 15)}
\})
\])
\]

\[
\text{vepR}
\]

\[
\text{vcfRules} <- \text{VcfFilterRules(fixedR, infoR, vepR)}
\]

\[
\text{vcfRules}
\]

# Accessors ----

# Type of each filter stored in the VcfFilterRules object
\[
\text{type(vcfRules)}
\]
# Example data ----

# VCF file
vcfFile <- system.file("extdata", "moderate.vcf", package = "TVTB")

# TVTB parameters
tparam <- TVTBparam(Genotypes("0|0", c("0|1", "1|0"), "1|1"))

# Pre-process variants
vcf <- VariantAnnotation::readVcf(vcfFile, param = tparam)
vcf <- VariantAnnotation::expand(vcf, row.names = TRUE)
vcf <- addOverallFrequencies(vcf, tparam)

# Applying filters to VCF objects ----

## Evaluate filters
eval(vcfRules, vcf)

## Evaluate filters separately
as.data.frame(evalSeparately(vcfRules, vcf))

# Interestingly, the only common missense variant has a lower CADD score
## Deactivate the CADD score filter
active(vcfRules)["CADD"] <- FALSE

## Subset VCF by filters (except CADD, deactivated above)
subsetByFilter(vcf, vcfRules)

# Subsetting and Replacement ----

v123 <- vcfRules[1:3]

# Extract the expression
v5expr <- vcfRules[[5]]
# Subset the object
v5obj <- vcfRules[5]

# Replace the expression (active reset to TRUE, original name retained)
v123[2] <- v5expr

# Replace the rule (active state and name transferred from v5obj)
v123[2] <- v5obj

---

vepInPhenoLevel-methods

VEP predictions of variants observed in samples

### Description

Considers only variants observed (unique) in samples associated with a given phenotype level, and tabulates the corresponding values for a given VEP prediction field.
Usage

```r
## S4 method for signature 'ExpandedVCF'
vepInPhenoLevel(
  vcf, phenoCol, level, vepCol,
  unique = FALSE, facet = NULL)
```

Arguments

- `vcf` : ExpandedVCF object. metadata(vcf)['TVTBparam'] must contain a TVTBparam object.
- `phenoCol` : Name of a column in pheno.
- `level` : Phenotype level; only variants observed in at least one sample will be considered.
- `vepCol` : VEP prediction field; Name of a metadata column in ensemblVEP::parseCSQToGRanges(vcf).
- `unique` : If TRUE, consider only variants unique to the phenotype level (i.e. absent from all other phenotype levels).
- `facet` : Name of a metadata column in ensemblVEP::parseCSQToGRanges(vcf). Additional VEP field appended as an additional column to the data.frame returned. If plot=TRUE, this field will be used to create one sub-plot for each level of the faceting field.

If available, “Feature” is a recommended value for this argument, as VEP typically produce one prediction per variant per feature.

Value

A data.frame in long format including one row for each VEP prediction associated with a variant seen in at least one sample (heterozygote or alternate homozygote) associated with the phenotype level. The data.frame contains at least one column for the VEP prediction value. An additional column containing another VEP prediction field may be added using the facet argument.

Author(s)

Kevin Rue-Albrecht

See Also

VCF, ensemblVEP, GRanges, and DataFrame.

Examples

```r
# Example data ----
# VCF file
vcfFile <- system.file("extdata", "moderate.vcf", package = "TVTB")

# Phenotype file
phenoFile <- system.file("extdata", "moderate_pheeno.txt", package = "TVTB")
phenotypes <- S4Vectors::DataFrame(read.table(file = phenoFile, header = TRUE, row.names = 1))

# TVTB parameters
tparam <- TVTBparam(Genotypes("0|0", c("0|1", "1|0"), "1|1"))
```
# Pre-process variants
vcf <- VariantAnnotation::readVcf(
  vcfFile, param = tparam, colData = phenotypes)
vcf <- VariantAnnotation::expand(vcf, row.names = TRUE)

# Example usage ----
vepInPhenoLevel(vcf, "super_pop", "AFR", "CADD_PHRED")
Index

*Topic methods
  addCountGenos-methods, 3
  countGenos-methods, 9
  dropInfo-methods, 11
  variantsInSamples-methods, 18
  vepInPhenoLevel-methods, 25

