Package: SeqVarTools (via r-universe)

November 3, 2026

Version 1.51.3

Type Package

Title Tools for variant data

Description An interface to the fast-access storage format for VCF data provided in SeqArray, with tools for common operations and analysis.

Author Stephanie M. Gogarten, Xiuwen Zheng, Adrienne Stilp

Maintainer Stephanie M. Gogarten <sdmorris@uw.edu>

Depends SeqArray

Imports grDevices, graphics, stats, methods, Biobase, BiocGenerics, gdsfmt, GenomicRanges, IRanges, S4Vectors, GWASExactHW, logistf, Matrix, data.table,

Suggests BiocStyle, RUnit, stringr, knitr, rmarkdown

License GPL-3

URL https://github.com/smgogarten/SeqVarTools

LazyData yes

VignetteBuilder knitr

biocViews SNP, GeneticVariability, Sequencing, Genetics

Collate AllClasses.R AllGenerics.R AllUtilities.R Methods-SeqVarGDSClass.R Methods-SeqVarData.R Methods-Iterator.R chromWithPAR.R duplicateDiscordance.R hwe.R inbreedCoeff.R mendelErr.R pca.R setVariantID.R alternateAlleleDetection.R refFrac.R regression.R

Config/pak/sysreqs cmake make libicu-dev libx11-dev zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-09-25 18:29:35 UTC

RemoteUrl https://github.com/bioc/SeqVarTools

RemoteRef HEAD

RemoteSha 53069087460f46bf1c97cce18d1ee4d4620b9e46

Contents

SeqVarTools-package . . . 2
allele-methods . . . 3
alleleFrequency . . . 4
alternateAlleleDetection . . . 5
applyMethod . . . 7
chromWithPAR . . . 8
countSingletons . . . 9
duplicateDiscordance . . . 10
getGenotype . . . 12
getVariableLengthData . . . 14
heterozygosity . . . 16
hwe . . . 17
imputedDosage . . . 19
inbreedCoeff . . . 20
isSNV . . . 21
isVariant . . . 22
Iterator . . . 23
meanBySample . . . 25
mendelErr . . . 26
missingGenotypeRate . . . 27
pca . . . 28
pedigree . . . 29
refFrac . . . 30
regression . . . 32
SeqVarData . . . 33
setVariantID . . . 35
titv . . . 36
variantInfo . . . 37

Index **39**

`SeqVarTools-package` *Tools for Variant Analysis*

Description

This package provides tools for data exploration and analysis of variants, extending the functionality of the package **SeqArray**.

Details

SeqArray provides an alternative to the Variant Call Format (VCF) for storage of variants called from sequencing data, enabling efficient storage, fast access to subsets of the data, and rapid computation.

SeqVarTools provides an interface to the **SeqArray** storage format with tools for many common tasks in variant analysis and integration with basic S4 classes in Bioconductor.

Author(s)

Stephanie M. Gogarten, Xiuwen Zheng

Maintainer: Stephanie M. Gogarten <sdmorris@u.washington.edu>

`allele-methods` *Extract allele information from a GDS object*

Description

Extract reference and alternate alleles and allele counts from a GDS object.

Usage

```
## S4 method for signature 'SeqVarGDSClass'
refChar(gdsobj)
## S4 method for signature 'SeqVarGDSClass'
altChar(gdsobj, n=0)
## S4 method for signature 'SeqVarGDSClass'
nAlleles(gdsobj)
```

Arguments

`gdsobj`	A `SeqVarGDSClass` object with VCF data.
`n`	An integer indicating which alternate allele to return. `n=0` returns a comma-separated string of all alternate alleles.

Details

These methods parse the "allele" field of a GDS object.

Value

`refChar` returns a character vector of reference alleles.

`altChar` returns a character vector of alternate alleles. If `n=0`, multiple alternate alleles are represented as a comma-separated string. If `n>0`, only the nth alternate allele is returned.

`nAlleles` returns an integer vector of the number of alleles (reference and alternate) for each variant.

Author(s)

Stephanie Gogarten

See Also

`SeqVarGDSClass`, `applyMethod`

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
table(refChar(gds))
table(altChar(gds))
table(altChar(gds, n=1))
table(altChar(gds, n=2), useNA="ifany")
table(nAlleles(gds))
seqClose(gds)
```

`alleleFrequency` *Allele frequency*

Description

Calculate allele frequency for each variant

Usage

```
## S4 method for signature 'SeqVarGDSClass'
alleleFrequency(gdsobj, n=0, use.names=FALSE, parallel=FALSE)
## S4 method for signature 'SeqVarData'
alleleFrequency(gdsobj, n=0, use.names=FALSE, sex.adjust=TRUE, male.diploid=TRUE,
    genome.build=c("hg19", "hg38"), parallel=FALSE)
## S4 method for signature 'SeqVarGDSClass'
alleleCount(gdsobj, n=0, use.names=FALSE, parallel=FALSE)
## S4 method for signature 'SeqVarData'
alleleCount(gdsobj, n=0, use.names=FALSE, sex.adjust=TRUE, male.diploid=TRUE,
    genome.build=c("hg19", "hg38"), parallel=FALSE)
## S4 method for signature 'SeqVarData'
minorAlleleCount(gdsobj, use.names=FALSE, sex.adjust=TRUE, male.diploid=TRUE,
    genome.build=c("hg19", "hg38"), parallel=FALSE)
```

Arguments

`gdsobj`	A `SeqVarGDSClass` object with VCF data.
`n`	An integer indicating which allele to calculate the frequency of. `n=0` is the reference allele, `n=1` is the first alternate allele, and so on.
`use.names`	A logical indicating whether to assign variant IDs as names of the output vector.
`sex.adjust`	Logical for whether to adjust frequency calculations based on sex. If `TRUE`, X chromosome frequency (excluding the PAR) will be calculated assuming the dosage of the specifed allele for males is half that for females. Y chromosome frequency will be calculated using males only.
`male.diploid`	Logical for whether males on sex chromosomes are coded as diploid.
`genome.build`	A character sting indicating genome build; used to identify pseudoautosomal regions on the X and Y chromosomes.
`parallel`	Logical, numeric, or other value to control parallel processing; see `seqParallel` for details.

Details

Frequency or count can be calculated over any allele, specified by the argument `n`. Default is the reference allele frequency (`n=0`).

The `SeqVarData` method will calculate frequency and count correctly for X and Y chromosomes, provided a column "sex" is included in the sampleData slot with values "M"/"F" or 1/2. Arguments given to this method are passed to the parent method for `SeqVarGDSClass`. If the ploidy of the "genotype" node in the GDS file is 2, the default assumption is that genotypes for males on sex chromosomes are coded as diploid, "0/0" or "1/1". If this is not the case, use `male.diploid=FALSE`.

For multiallelic variants, the minor allele count will be the smaller of the reference allele count or the sum of all alternate allele counts.

Value

A numeric vector of allele frequencies.

Author(s)

Stephanie Gogarten

See Also

`chromWithPAR`, `SeqVarGDSClass`, `applyMethod`, `heterozygosity`

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
head(alleleFrequency(gds))
head(alleleFrequency(gds, n=1))
head(alleleFrequency(gds, n=2))
seqClose(gds)
```

`alternateAlleleDetection` *alternateAlleleDetection*

Description

Calculate rates of detecting minor alleles given a "gold standard" dataset

Usage

```
## S4 method for signature 'SeqVarData,SeqVarData'
alternateAlleleDetection(gdsobj, gdsobj2,
    match.samples.on=c("subject.id", "subject.id"), verbose=TRUE)
```

Arguments

gdsobj	A SeqVarData object with VCF data.
gdsobj2	A SeqVarData object with VCF data to be used as the "gold standard".
match.samples.on	A length-2 character vector indicating the column to be used for matching in each dataset's sampleData annotation
verbose	A logical indicating whether to print progress messages.

