Package ‘SeqVarTools’

April 23, 2016

Version 1.8.1

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

Maintainer Stephanie M. Gogarten <sdmorris@u.washington.edu>, Xiuwen
Zheng <zhengx@u.washington.edu>, Adrienne Stilp <amstilp@u.washington.edu>

Depends SeqArray (>= 1.1.1)

Imports methods, stringr, gdsfmt, GenomicRanges, IRanges, S4Vectors,
GWASExactHW, VariantAnnotation, Biobase

Suggests BiocGenerics, BiocStyle, RUnit

License GPL-3

LazyData yes

biocViews SNP, GeneticVariability, Sequencing, Genetics

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

NeedsCompilation no

R topics documented:

SeqVarTools-package 2
allele-methods 3
alleleFrequency 4
alternateAlleleDetection 5
applyMethod 6
duplicateDiscordance 8
getGenotype 10

getVariableLengthData 12
heterozygosity 13
hwe 15
inbreedCoeff 16
isSNV 17
isVariant 18
meanBySample 19
mendelErr 20
missingGenotypeRate 21
pca 22
pedigree 23
refFrac 24
SeqVarData 25
setVariantID 26
titv 27

Index **29**

`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)
```

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.

Details

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

Value

A numeric vector of allele frequencies.

Author(s)

Stephanie Gogarten

See Also

`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 'SeqVarGDSClass,SeqVarGDSClass'
alternateAlleleDetection(gdsobj, gdsobj2, verbose=TRUE)
```

Arguments

`gdsobj`	A `SeqVarGDSClass` object with VCF data.
`gdsobj2`	A `SeqVarGDSClass` object with VCF data to be used as the “gold standard”.
`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 currently matched on sample.id, but support for subjectID matching will likely be added in the future. Variants are matched on position and alleles using bi-allelic SNVs only. No allele flipping is done; the variant must have the same reference and alternate alleles to be considered a match. 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
genoData1	aa	2TP	1TP + 1FP	2FP
	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
gds2 <- seqOpen(gdsfile.2) # gold standard dataset, e.g. array genotyping
res <- alleleDetectionAccuracy(gds1, gds2)

## 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)
```

duplicateDiscordance *Duplicate discordance*

Description

Find discordance rate for duplicate sample pairs

Usage

```
## S4 method for signature 'SeqVarGDSClass,missing'
duplicateDiscordance(gdsobj, samples=NULL, check.phase=FALSE, verbose=TRUE)
## S4 method for signature 'SeqVarGDSClass,SeqVarGDSClass'
duplicateDiscordance(gdsobj, obj2, verbose=TRUE)
```

Arguments

gdsobj	A SeqVarGDSClass object with VCF data.
obj2	A SeqVarGDSClass object with VCF data.
samples	A data.frame with columns (sample.id, subject.id). "sample.id" values should correspond to "sample.id" in gdsobj. "subject.id" should match for duplicate samples.
check.phase	A logical indicating whether phase should be considered when calculating discordance.
verbose	A logical indicating whether to print progress messages.

Details

For calls that involve only one gds file, duplicate discordance is calculated by sample pair and by variant. If there are more than two samples per subject in `samples`, only the first two samples are used and a warning message is printed. If `check.phase=TRUE`, variants with mismatched phase are considered discordant. If `check.phase=FALSE`, phase is ignored.

For calls that invovle two gds files, duplicate discordance is calculated by matching sample pairs (not subjects). Samples are matched using must have the same sample.id in each gds object, so the samples must have the same identifier in both gds files. In the future, support will be added for matching on a subjectID instead of on sample IDs. Variants are matched using position and alleles. Only biallelic SNVs are considered. No flipping is done; the same allele must be the reference in both gds files.

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

Value

For calls involving one gds file, a list with the following elements:

by.variant	A data.frame with the number of discordances for each variant, the number of sample pairs with non-missing data, and the discordance rate (num.discord / num.pair). Row names are variant ids.

`by.subject` A data.frame with the sample ids for each pair, the number of discordances, the number of non-missing variants, and the discordance rate (num.discord / num.var). Row.names are subject.id (as given in `samples`).

For calls invovling two gds files, A data frame with the following columns:

`subjectID` currently, this is the sample ID

`sample.id.1` sample id in the first gds file

`sample.id.2` sample id in the second gds file

`n.variants` the number of non-missing variants 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

`SeqVarGDSClass`, `applyMethod`

Examples

```
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 <- data.frame(subject.id=rep(c("subj1", "subj2"), each=2),
                      sample.id=sample.id[1:4],
                      stringsAsFactors=FALSE)
disc <- duplicateDiscordance(gds, samples=samples)
head(disc$by.variant)
disc$by.subject
seqClose(gds)
```

getGenotype *Get genotype data*

Description

Get matrix of genotype values from a GDS object as VCF-style character strings

Usage

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

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.

`sort` Logical for whether to sort alleles lexographically ("G/T" instead of "T/G").

`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.

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 diploid genotypes (the first two alleles at a given site) are returned.

If the argument `n` to`alleleDosage` 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 diploid genotypes.

getGenotype returns alleles as "0", "1", "2", etc. indicating refernence 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).

alleleDosage returns an integer matrix with the dosage of the specified allele(s) 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(s), and 0 for no copies of the allele(s).

Author(s)

Stephanie Gogarten

See Also

SeqVarGDSClass, applyMethod, seqGetData, seqSetFilter, alleleFrequency

Examples