*Topic package
  TVTB-package, 2
  [,VcfFilterRules,ANY,ANY,ANY-method
    (VcfFilterRules-class), 22
  [,VcfFilterRules,ANY,ANY,logical-method
    (VcfFilterRules-class), 22
  [,VcfFilterRules,ANY,ANY,missing-method
    (VcfFilterRules-class), 22
  [,VcfFixedRules,ANY,ANY-method
    (VcfBasicRules-class), 19
  [,VcfInfoRules,ANY,ANY-method
    (VcfBasicRules-class), 19
  [,VcfVepRules,ANY,ANY-method
    (VcfBasicRules-class), 19
  [<-.,VcfFilterRules,numeric,missing,VcfFilterRules-method
    (VcfFilterRules-class), 22
  [<-.,VcfFixedRules,numeric,missing,VcfFixedRules-method
    (VcfBasicRules-class), 19
  [<-.,VcfInfoRules,numeric,missing,VcfInfoRules-method
    (VcfBasicRules-class), 19
  [<-.,VcfVepRules,numeric,missing,VcfVepRules-method
    (VcfBasicRules-class), 19
  [<-.,VcfFilterRules,ANY,ANY-method
    (VcfFilterRules-class), 22
  [<-.,VcfFixedRules,ANY,ANY-method
    (VcfBasicRules-class), 19
  [<-.,VcfInfoRules,ANY,ANY-method
    (VcfBasicRules-class), 19
  [<-.,VcfVepRules,ANY,ANY-method
    (VcfBasicRules-class), 19
  aaf (TVTBparam-class), 16
  aaf,TVTBparam-method (TVTBparam-class), 16
  aaf<-,TVTBparam-class), 16
  addCountGenos (addCountGenos-methods), 3
  addCountGenos,ExpandedVCF-method
    (addCountGenos-methods), 3
  addCountGenos-methods, 3
  addFrequencies
    (addFrequencies-methods), 5
  addOverallFrequencies
    (addOverallFrequencies-methods), 7
  addPhenoLevelFrequencies
    (addPhenoLevelFrequencies-methods), 8
  alt,Genotypes-method (Genotypes-class), 12
alt, TVTBparam-method (TVTBparam-class), 16
alt<-, Genotypes, character-method (Genotypes-class), 12
alt<-, TVTBparam, character-method (TVTBparam-class), 16
alt<-, TVTBparam, list-method (TVTBparam-class), 16
append, VcfFilterRules, FilterRules-method (VcfFilterRules-class), 22
append, VcfFixedRules, FilterRules-method (VcfBasicRules-class), 19
append, VcfInfoRules, FilterRules-method (VcfBasicRules-class), 19
append, VcfVepRules, FilterRules-method (VcfBasicRules-class), 19
BiocParallelParam, 17
bp (TVTBparam-class), 16
bp, TVTBparam-method (TVTBparam-class), 16
bp<-, (TVTBparam-class), 16
bp<-, TVTBparam, BiocParallelParam-method (TVTBparam-class), 16
c, VcfFilterRules-method (VcfFilterRules-class), 22
c, VcfFixedRules-method (VcfBasicRules-class), 19
c, VcfInfoRules-method (VcfBasicRules-class), 19
c, VcfVepRules-method (VcfBasicRules-class), 19
carrier (Genotypes-class), 12
carrier, Genotypes-method (Genotypes-class), 12
carrier, TVTBparam-method (TVTBparam-class), 16
class: Genotypes (Genotypes-class), 12
class: TVTBparam (TVTBparam-class), 16
class: VcfFilterRules (VcfFilterRules-class), 22
class: VcfFixedRules (VcfBasicRules-class), 19
class: VcfInfoRules (VcfBasicRules-class), 19
class: VcfVepRules (VcfBasicRules-class), 19
countGenos (countGenos-methods), 9
countGenos, ExpandedVCF-method (countGenos-methods), 9
countGenos-methods, 9
Dataframe, 14, 26
dropInfo (dropInfo-methods), 11
dropInfo, VCF-method (dropInfo-methods), 11
dropInfo-methods, 11
enssemblVEP, 26
eval, VcfFilterRules, VCF-method (VcfFilterRules-class), 22
eval, VcfFixedRules, VCF-method (VcfBasicRules-class), 19
eval, VcfInfoRules, VCF-method (VcfBasicRules-class), 19
eval, VcfVepRules, VCF-method (VcfBasicRules-class), 19
ExpandedVCF, 17
FilterRules, 20, 21, 23, 24
genos (Genotypes-class), 12
genos, Genotypes-method (Genotypes-class), 12
genos, TVTBparam-method (TVTBparam-class), 16
genos<-, (TVTBparam-class), 16
genos<-, TVTBparam, Genotypes-method (TVTBparam-class), 16
Genotypes, 17
Genotypes (Genotypes-class), 12
Genotypes, character, character, character-method (Genotypes-class), 12
Genotypes-class, 12
Genotypes-methods (Genotypes-class), 12
GRanges, 26
het (Genotypes-class), 12
het, Genotypes-method (Genotypes-class), 12
het, TVTBparam-method (TVTBparam-class), 16
het<-, (Genotypes-class), 12
het<-, Genotypes, character-method (Genotypes-class), 12
het<-, TVTBparam, character-method (TVTBparam-class), 16
het<-, TVTBparam, list-method (TVTBparam-class), 16
initialize, Genotypes-method (Genotypes-class), 12
initialize, TVTBparam-method (TVTBparam-class), 16
initialize, VcfFilterRules-method (VcfFilterRules-class), 22
initialize,VcfFixedRules-method (VcfBasicRules-class), 19
initialize,VcfInfoRules-method (VcfBasicRules-class), 19
initialize,VcfVepRules-method (VcfBasicRules-class), 19