Details

Calculates the accuracy of detecting alternate alleles in one dataset (gdsobj) given a "gold standard" dataset (gdsobj2). Samples are matched using the match.samples.on argument. The first element of match.samples.on indicates the column to be used as the subject identifier for the first dataset, and the second element is the column to be used for the second dataset. Variants are matched on position and alleles using bi-allelic SNVs only. Genotype dosages are recoded to count the same allele if the reference allele in one dataset is the alternate allele in the other dataset. If a variant in one dataset matches to multiple variants in the second dataset, then only the first match will be used. If a variant is missing in either dataset for a given sample pair, that sample pair is ignored for that variant. To exclude certain variants or samples from the calculate, use seqSetFilter to set appropriate filters on each gds object.

This test is positive if an alternate allele was been detected. Results are returned on an allele level, such that:

TP, TN, FP, and FN are calculated as follows:

		genoData2		
		aa	Ra	RR
	aa	2TP	1TP + 1FP	2FP
genoData1	Ra	1TP + 1FN	1TN + 1TP	1TN + 1FP
	RR	2FN	1FN + 1TN	2TN

where "R" indicates a reference allele and "a" indicates an alternate allele.

Value

A data frame with the following columns:

variant.id.1	variant id from the first dataset
variant.id.2	matched variant id from the second dataset
n.samples	the number of samples with non-missing data for this variant
true.pos	the number of alleles that are true positives for this variant
true.neg	the number of alleles that are true negatives for this variant
false.pos	the number of alleles that are false positives for this variant
false.neg	the number of alleles that are false negatives for this variant

Author(s)

Adrienne Stilp

See Also

SeqVarGDSClass

Examples

```
## Not run:
gds1 <- seqOpen(gdsfile.1) # dataset to test, e.g. sequencing
sample1 <- data.frame(subject.id=c("a", "b", "c"), sample.id=c("A", "B", "C"), stringsAsFactors=F)
seqData1 <- SeqVarData(gds1, sampleData=sample1)

gds2 <- seqOpen(gdsfile.2) # gold standard dataset, e.g. array genotyping
sample2 <- data.frame(subject.id=c("b", "c", "d"), sample.id=c("B", "C", "D"), stringsAsFactors=F)
seqData2 <- SeqVarData(gds2, sampleData=sample2)

res <- alleleDetectionAccuracy(seqData1, seqData2)

## End(Not run)
```

---

applyMethod *Apply method to GDS object*

Description

Apply a method to a subset of variants and/or samples in a GDS object

Usage

```
## S4 method for signature 'SeqVarGDSClass,function,character'
applyMethod(gdsobj, FUN, variant, sample=NULL, ...)
## S4 method for signature 'SeqVarGDSClass,function,numeric'
applyMethod(gdsobj, FUN, variant, sample=NULL, ...)
## S4 method for signature 'SeqVarGDSClass,function,GRanges'
applyMethod(gdsobj, FUN, variant, sample=NULL, ...)
## S4 method for signature 'SeqVarGDSClass,function,missing'
applyMethod(gdsobj, FUN, variant, sample=NULL, ...)
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
FUN	A method or function to be applied to gdsobj.
variant	A vector of variant.id values or a GRanges object defining the variants to be included in the call to FUN.
sample	A vector of sample.id values defining the samples to be included in the call to FUN.
...	Additional arguments, passed to FUN.

Details

applyMethod applies a method or function FUN to the subset of variants defined by variant and samples defined by sample in a GDS object.

If a filter was previously set with seqSetFilter, it will be saved and reset after the call to applyMethod.

Value

The result of the call to FUN.

Author(s)

Stephanie Gogarten

See Also

SeqVarGDSClass

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
variant.id <- seqGetData(gds, "variant.id")
sample.id <- seqGetData(gds, "sample.id")
applyMethod(gds, getGenotype, variant.id[1:5], sample.id[1:10])

library(GenomicRanges)
chrom <- seqGetData(gds, "chromosome")
pos22 <- seqGetData(gds, "position")[chrom == 22]
ranges <- GRanges(seqnames="22", IRanges(min(pos22), max(pos22)))
applyMethod(gds, heterozygosity, ranges, margin="by.sample")
applyMethod(gds, heterozygosity, ranges, margin="by.variant")

seqClose(gds)
```

---

chromWithPAR *Identify pseudoautosomal region*

Description

Flag single nucleotide variants

Usage

```
## S4 method for signature 'SeqVarGDSClass'
chromWithPAR(gdsobj, genome.build=c("hg19", "hg38"))
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
genome.build	A character sting indicating genome build.

Details

The pseudoautosomal region (PAR) should be treated like the autosomes for purposes of calculating allele frequency. This method returns a vector where sex chromosome variants are labeled wither "X", "Y", or "PAR".

Value

A character vector of chromosome, with values "PAR" for the pseudoautosomal region.

Author(s)

Stephanie Gogarten

References

https://www.ncbi.nlm.nih.gov/grc/human

---

countSingletons *Count singletons*

Description

Count singleton variants for each sample

Usage

```
## S4 method for signature 'SeqVarGDSClass'
countSingletons(gdsobj, use.names=FALSE)
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
use.names	A logical indicating whether to assign variant IDs as names of the output vector.

Details

A singleton variant is a variant in which only one sample has a non-reference allele. For each sample, countSingletons finds the number of variants for which that sample has the only non-reference allele.

Value

A vector of the number of singleton variants per sample.

Author(s)

Stephanie Gogarten

See Also

SeqVarGDSClass, applyMethod, alleleFrequency

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
head(countSingletons(gds))
seqClose(gds)
```

---

duplicateDiscordance *Duplicate discordance*

Description

Find discordance rate for duplicate sample pairs

Usage

```
## S4 method for signature 'SeqVarData,missing'
duplicateDiscordance(gdsobj, match.samples.on="subject.id", by.variant=FALSE,
    all.pairs=TRUE, verbose=TRUE)
## S4 method for signature 'SeqVarIterator,missing'
duplicateDiscordance(gdsobj, match.samples.on="subject.id", by.variant=FALSE,
    all.pairs=TRUE, verbose=TRUE)
## S4 method for signature 'SeqVarData,SeqVarData'
duplicateDiscordance(gdsobj, obj2, match.samples.on=c("subject.id", "subject.id"),
    match.variants.on=c("alleles", "position"),
    discordance.type=c("genotype", "hethom"),
    by.variant=FALSE, verbose=TRUE)
```

Arguments

gdsobj	A SeqVarData object with VCF data.
obj2	A SeqVarData object with VCF data.
match.samples.on	Character string or vector of strings indicating which column should be used for matching samples. See details.
match.variants.on	Character string of length one indicating how to match variants. See details.
discordance.type	Character string describing how discordances should be calculated. See details.
by.variant	Calculate discordance by variant, otherwise by sample
all.pairs	Logical for whether to include all possible pairs of samples (all.pairs=TRUE) or only the first pair per subject (all.pairs=FALSE).
verbose	A logical indicating whether to print progress messages.

Details

For calls that involve only one gds file, duplicate discordance is calculated by matching samples on common values of a column in sampleData. If all.pairs=TRUE, every possible pair of samples is included, so there may be multiple pairs per subject. If all.pairs=FALSE, only the first pair for each subject is used.

For calls that involve two gds files, duplicate discordance is calculated by matching sample pairs and variants between the two data sets. Only biallelic SNVs are considered in the comparison. Variants can be matched using chromosome and position only (match.variants.on="position") or by using chromosome, position, and alleles (match.variants.on="alleles"). If matching on alleles and the reference allele in the first dataset is the alternate allele in the second dataset, the genotype dosage will be recoded so the same allele is counted before making the comparison. If a variant in one dataset maps to multiple variants in the other dataset, only the first pair is considered for the comparison. Discordances can be calculated using either genotypes (discordance.type = "genotype") or heterozygote/homozygote status (discordance.type = "hethom"). The latter is a method to calculate discordance that does not require alleles to be measured on the same strand in both datasets, so it is probably best to also set match.variants.on = "position" if using the "hethom" option.

The argument match.samples.on can be used to select which column in the sampleData of the input SeqVarData object should be used for matching samples. For one gds file, match.samples.on should be a single string. For two gds files, match.samples.on should be a length-2 vector of character strings, where the first element is the column to use for the first gds object and the second element is the column to use for the second gds file.

To exclude certain variants or samples from the calculate, use seqSetFilter to set appropriate filters on each gds object.

Value

A data frame with the following columns, depending on whether by.variant=TRUE or FALSE:

subject.id	currently, this is the sample ID (by.variant=FALSE only)
sample.id.1/variant.id.1	sample id or variant id in the first gds file
sample.id.2/variant.id.2	sample id or variant id in the second gds file
n.variants/n.samples	the number of non-missing variants or samples that were compared
n.concordant	the number of concordant variants
n.alt	the number of variants involving the alternate allele in either sample
n.alt.conc	the number of concordant variants invovling the alternate allele in either sample
n.het.ref	the number of mismatches where one call is a heterozygote and the other is a reference homozygote
n.het.alt	the number of mismatches where one call is a heterozygote and the other is an alternate homozygote
n.ref.alt	the number of mismatches where the calls are opposite homozygotes

Author(s)

Stephanie Gogarten, Adrienne Stilp

See Also

SeqVarData, SeqVarIterator

Examples

```
require(Biobase)

gds <- seqOpen(seqExampleFileName("gds"))

## the example file has one sample per subject, but we
## will match the first four samples into pairs as an example
sample.id <- seqGetData(gds, "sample.id")
samples <- AnnotatedDataFrame(data.frame(data.frame(subject.id=rep(c("subj1", "subj2"), times=45),
                      sample.id=sample.id,
                      stringsAsFactors=FALSE)))
seqData <- SeqVarData(gds, sampleData=samples)

## set a filter on the first four samples
seqSetFilter(seqData, sample.id=sample.id[1:4])

disc <- duplicateDiscordance(seqData, by.variant=FALSE)
disc
disc <- duplicateDiscordance(seqData, by.variant=TRUE)
head(disc)

## recommended to use an iterator object for large datasets
iterator <- SeqVarBlockIterator(seqData)
disc <- duplicateDiscordance(iterator, by.variant=FALSE)
disc

seqClose(gds)
```

---

getGenotype *Get genotype data*

Description

Get matrix of genotype values from a GDS object

Usage

```
## S4 method for signature 'SeqVarGDSClass'
getGenotype(gdsobj, use.names=TRUE, parallel=FALSE)
## S4 method for signature 'SeqVarGDSClass'
getGenotypeAlleles(gdsobj, use.names=TRUE, sort=FALSE, parallel=FALSE)
```

```
## S4 method for signature 'SeqVarGDSClass'
refDosage(gdsobj, use.names=TRUE, ...)
## S4 method for signature 'SeqVarGDSClass'
altDosage(gdsobj, use.names=TRUE, sparse=FALSE, parallel=FALSE, ...)
## S4 method for signature 'SeqVarGDSClass'
expandedAltDosage(gdsobj, use.names=TRUE, sparse=FALSE, parallel=FALSE)
## S4 method for signature 'SeqVarGDSClass,numeric'
alleleDosage(gdsobj, n=0, use.names=TRUE, parallel=FALSE)
## S4 method for signature 'SeqVarGDSClass,list'
alleleDosage(gdsobj, n, use.names=TRUE, parallel=FALSE)
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
use.names	A logical indicating whether to assign sample and variant IDs as dimnames of the resulting matrix.
parallel	Logical, numeric, or other value to control parallel processing; see seqParallel for details.
sort	Logical for whether to sort alleles lexographically ("G/T" instead of "T/G").
sparse	Logical for whether to return the alterate allele dosage as a sparse matrix using the **Matrix** package. In most cases, setting sparse=TRUE will dramatically reduce the size of the returned object.
n	An integer, vector, or list indicating which allele(s) to return dosage for. n=0 is the reference allele, n=1 is the first alternate allele, and so on.
...	Arguments to pass to seqBlockApply, e.g. bsize to set the block size.

Details

In getGenotype, genotypes are coded as in the VCF file, where "0/0" is homozygous reference, "0/1" is heterozygous for the first alternate allele, "0/2" is heterozygous for the second alternate allele, etc. Separators are "/" for unphased and "|" for phased. If sort=TRUE, all returned genotypes will be unphased. Missing genotypes are coded as NA. Only haploid or diploid genotypes (the first two alleles at a given site) are returned.

If the argument n toalleleDosage is a single integer, the same allele is counted for all variants. If n is a vector with length=number of variants in the current filter, a different allele is counted for each variant. If n is a list, more than one allele can be counted for each variant. For example, if n[[1]]=c(1,3), genotypes "0/1" and "0/3" will each have a dosage of 1 and genotype "1/3" will have a dosage of 2.

Value

getGenotype and getGenotypeAlleles return a character matrix with dimensions [sample,variant] containing haploid or diploid genotypes.

getGenotype returns alleles as "0", "1", "2", etc. indicating reference and alternate alleles.

getGenotypeAlleles returns alleles as "A", "C", "G", "T". sort=TRUE sorts lexographically, which may be useful for comparing genotypes with data generated using a different reference sequence.

`refDosage` returns an integer matrix with the dosage of the reference allele: 2 for two copies of the reference allele ("0/0"), 1 for one copy of the reference allele, and 0 for two alternate alleles.

`altDosage` returns an integer matrix with the dosage of any alternate allele: 2 for two alternate alleles ("1/1", "1/2", etc.), 1 for one alternate allele, and 0 for no alternate allele (homozygous reference).

`expandedAltDosage` returns an integer matrix with the dosage of each alternate allele as a separate column. A variant with 2 possible alternate alleles will have 2 columns of output, etc.

`alleleDosage` with an integer argument returns an integer matrix with the dosage of the specified allele only: 2 for two copies of the allele ("0/0" if `n=0`, "1/1" if `n=1`, etc.), 1 for one copy of the specified allele, and 0 for no copies of the allele.

`alleleDosage` with a list argument returns a list of sample x allele matrices with the dosage of each specified allele for each variant.

Author(s)

Stephanie Gogarten

See Also

`SeqVarGDSClass`, `applyMethod`, `seqGetData`, `seqSetFilter`, `alleleFrequency`

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
seqSetFilter(gds, variant.sel=1323:1327, sample.sel=1:10)
nAlleles(gds)
getGenotype(gds)
getGenotypeAlleles(gds)
refDosage(gds)
altDosage(gds)
expandedAltDosage(gds)
alleleDosage(gds, n=0)
alleleDosage(gds, n=1)
alleleDosage(gds, n=c(0,1,0,1,0))
alleleDosage(gds, n=list(0,c(0,1),0,c(0,1),1))
seqClose(gds)
```

---

getVariableLengthData *Get variable-length data*

Description

Get data with multiple values per sample from a GDS object and return as an array

Usage

```
## S4 method for signature 'SeqVarGDSClass,character'
getVariableLengthData(gdsobj, var.name, use.names=TRUE, parallel=FALSE)
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
var.name	Character string with name of the variable, most likely "annotation/format/VARIABLE_NAME".
use.names	A logical indicating whether to assign sample and variant IDs as dimnames of the resulting matrix.
parallel	Logical, numeric, or other value to control parallel processing; see seqParallel for details.

Details

Data which are indicated as having variable length (possibly different numbers of values for each variant) in the VCF header are stored as variable-length data in the GDS file. Each such data object has two components, "length" and "data." "length" indicates how many values there are for each variant, while "data" is a matrix with one row per sample and columns defined as all values for variant 1, followed by all values for variant 2, etc.

getVariableLengthData converts this format to a 3-dimensional array, where the length of the first dimension is the maximum number of values in "length," and the remaining dimensions are sample and variant. Missing values are given as NA. If the first dimension of this array would have length 1, the result is converted to a matrix.

Value

An array with dimensions [n, sample, variant] where n is the maximum number of values possible for a given sample/variant cell. If n=1, a matrix with dimensions [sample,variant].