List, 20, 23

maf,TVTBparam-method (TVTBparam-class), 16
maf<- (TVTBparam-class), 16
maf<-,TVTBparam,character-method (TVTBparam-class), 16

ranges (TVTBparam-class), 16
ranges,TVTBparam-method (TVTBparam-class), 16
ranges<-,TVTBparam-class), 16
ranges<-,TVTBparam,GRangesList-method (TVTBparam-class), 16

readVcf, 14
readVcf,character,TVTBparam-method (readVcf-methods), 14
readVcf,TabixFile,TVTBparam-method (readVcf-methods), 14
readVcf-methods, 14
ref,Genotypes-method (Genotypes-class), 12
ref,TVTBparam-method (TVTBparam-class), 16
ref<-,Genotypes,character-method (Genotypes-class), 12
ref<-,TVTBparam,character-method (TVTBparam-class), 16
ref<-,TVTBparam,list-method (TVTBparam-class), 16

runApp, 15
runEnsDbApp, 15

ScanVcfParam, 14, 16
suffix (Genotypes-class), 12
suffix,Genotypes-method (Genotypes-class), 12
suffix,TVTBparam-method (TVTBparam-class), 16
svp (TVTBparam-class), 16
svp,TVTBparam-method (TVTBparam-class), 16
svp<-,TVTBparam-class), 16
svp<-,TVTBparam,ScanVcfParam-method (TVTBparam-class), 16

tSVE, 15
TVTB-package, 2
TVTBparam, 5–9, 13, 14, 18, 19, 26
TVTBparam (TVTBparam-class), 16
TVTBparam,Genotypes-method (TVTBparam-class), 16
TVTBparam-class, 16
TVTBparam-methods (TVTBparam-class), 16
type,FilterRules-method (VcfBasicRules-class), 19
type,VcfFilterRules-method (VcfFilterRules-class), 22
type,VcfFixedRules-method (VcfBasicRules-class), 19
type,VcfInfoRules-method (VcfBasicRules-class), 19
type,VcfVepRules-method (VcfBasicRules-class), 19

variantsInSamples (variantsInSamples-methods), 18
variantsInSamples,ExpandedVCF-method (variantsInSamples-methods), 18
variantsInSamples-methods, 18
VCF, 6, 7, 9–11, 13, 14, 17, 19, 21, 24, 26
VcfBasicRules-class, 19
VcfFilterRules, 21
VcfFilterRules (VcfFilterRules-class), 22
VcfFilterRules-class, 22
VcfFixedRules, 24
VcfFixedRules (VcfBasicRules-class), 19
VcfFixedRules-class (VcfBasicRules-class), 19
VcfInfoRules, 24
VcfInfoRules (VcfBasicRules-class), 19
VcfInfoRules-class (VcfBasicRules-class), 19
VcfVepRules, 24
VcfVepRules (VcfBasicRules-class), 19
VcfVepRules-class (VcfBasicRules-class), 19
Vector, 23
vep (TVTBparam-class), 16
vep,FilterRules-method (VcfBasicRules-class), 19
vep,TVTBparam-method (TVTBparam-class), 16
vep,VcfFilterRules-method (VcfFilterRules-class), 22
vep,VcfFixedRules-method (VcfBasicRules-class), 19
vep,VcfInfoRules-method
  (VcfBasicRules-class), 19
vep,VcfVepRules-method
  (VcfBasicRules-class), 19
vep<-(TVTBparam-class), 16
vep<-,TVTBparam,character-method
  (TVTBparam-class), 16
vep<-,VcfFilterRules,character-method
  (VcfFilterRules-class), 22
vep<-,VcfVepRules,character-method
  (VcfBasicRules-class), 19
vepInPhenoLevel
  (vepInPhenoLevel-methods), 25
vepInPhenoLevel,ExpandedVCF-method
  (vepInPhenoLevel-methods), 25
vepInPhenoLevel-methods, 25