Author(s)

Stephanie Gogarten

See Also

SeqVarGDSClass, applyMethod, seqGetData

Examples

```
file <- system.file("extdata", "gl_chr1.gds", package="SeqVarTools")
gds <- seqOpen(file)
## genotype likelihood
gl <- seqGetData(gds, "annotation/format/GL")
names(gl)
gl$length
## 3 values per variant - likelihood of RR,RA,AA genotypes
dim(gl$data)
## 85 samples (rows) and 9 variants with 3 values each - 27 columns

gl.array <- getVariableLengthData(gds, "annotation/format/GL")
dim(gl.array)
## 3 genotypes x 85 samples x 9 variants
head(gl.array[1,,])
```

```
head(gl.array[2,,])
head(gl.array[3,,])

seqClose(gds)
```

`heterozygosity` *Heterozygosity and Homozygosity*

Description

Calculate heterozygosity and homozygosity by variant or by sample

Usage

```
## S4 method for signature 'SeqVarGDSClass'
heterozygosity(gdsobj, margin=c("by.variant", "by.sample"),
    use.names=FALSE, parallel=FALSE)
## S4 method for signature 'SeqVarGDSClass'
homozygosity(gdsobj, allele=c("any", "ref", "alt"), margin=c("by.variant", "by.sample"),
    use.names=FALSE, parallel=FALSE)
## S4 method for signature 'SeqVarGDSClass'
hethom(gdsobj, use.names=FALSE)
```

Arguments

`gdsobj`	A `SeqVarGDSClass` object with VCF data.
`margin`	Possible values are "by.variant" or "by.sample," indicating whether the calculation should be done over all samples for each variant, or over all variants for each sample.
`use.names`	A logical indicating whether to assign variant or samples IDs as names of the output vector.
`allele`	Possible values are "any", "ref," or "alt," indicating which alleles to consider when calculating homozygosity.
`parallel`	Logical, numeric, or other value to control parallel processing; see `seqParallel` for details. Only applies if `margin="as.variant"`.

Details

`heterozyogosity` calulates the fraction of heterozygous genotypes in a GDS object, either by variant or by sample.

`homozygosity` calculates the rate of homozygous genotypes in a GDS object, either by sample or by variant. If `allele="any"`, all homozygous genotypes are considered (reference or any alternate allele). If `allele="ref"`, only reference homozygotes are considered. If `allele="alt"`, any alternate allele homozygote is considered. For example, "ref" will count "0/0" genotypes only, "alt" will count "1/1", "2/2", etc. (but not "0/0"), and "any" will count all of the above.

`hethom` calculates the ratio of heterozygous genotypes to alternate homozygous genotypes by sample.

Value

A numeric vector of heterozyogity or homozygosity rates. If `margin="by.variant"`, the vector will have length equal to the number of variants in the GDS object. If `margin="by.sample"`, the vector will have length equal to the number of samples.

Author(s)

Stephanie Gogarten

See Also

`SeqVarGDSClass`, `applyMethod`, `alleleFrequency`

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
head(heterozygosity(gds, margin="by.variant"))
head(homozygosity(gds, allele="any", margin="by.variant"))
head(homozygosity(gds, allele="ref", margin="by.variant"))
head(homozygosity(gds, allele="alt", margin="by.variant"))

## Het/Hom Non-Ref by sample
head(hethom(gds))

seqClose(gds)
```

---

hwe *Exact test for Hardy-Weinberg equilibrium*

Description

Performs an exact test for Hardy-Weinberg equilibrium on Single-Nucleotide Variants

Usage

```
## S4 method for signature 'SeqVarGDSClass'
hwe(gdsobj, permute=FALSE, parallel=FALSE)
```

Arguments

`gdsobj`	A `SeqVarGDSClass` object with VCF data.
`permute`	A logical indicating whether to permute the genotypes to get a set of p-values under the null hypothesis.
`parallel`	Logical, numeric, or other value to control parallel processing; see `seqParallel` for details.

Details

HWE calculations are performed with the `HWExact` function in the **GWASExactHW** package.

`permute=TRUE` will permute the genotypes prior to running the test. This can be useful for obtaining a set of expected values under the null hypothesis to compare to the observed values.

P values are set to `NA` for all multiallelic and monomorphic variants.

Value

A data.frame with the following columns:

`variant.id`	The unique identifier for the variant
`nAA`	The number of reference homozygotes
`nAa`	The number of heterozygotes
`naa`	The number of alternate homozygotes
`afreq`	The reference allele frequency
`p`	p values for the exact test
`f`	The inbreeding coefficient, 1 - observed heterozygosity / expected heterozygosity

Author(s)

Stephanie Gogarten

See Also

`SeqVarGDSClass`, `applyMethod`

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
## autosomal variants only
auto <- seqGetData(gds, "chromosome") %in% 1:22
var.auto <- seqGetData(gds, "variant.id")[auto]
hw <- applyMethod(gds, hwe, variant=var.auto)
head(hw)
sum(is.na(hw$p))
range(hw$p, na.rm=TRUE)
seqClose(gds)
```

imputedDosage *Get imputed dosage*

Description

Get matrix of imputed dosage values from a GDS object

Usage

```
## S4 method for signature 'SeqVarGDSClass'
imputedDosage(gdsobj, dosage.field="DS", use.names=TRUE)
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
dosage.field	The name of the dosage field in the GDS object (will be prepended with "annotation/format").
use.names	A logical indicating whether to assign sample and variant IDs as dimnames of the resulting matrix.

Details

Reads dosage from the dosage-specific field in the GDS object, rather than counting alleles from called genotypes.

Only one dosage value per variant is allowed; the method will return an error if multiple dosages are present for a single variant.

Value

A numeric matrix of dosage values with dimensions [sample,variant].

Author(s)

Stephanie Gogarten

See Also

refDosage, altDosage

Examples

```
# convert VCF to GDS keeping dosage field
vcffile <- system.file("extdata", "gl_chr1.vcf", package="SeqVarTools")
gdsfile <- tempfile()
seqVCF2GDS(vcffile, gdsfile, fmt.import="DS", storage.option="ZIP_RA",
           verbose=FALSE)

gds <- seqOpen(gdsfile)
dos <- imputedDosage(gds)
```

```
head(dos)
seqClose(gds)
unlink(gdsfile)
```

`inbreedCoeff` *Inbreeding coefficient*

Description

Calculates the inbreeding coefficient by variant or by sample

Usage

```
## S4 method for signature 'SeqVarGDSClass'
inbreedCoeff(gdsobj, margin=c("by.variant", "by.sample"), use.names=FALSE,
    parallel=FALSE)
```

Arguments

`gdsobj`	A `SeqVarGDSClass` object with VCF data.
`margin`	Possible values are "by.variant" or "by.sample," indicating whether the calculation should be done over all samples for each variant, or over all variants for each sample.
`use.names`	A logical indicating whether to assign variant or sample IDs as names of the output vector.
`parallel`	Logical, numeric, or other value to control parallel processing; see `seqParallel` for details. Only applies if `margin="as.variant"`.

Details

For inbreeding coefficients by variant, calculates 1 - observed heterozygosity / expected heterozygosity.

For individual inbreeding coefficients (`margin="by.sample"`), calculates Visscher's estimator described in Yang et al. (2010).

Value

Values for the inbreeding coefficient.

Author(s)

Xiuwen Zheng, Stephanie Gogarten

References

Yang J, Benyamin B, McEvoy BP, Gordon S, Henders AK, Nyholt DR, Madden PA, Heath AC, Martin NG, Montgomery GW, Goddard ME, Visscher PM. 2010. Common SNPs explain a large proportion of the heritability for human height. Nat Genet. 42(7):565-9. Epub 2010 Jun 20.

See Also

SeqVarGDSClass, applyMethod

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
f <- inbreedCoeff(gds, margin="by.variant")
range(f, na.rm=TRUE)

ic <- inbreedCoeff(gds, margin="by.sample")
range(ic)
seqClose(gds)
```

---

isSNV *Flag single nucleotide variants*

Description

Flag single nucleotide variants

Usage

```
## S4 method for signature 'SeqVarGDSClass'
isSNV(gdsobj, biallelic=TRUE)
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
biallelic	A logical indicating whether only biallelic SNVs are considered.

Details

If biallelic=TRUE, a variant is considered a single nucleotide variant (SNV) if there is one reference allele and one alternate allele, each one base in length. If biallelic=FALSE, there may be multiple alternate alleles, each one base in length.

Setting biallelic=TRUE is considerably faster for large data sets.

Value

A logical vector indicating which variants are SNVs.

Author(s)

Stephanie Gogarten

See Also

SeqVarGDSClass, allele-methods, applyMethod

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
table(isSNV(gds))
seqClose(gds)
```

`isVariant` *Locate variant samples across sites*

Description

Locate which samples are variant for each site in a GDS object

Usage

```
## S4 method for signature 'SeqVarGDSClass'
isVariant(gdsobj, use.names=FALSE, parallel=FALSE)
```

Arguments

`gdsobj`	A `SeqVarGDSClass` object with VCF data.
`use.names`	A logical indicating whether to assign sample and variant IDs as dimnames of the resulting matrix.
`parallel`	Logical, numeric, or other value to control parallel processing; see `seqParallel` for details.

Details

Each sample/site cell of the resulting matrix is `TRUE` if the genotype at that location for that sample contains an alternate allele. A genotype of "0/0" is not variant, while genotypes "0/1", "1/0", "0/2", etc. are variant.

Value

A logical matrix with dimensions [sample,site] which is `TRUE` for cells where the genotype contains an alternate allele.

Author(s)

Stephanie Gogarten

See Also

`SeqVarGDSClass`, `applyMethod`, `getGenotype`

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
variant.id <- seqGetData(gds, "variant.id")
sample.id <- seqGetData(gds, "sample.id")
applyMethod(gds, isVariant, variant.id[1:5], sample.id[1:10])
applyMethod(gds, isVariant, variant.id[1:5], sample.id[1:10], use.names=TRUE)
seqClose(gds)
```

`Iterator` *Iterators*

Description

Extends `SeqVarData` to provide iterators over variants.

Details

Iterator classes allow for iterating filters over blocks of variants, ranges, or sliding windows.

For `SeqVarBlockIterator`, each call to `iterateFilter` will select the next unit of `variantBlock` variants.

For `SeqVarRangeIterator`, each call to `iterateFilter` will select the next range in `variantRanges`.

`SeqVarWindowIterator` is an extension of `SeqVarRangeIterator` where the ranges are determined automatically by sliding a window of size `windowSize` base pairs by steps of `windowShift` across the genome. Only windows containing unique sets of variants are kept.

For `SeqVarListIterator`, each call to `iterateFilter` will select the next set of ranges in `variantRanges`.

Any filter set on the object previously will be applied in addition to the selected blocks or ranges.

Constructors

- `SeqVarBlockIterator(seqData, variantBlock=10000, verbose=TRUE)`: Returns a `SeqVarBlockIterator` object with the filter set to the first block.
  `seqData` is a `SeqVarData` object.
  `variantBlock` is an integer specifying the number of variants in an iteration block.
  `verbose` is a logical indicator for verbose output.
- `SeqVarRangeIterator(seqData, variantRanges=GRanges(), verbose=TRUE)`: Returns a `SeqVarRangeIterator` object with the filter set to the first range.
  `seqData` is a `SeqVarData` object.
  `variantRanges` is a `GRanges` object specifying the ranges for iteration.
  `verbose` is a logical indicator for verbose output.
- `SeqVarWindowIterator(seqData, windowSize=10000, windowShift=5000, verbose=TRUE)`: Returns a `SeqVarWindowIterator` object with the filter set to the first window.
  `seqData` is a `SeqVarData` object.
  `windowSize` is the size in base pairs of the sliding window.
  `windowShift` is the size in base pairs of the step for each consecutive window.
  `verbose` is a logical indicator for verbose output.

- `SeqVarListIterator(seqData, variantRanges, verbose=TRUE)`: Returns a `SeqVarRangeIterator` object with the filter set to the first range.

  `seqData` is a `SeqVarData` object.

  `variantRanges` is a `GRangesList` object specifying the ranges for iteration.

  `verbose` is a logical indicator for verbose output.

Accessors

- `iterateFilter(x)`: Advance the filter to the next block, range, or set of ranges. Returns `TRUE` while there are still variants left to be read, `FALSE` if the end of iteration is reached.
- `lastFilter(x)`, `lastFilter(x)<- value`: Get or set the last filter index from the previous call to `iterateFilter`.
- `variantBlock(x)`: Get the size of the variant block.
- `variantFilter(x)`: Get the list of variant indices.
- `variantRanges(x)`: Get the variant ranges.
- `currentRanges(x)`: Get the ranges selected in the current iteration.
- `currentVariants(x)`: Get the variants selected in the current iteration. Returns a `DataFrame` where the row name is the variant.id, "variant" is the variant position as a `link{GRanges}`, "range" is the range the variant is from, and any columns in either `variantData` or the metadata columns of `currentRanges` are included.
- `resetIterator(x)`: Set the filter to the first block, range, or set of ranges (the same variants that are selected when the iterator object is created).

Author(s)

Stephanie Gogarten

See Also

`SeqVarGDSClass`, `SeqVarData`, `seqSetFilter`

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
seqData <- SeqVarData(gds)

# iterate by blocks
seqSetFilter(seqData, variant.sel=seq(1,1000,2))
iterator <- SeqVarBlockIterator(seqData, variantBlock=10)
seqGetData(iterator, "variant.id")
iterateFilter(iterator)
seqGetData(iterator, "variant.id")
seqResetFilter(iterator)

# iterate by ranges
library(GenomicRanges)
gr <- GRanges(seqnames=rep(1,3), ranges=IRanges(start=c(1e6, 2e6, 3e6), width=1e6))
iterator <- SeqVarRangeIterator(seqData, variantRanges=gr)
```

```
granges(iterator)
iterateFilter(iterator) # no variants in the second range
granges(iterator)
iterateFilter(iterator)
granges(iterator)
iterateFilter(iterator)
seqResetFilter(iterator)

# iterate by windows
seqSetFilterChrom(seqData, include="22")
iterator <- SeqVarWindowIterator(seqData)
seqGetData(iterator, "variant.id")
while (iterateFilter(iterator)) {
    print(seqGetData(iterator, "variant.id"))
}
seqResetFilter(iterator)

# iterate by list of ranges
gr <- GRangesList(
  GRanges(seqnames=rep(22,2), ranges=IRanges(start=c(16e6, 17e6), width=1e6)),
  GRanges(seqnames=rep(22,2), ranges=IRanges(start=c(18e6, 20e6), width=1e6)))
iterator <- SeqVarListIterator(seqData, variantRanges=gr)
granges(iterator)
iterateFilter(iterator)
granges(iterator)
iterateFilter(iterator)
resetIterator(iterator)

seqClose(iterator)
```

`meanBySample` *Mean value by sample*

Description

Calculate the mean value of a variable by sample over all variants

Usage

```
## S4 method for signature 'SeqVarGDSClass'
meanBySample(gdsobj, var.name, use.names=FALSE)
```

Arguments

`gdsobj`	A `SeqVarGDSClass` object with VCF data.
`var.name`	Character string with name of the variable, most likely "annotation/format/VARIABLE_NAME".
`use.names`	A logical indicating whether to assign sample IDs as names of the output vector.

Details

Mean values by variant can be calculated using `seqApply(gdsobj, var.name, mean, na.rm=TRUE)`. Currently `seqApply` can only be used with the option `margin="by.variant"`. This method provides a way to calculate mean values by sample.

Value

A numeric vector of mean values.

Author(s)

Stephanie Gogarten

See Also

`SeqVarGDSClass`, `applyMethod`, `seqApply`

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
head(meanBySample(gds, "annotation/format/DP", use.names=TRUE))
seqClose(gds)
```

---

mendelErr *Mendelian errors*

Description

Detect Mendelian errors

Usage

```
## S4 method for signature 'SeqVarGDSClass'
mendelErr(gdsobj, pedigree, use.names=FALSE,
autosomes=1:22, xchrom="X", ychrom="Y", verbose=TRUE)
```

Arguments

`gdsobj`	A `SeqVarGDSClass` object with VCF data.
`pedigree`	A data.frame with columns (family, individ, father, mother, sex, sample.id). "sex" column should have values "M"/"F". "sample.id" values should correspond to "sample.id" in `gdsobj`.
`use.names`	A logical indicating whether to assign variant IDs as names of the output vector.
`autosomes`	A vector with chromosome values in `gdsobj` corresponding to autosomes.
`xchrom`	The chromosome value in `gdsobj` corresponding to the X chromosome.
`ychrom`	The chromosome value in `gdsobj` corresponding to the Y chromosome.
`verbose`	A logical indicating whether to print the number of samples selected for each trio.

Details

Mendelian errors are detected for each trio in `pedigree`. Duos (mother or father missing) are included. The pedigree must have only one sample per individual.

Value

A list with the following elements:

`by.variant`	An integer vector with the number of mendelian errors detected for each variant. If `use.names=TRUE`, the vector will be named with variant IDs.
`by.trio`	An integer vector with the number of mendelian errors detected for each trio. The vector will be named with the sample ID of the child in each trio.

Author(s)

Stephanie Gogarten

See Also

`SeqVarGDSClass`, `applyMethod`

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
data(pedigree)
err <- mendelErr(gds, pedigree)
table(err$by.variant)
err$by.trio
seqClose(gds)
```

`missingGenotypeRate` *Missing genotype rate*

Description

Calculate missing genotype rate by variant or by sample

Usage

```
## S4 method for signature 'SeqVarGDSClass'
missingGenotypeRate(gdsobj, margin=c("by.variant", "by.sample"), use.names=FALSE,
    parallel=FALSE)
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
margin	Possible values are "by.variant" or "by.sample," indicating whether the calculation should be done over all samples for each variant, or over all variants for each sample.
use.names	A logical indicating whether to assign variant IDs as names of the output vector.
parallel	Logical, numeric, or other value to control parallel processing; see seqParallel for details.

Details

Calculates the fraction of missing genotypes in a GDS object, either by variant or by sample.

Value

A numeric vector of missing genotype rates. If `margin="by.variant"`, the vector will have length equal to the number of variants in the GDS object. If `margin="by.sample"`, the vector will have length equal to the number of samples.

Author(s)

Stephanie Gogarten

See Also

SeqVarGDSClass, applyMethod, getGenotype

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
head(missingGenotypeRate(gds, margin="by.variant"))
head(missingGenotypeRate(gds, margin="by.sample"))
seqClose(gds)
```

---

pca *Principal Component Analysis*

Description

Calculates the eigenvalues and eignevectors of a SeqVarGDSClass object with Principal Component Analysis

Usage

```
## S4 method for signature 'SeqVarGDSClass'
pca(gdsobj, eigen.cnt=32)
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
eigen.cnt	An integer indicating how many eigenvalues and eignvectors to return.

Details

Calculates the genetic covariance matrix and finds the eigen decomposition.

Value

A list with two elements:

eigenval	A vector of length eigen.cnt with eigenvalues
eigenvect	A matrix of dimension ("selected samples", eigen.cnt).

Author(s)

Xiuwen Zheng, Stephanie Gogarten

References

Patterson N, Price AL, Reich D (2006) Population structure and eigenanalysis. PLoS Genetics 2:e190.

See Also

SeqVarGDSClass, applyMethod

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
pca <- pca(gds)
pca$eigenval
head(pca$eigenvect)
seqClose(gds)
```

---

pedigree *Pedigree for example data*

Description

Pedigree for example data files in SeqArray.

Usage

```
pedigree
```

Format

A data.frame with the following columns.

`family` Family ID

`individ` Individual ID

`father` Father ID

`mother` Mother ID

`sex` Sex

`sample.id` sample.id in VCF/GDS files

Details

There is one trio in the pedigree.

Source

HapMap

Examples

```
data(pedigree)
head(pedigree)
gds <- seqOpen(seqExampleFileName("gds"))
setdiff(seqGetData(gds, "sample.id"), pedigree$sample.id)
seqClose(gds)
```

---

refFrac *Reference allele fraction*

Description

Calculate fraction of reference allele reads

Usage

```
## S4 method for signature 'SeqVarGDSClass'
refFrac(gdsobj, use.names=TRUE, parallel=FALSE)
## S4 method for signature 'SeqVarGDSClass'
refFracOverHets(gdsobj, FUN=mean, use.names=TRUE, parallel=FALSE)
## S4 method for signature 'SeqVarGDSClass'
refFracPlot(gdsobj, variant.id, highlight=NULL, ...)
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
FUN	The function to apply over heterozygote calls (mean or median).
use.names	A logical indicating whether to assign variant or samples IDs as names of the output vector.
parallel	Logical, numeric, or other value to control parallel processing; see seqParallel for details.
variant.id	A vector of variant.ids to plot.
highlight	A list of sample.ids to highlight with sequential integers on each plot
...	Additional arguments passed to plot.

Details

The variable "annotation/format/AD" (allelic depth) is required to compute the reference allele fraction.

refFracPlot generates plots of total unfiltered depth (sum over "AD" for all alleles) versus reference allele fraction. Points are color-coded by called genotype: teal = reference homozygote, orange = heterozygote including the reference allele, fuschia = heterozygote with two alternate alleles, purple = alternate homozygote, black = missing. Darker colors indicate a higher density of points. Vertical black line is at 0.5, vertical orange line is the median reference allele fraction for ref/alt heterozygotes. Values significantly different from 0.5 (after applying a Bonferroni correction) are plotted with triangles.

Value

refFrac returns a sample by variant array of the reference allele fraction, defined as ref_depth / total_depth.

refFracOverHets returns the mean (or other function, e.g. median) of reference allele depth (per variant) over all samples called as heterozygotes.

Author(s)

Stephanie Gogarten

See Also

SeqVarGDSClass, applyMethod

Examples

```
gdsfile <- system.file("extdata", "hapmap_exome_chr22.gds", package="SeqVarTools")
gds <- seqOpen(gdsfile)
RF <- refFrac(gds)
dim(RF)
samples <- seqGetData(gds, "sample.id")
refFracPlot(gds, variant.id=5:6,
            highlight=list(samples[2:3], samples[4:5]))
seqClose(gds)
```

`regression` *Linear or logistic regression*

Description

Run linear or logistic regression on variants

Usage

```
## S4 method for signature 'SeqVarData'
regression(gdsobj, outcome, covar=NULL,
    model.type=c("linear", "logistic", "firth"),
    parallel=FALSE)
```

Arguments

`gdsobj`	A `SeqVarData` object
`outcome`	A character string with the name of the column in `sampleData(gdsobj)` containing the outcome variable
`covar`	A character vector with the name of the column(s) in `sampleData(gdsobj)` containing the covariates
`model.type`	the type of model to be run. "linear" uses `lm`, "logistic" uses `glm` with `family=binomial()`, and "firth" uses `logistf`.
`parallel`	Logical, numeric, or other value to control parallel processing; see `seqParallel` for details.

Details

`regression` tests the additive effect of the reference allele.

Value

a data.frame with the following columns (if applicable):

`variant.id`	variant identifier
`n`	number of samples with non-missing data
`n0`	number of controls (outcome=0) with non-missing data
`n1`	number of cases (outcome=1) with non-missing data
`freq`	reference allele frequency
`freq0`	reference allele frequency in controls
`freq1`	reference allele frequency in cases
`Est`	beta estimate for genotype
`SE`	standard error of beta estimate for the genotype
`Wald.Stat`	chi-squared test statistic for association

Wald.pval	p-value for association
PPL.Stat	firth only: profile penalized likelihood test statistic for association
PPL.pval	firth only: p-value for association

Author(s)

Stephanie Gogarten

See Also

SeqVarData, seqSetFilter, lm, glm, logistf

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))

## create some phenotype data
library(Biobase)
sample.id <- seqGetData(gds, "sample.id")
n <- length(sample.id)
df <- data.frame(sample.id,
   sex=sample(c("M", "F"), n, replace=TRUE),
   age=sample(18:70, n, replace=TRUE),
   phen=rnorm(n),
   stringsAsFactors=FALSE)
meta <- data.frame(labelDescription=c("sample identifier",
   "sex", "age", "phenotype"), row.names=names(df))
sample.data <- AnnotatedDataFrame(df, meta)
seqData <- SeqVarData(gds, sample.data)

## select samples and variants
seqSetFilter(gds, sample.id=sample.id[1:50], variant.id=1:10)

res <- regression(seqData, outcome="phen", covar=c("sex", "age"))
res
seqClose(gds)
```

SeqVarData *SeqVarData*

Description

Extends SeqVarGDSClass to include annotation for samples and variants.

Details

A SeqVarData object adds an AnnotatedDataFrame for both samples and variants to a SeqVarGDSClass object.

Note that a SeqVarData object must be created using an *unfiltered* SeqVarGDSClass object. The sample.id column in the sampleData AnnotatedDataFrame must exactly match the sample.id node in the GDS file (and similarly for variant.id in variantData). This enables all subsequent filters set on the SeqVarData object to apply to the GDS and the annotation simultaneously.

Constructor

- SeqVarData(gds, sampleData, variantData): Returns a SeqVarData object.
  gds can be either the filename of a sequencing GDS file or an existing SeqVarGDSClass object.
  sampleData must be an AnnotatedDataFrame with a column sample.id matching sample.id in the GDS file. If this argument is missing, a data frame with 0 columns will be created.
  variantData must be an AnnotatedDataFrame with a column variant.id matching variant.id in the GDS file. If this argument is missing, a data frame with 0 columns will be created.

Accessors

- sampleData(x), sampleData(x)<- value: Get or set the AnnotatedDataFrame with sample data. If a sample filter has been applied with seqSetFilter, only selected samples will be returned. value must include all samples.
- variantData(x), variantData(x)<- value: Get or set the AnnotatedDataFrame with variant data. If a variant filter has been applied with seqSetFilter, only selected variants will be returned. value must include all variants.
- granges(x): Return a GRanges object with the columns of variantData as metadata columns.
- validateSex(x): Return the contents of a column named "sex" in sampleData(x), provided the contents are valid (values either "M"/"F" or 1/2, or NA). If the column is missing or invalid, return NULL.

See SeqVarGDSClass for additional methods.

Author(s)

Stephanie Gogarten

See Also

SeqVarGDSClass, seqVCF2GDS, seqOpen, seqGetData, seqSetFilter, seqApply, seqClose

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))

## create sample annotation
library(Biobase)
sample.id <- seqGetData(gds, "sample.id")
sex <- sample(c("M","F"), length(sample.id), replace=TRUE)
phenotype <- rnorm(length(sample.id), mean=10)
```

```
samp <- data.frame(sample.id, sex, phenotype, stringsAsFactors=FALSE)
meta <- data.frame(labelDescription=c("unique sample identifier",
     "sex (M=male, f=female)", "example phenotype"),
      row.names=names(samp), stringsAsFactors=FALSE)
sample.data <- AnnotatedDataFrame(samp, meta)

seqData <- SeqVarData(gds, sample.data)

head(validateSex(seqData))

## add another annotation column
sample.data$site <- sample(letters, length(sample.id), replace=TRUE)
varMetadata(sample.data)["site", "labelDescription"] <- "study site"
sampleData(seqData) <- sample.data

## set a filter
seqSetFilter(seqData, sample.id=sample.id[1:10])
nrow(sampleData(seqData))

seqClose(seqData)
```

`setVariantID` *Change the variant ID of a GDS file*

Description

Replace the variable "variant.id" in a GDS file with a user-supplied unique vector of the same length.

Usage

```
setVariantID(gdsfile, variant.id)
```

Arguments

`gdsfile`	A character string with the file path of a GDS file.
`variant.id`	A vector with the new variant IDs.

Details

A VCF file created by `seqVCF2GDS` creates a variable "variant.id" containing sequential integers to identify each variant. `setVariantID` allows the user to replace these values with something more meaningful. The replacement values in `variant.id` must be unique and have the same length as the original "variant.id" vector.

Using character values for `variant.id` may affect performance for large datasets.

Author(s)

Stephanie Gogarten

See Also

SeqVarGDSClass, seqVCF2GDS

Examples

```
oldfile <- system.file("extdata", "gl_chr1.gds", package="SeqVarTools")
newfile <- tempfile()
file.copy(oldfile, newfile)

gds <- seqOpen(newfile)
rsID <- seqGetData(gds, "annotation/id")
seqClose(gds)

setVariantID(newfile, rsID)
gds <- seqOpen(newfile)
seqGetData(gds, "variant.id")
head(getGenotype(gds))
seqClose(gds)

unlink(newfile)
```

---

titv *Transition/Transversion Ratio*

Description

Calculate transition/transversion ratio overall or by sample

Usage

```
## S4 method for signature 'SeqVarGDSClass'
titv(gdsobj, by.sample=FALSE, use.names=FALSE)
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
by.sample	A logical indicating whether TiTv should be calculated by sample or overall for the entire GDS object.
use.names	A logical indicating whether to assign sample IDs as names of the output vector (if by.sample=TRUE).

Details

If by.sample=FALSE (the default), titv calulates the transition/transversion ratio (TiTv) over all samples.

If by.sample=TRUE, titv calculates TiTv over all variant genotypes (heterozygous or homozygous non-reference) for each sample.

Value

A single value for TiTv if `by.sample=FALSE`. If `by.sample=TRUE`, a numeric vector containing TiTv for each sample.

Author(s)

Stephanie Gogarten

See Also

SeqVarGDSClass, applyMethod, isVariant

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
titv(gds)
titv(gds, by.sample=TRUE)

## apply to a subset of variants
library(GenomicRanges)
chrom <- seqGetData(gds, "chromosome")
pos22 <- seqGetData(gds, "position")[chrom == 22]
ranges <- GRanges(seqnames="22", IRanges(min(pos22), max(pos22)))
applyMethod(gds, titv, ranges)

seqClose(gds)
```

---

variantInfo *Variant info*

Description

Return basic variant info as a data.frame.

Usage

```
## S4 method for signature 'SeqVarGDSClass'
variantInfo(gdsobj, alleles=TRUE, expanded=FALSE)
## S4 method for signature 'SeqVarGDSClass'
expandedVariantIndex(gdsobj)
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
alleles	A logical value for whether to include ref and alt alleles
expanded	A logical value for whether to expand multi-allelic variants with one row per alternate allele.

Details

Variants can be represented in collapsed form, with one row per variant, or in expanded form, with one row per alternate allele for multiallelic variants.

Value

`variantInfo` returns a data.frame with variant.id, chromosome, and position for each variant. If `alleles=TRUE`, the data.frame includes ref and alt. If `expanded=TRUE`, the data.frame includes allele.index, which is 1 for the first alternate allele, 2 for the second alternate, etc.

`expandedVariantIndex` returns an index to transform a vector or matrix from collapsed to expanded form.

Author(s)

Stephanie Gogarten

See Also

SeqVarGDSClass

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
seqSetFilter(gds, variant.sel=1323:1327)
variantInfo(gds, alleles=TRUE)
variantInfo(gds, alleles=TRUE, expanded=TRUE)
expandedVariantIndex(gds)
seqClose(gds)
```

Index

∗ **classes**
  Iterator, 23
  SeqVarData, 33
∗ **datasets**
  pedigree, 29
∗ **manip**
  allele-methods, 3
  alleleFrequency, 4
  alternateAlleleDetection, 5
  applyMethod, 7
  countSingletons, 9
  duplicateDiscordance, 10
  getGenotype, 12
  getVariableLengthData, 14
  heterozygosity, 16
  hwe, 17
  imputedDosage, 19
  inbreedCoeff, 20
  isVariant, 22
  meanBySample, 25
  mendelErr, 26
  missingGenotypeRate, 27
  pca, 28
  refFrac, 30
  regression, 32
  setVariantID, 35
  titv, 36
∗ **methods**
  Iterator, 23
  SeqVarData, 33
∗ **package**
  SeqVarTools-package, 2

allele-methods, 3
alleleCount *(*alleleFrequency*)*, 4
alleleCount,SeqVarData-method *(*alleleFrequency*)*, 4
alleleCount,SeqVarGDSClass-method *(*alleleFrequency*)*, 4
alleleDosage *(*getGenotype*)*, 12
alleleDosage,SeqVarGDSClass,list-method *(*getGenotype*)*, 12
alleleDosage,SeqVarGDSClass,numeric-method *(*getGenotype*)*, 12
alleleFrequency, 4, *10*, *14*, *17*
alleleFrequency,SeqVarData-method *(*alleleFrequency*)*, 4
alleleFrequency,SeqVarGDSClass-method *(*alleleFrequency*)*, 4
altChar *(*allele-methods*)*, 3
altChar,SeqVarGDSClass-method *(*allele-methods*)*, 3
altDosage, *19*
altDosage *(*getGenotype*)*, 12
altDosage,SeqVarGDSClass-method *(*getGenotype*)*, 12
alternateAlleleDetection, 5
alternateAlleleDetection,SeqVarData,SeqVarData-method *(*alternateAlleleDetection*)*, 5
AnnotatedDataFrame, *34*
applyMethod, *3*, *5*, 7, *10*, *14*, *15*, *17*, *18*, *21*, *22*, *26–29*, *31*, *37*
applyMethod,SeqVarGDSClass,function,character-method *(*applyMethod*)*, 7
applyMethod,SeqVarGDSClass,function,GRanges-method *(*applyMethod*)*, 7
applyMethod,SeqVarGDSClass,function,missing-method *(*applyMethod*)*, 7
applyMethod,SeqVarGDSClass,function,numeric-method *(*applyMethod*)*, 7

chromWithPAR, *5*, 8
chromWithPAR,SeqVarGDSClass-method *(*chromWithPAR*)*, 8
countSingletons, 9
countSingletons,SeqVarGDSClass-method *(*countSingletons*)*, 9
currentRanges *(*Iterator*)*, 23
currentRanges,SeqVarBlockIterator-method *(*Iterator*)*, 23

currentRanges,SeqVarListIterator-method (Iterator), 23
currentRanges,SeqVarRangeIterator-method (Iterator), 23
currentVariants (Iterator), 23
currentVariants,SeqVarBlockIterator-method (Iterator), 23
currentVariants,SeqVarIterator-method (Iterator), 23

DataFrame, *24*
duplicateDiscordance, 10
duplicateDiscordance,SeqVarData,missing-method (duplicateDiscordance), 10
duplicateDiscordance,SeqVarData,SeqVarData-method (duplicateDiscordance), 10
duplicateDiscordance,SeqVarIterator,missing-method (duplicateDiscordance), 10

expandedAltDosage (getGenotype), 12
expandedAltDosage,SeqVarGDSClass-method (getGenotype), 12
expandedVariantIndex (variantInfo), 37
expandedVariantIndex,SeqVarGDSClass-method (variantInfo), 37

getGenotype, 12, *22*, *28*
getGenotype,SeqVarGDSClass-method (getGenotype), 12
getGenotypeAlleles (getGenotype), 12
getGenotypeAlleles,SeqVarGDSClass-method (getGenotype), 12
getVariableLengthData, 14
getVariableLengthData,SeqVarGDSClass,character-method (getVariableLengthData), 14
glm, *32, 33*
GRanges, *23*, *34*
granges (SeqVarData), 33
granges,SeqVarData-method (SeqVarData), 33
GRangesList, *24*
GWASExactHW, *18*

heterozygosity, *5*, 16
heterozygosity,SeqVarGDSClass-method (heterozygosity), 16
hethom (heterozygosity), 16
hethom,SeqVarGDSClass-method (heterozygosity), 16
homozygosity (heterozygosity), 16
homozygosity,SeqVarGDSClass-method (heterozygosity), 16
hwe, 17
hwe,SeqVarGDSClass-method (hwe), 17
HWExact, *18*

imputedDosage, 19
imputedDosage,SeqVarGDSClass-method (imputedDosage), 19
inbreedCoeff, 20
inbreedCoeff,SeqVarGDSClass-method (inbreedCoeff), 20
isSNV, 21
isSNV,SeqVarGDSClass-method (isSNV), 21
isVariant, 22, *37*
isVariant,SeqVarGDSClass-method (isVariant), 22
iterateFilter (Iterator), 23
iterateFilter,SeqVarIterator-method (Iterator), 23
Iterator, 23

lastFilter (Iterator), 23
lastFilter,SeqVarIterator-method (Iterator), 23
lastFilter<- (Iterator), 23
lastFilter<-,SeqVarIterator,numeric-method (Iterator), 23
lm, *32, 33*
logistf, *32, 33*

Matrix, *13*
meanBySample, 25
meanBySample,SeqVarGDSClass-method (meanBySample), 25
mendelErr, 26
mendelErr,SeqVarGDSClass-method (mendelErr), 26
minorAlleleCount (alleleFrequency), 4
minorAlleleCount,SeqVarData-method (alleleFrequency), 4
minorAlleleCount,SeqVarGDSClass-method (alleleFrequency), 4
missingGenotypeRate, 27
missingGenotypeRate,SeqVarGDSClass-method (missingGenotypeRate), 27

nAlleles (allele-methods), 3

nAlleles,SeqVarGDSClass-method
(allele-methods), 3

pca, 28
pca,SeqVarGDSClass-method (pca), 28
pedigree, 29
plot, *31*

refChar (allele-methods), 3
refChar,SeqVarGDSClass-method
(allele-methods), 3
refDosage, *19*
refDosage (getGenotype), 12
refDosage,SeqVarGDSClass-method
(getGenotype), 12
refFrac, 30
refFrac,SeqVarGDSClass-method
(refFrac), 30
refFracOverHets (refFrac), 30
refFracOverHets,SeqVarGDSClass-method
(refFrac), 30
refFracPlot (refFrac), 30
refFracPlot,SeqVarGDSClass-method
(refFrac), 30
regression, 32
regression,SeqVarData-method
(regression), 32
resetIterator (Iterator), 23
resetIterator,SeqVarIterator-method
(Iterator), 23

sampleData, *11*
sampleData (SeqVarData), 33
sampleData,SeqVarData-method
(SeqVarData), 33
sampleData<- (SeqVarData), 33
sampleData<-,SeqVarData,AnnotatedDataFrame-method
(SeqVarData), 33
seqApply, *26*, *34*
SeqArray, *2*
seqBlockApply, *13*
seqClose, *34*
seqGetData, *14, 15*, *34*
seqOpen, *34*
seqParallel, *4*, *13*, *15–17*, *20*, *22*, *28*, *31, 32*
seqSetFilter, *6*, *8*, *11*, *14*, *24*, *33, 34*
SeqVarBlockIterator (Iterator), 23
SeqVarBlockIterator-class (Iterator), 23
SeqVarData, *5, 6*, *10–12*, *23, 24*, *32*, 33, *33*
SeqVarData-class (SeqVarData), 33
SeqVarGDSClass, *3–5*, *7–10*, *13–22*, *24–29*,
31, *33, 34*, *36–38*
SeqVarIterator, *12*
SeqVarIterator (Iterator), 23
SeqVarIterator-class (Iterator), 23
SeqVarListIterator (Iterator), 23
SeqVarListIterator-class (Iterator), 23
SeqVarRangeIterator (Iterator), 23
SeqVarRangeIterator-class (Iterator), 23
SeqVarTools (SeqVarTools-package), 2
SeqVarTools-package, 2
SeqVarWindowIterator (Iterator), 23
SeqVarWindowIterator-class (Iterator),
23
seqVCF2GDS, *34–36*
setVariantID, 35
show,SeqVarData-method (SeqVarData), 33
show,SeqVarIterator-method (Iterator),
23
show,SeqVarListIterator-method
(Iterator), 23
show,SeqVarRangeIterator-method
(Iterator), 23

titv, 36
titv,SeqVarGDSClass-method (titv), 36

validateSex (SeqVarData), 33
validateSex,SeqVarData-method
(SeqVarData), 33
variantBlock (Iterator), 23
variantBlock,SeqVarBlockIterator-method
(Iterator), 23
variantData, *24*
variantData (SeqVarData), 33
variantData,SeqVarData-method
(SeqVarData), 33
variantData<- (SeqVarData), 33
variantData<-,SeqVarData,AnnotatedDataFrame-method
(SeqVarData), 33
variantFilter (Iterator), 23
variantFilter,SeqVarIterator-method
(Iterator), 23
variantInfo, 37
variantInfo,SeqVarGDSClass-method
(variantInfo), 37
variantRanges (Iterator), 23

variantRanges,SeqVarListIterator-method
(Iterator), 23
variantRanges,SeqVarRangeIterator-method
(Iterator), 23