```
gds <- seqOpen(seqExampleFileName("gds"))
variant.id <- seqGetData(gds, "variant.id")
sample.id <- seqGetData(gds, "sample.id")
seqSetFilter(gds, variant.id=variant.id[1:5],
             sample.id=sample.id[1:10])
getGenotype(gds)
getGenotypeAlleles(gds)
refDosage(gds)
altDosage(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)
```

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.

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,,])

## genotype dosage
ds <- seqGetData(gds, "annotation/format/DS")
names(ds)
ds$length
## 1 value per variant
dim(ds$data)
## 85 samples (rows) and 9 variants (columns)

ds.array <- getVariableLengthData(gds, "annotation/format/DS")
dim(ds.array)
## 85 samples x 9 variants
head(ds.array)

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

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.

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.

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
hhnr <- heterozygosity(gds, margin="by.sample") /
        homozygosity(gds, allele="alt", margin="by.sample")
head(hhnr)

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)
```

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.

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)
```

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)
```

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.

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(x, biallelic=TRUE)
```

Arguments

x	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)
```

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.

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)
```

---

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)
```

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.

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

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.
variant.id	A vector of variant.ids to 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 teal 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)
refFracPlot(gds, variant.id=5)
seqClose(gds)
```

SeqVarData *SeqVarData*

Description

Extends SeqVarGDSClass to include annotation for the samples.

Details

A SeqVarData object adds an AnnotatedDataFrame to a SeqVarGDSClass object.

Constructor

SeqVarData(gds, sampleData): 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.

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.

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)

## 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)
```

Index

*Topic **classes**
    SeqVarData, 25
*Topic **datasets**
    pedigree, 23
*Topic **manip**
    allele-methods, 3
    alleleFrequency, 4
    alternateAlleleDetection, 5
    applyMethod, 6
    duplicateDiscordance, 8
    getGenotype, 10
    getVariableLengthData, 12
    heterozygosity, 13
    hwe, 15
    inbreedCoeff, 16
    isVariant, 18
    meanBySample, 19
    mendelErr, 20
    missingGenotypeRate, 21
    pca, 22
    refFrac, 24
    setVariantID, 26
    titv, 27
*Topic **methods**
    SeqVarData, 25
*Topic **package**
    SeqVarTools-package, 2

allele-methods, 3
alleleDosage (getGenotype), 10
alleleDosage,SeqVarGDSClass,list-method (getGenotype), 10
alleleDosage,SeqVarGDSClass,numeric-method (getGenotype), 10
alleleFrequency, 4, *11*, *14*
alleleFrequency,SeqVarGDSClass-method (alleleFrequency), 4
altChar (allele-methods), 3
altChar,SeqVarGDSClass-method (allele-methods), 3
altDosage (getGenotype), 10
altDosage,SeqVarGDSClass-method (getGenotype), 10
alternateAlleleDetection, 5
alternateAlleleDetection,SeqVarGDSClass,SeqVarGDSClass-met (alternateAlleleDetection), 5
AnnotatedDataFrame, *25*
applyMethod, *3*, *4*, 6, *9*, *11*, *12*, *14*, *15*, *17–19*, *21–23*, *25*, *28*
applyMethod,SeqVarGDSClass,function,character-method (applyMethod), 6
applyMethod,SeqVarGDSClass,function,GRanges-method (applyMethod), 6
applyMethod,SeqVarGDSClass,function,missing-method (applyMethod), 6
applyMethod,SeqVarGDSClass,function,numeric-method (applyMethod), 6

duplicateDiscordance, 8
duplicateDiscordance,SeqVarGDSClass,missing-method (duplicateDiscordance), 8
duplicateDiscordance,SeqVarGDSClass,SeqVarGDSClass-method (duplicateDiscordance), 8

getGenotype, 10, *19*, *22*
getGenotype,SeqVarGDSClass-method (getGenotype), 10
getGenotypeAlleles (getGenotype), 10
getGenotypeAlleles,SeqVarGDSClass-method (getGenotype), 10
getVariableLengthData, 12
getVariableLengthData,SeqVarGDSClass,character-method (getVariableLengthData), 12
GWASExactHW, *15*

heterozygosity, *4*, 13
heterozygosity,SeqVarGDSClass-method (heterozygosity), 13
homozygosity (heterozygosity), 13

homozygosity,SeqVarGDSClass-method (heterozygosity), 13
hwe, 15
hwe,SeqVarGDSClass-method (hwe), 15
HWExact, *15*

inbreedCoeff, 16
inbreedCoeff,SeqVarGDSClass-method (inbreedCoeff), 16
isSNV, 17
isSNV,SeqVarGDSClass-method (isSNV), 17
isVariant, 18, *28*
isVariant,SeqVarGDSClass-method (isVariant), 18

meanBySample, 19
meanBySample,SeqVarGDSClass-method (meanBySample), 19
mendelErr, 20
mendelErr,SeqVarGDSClass-method (mendelErr), 20
missingGenotypeRate, 21
missingGenotypeRate,SeqVarGDSClass-method (missingGenotypeRate), 21

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

pca, 22
pca,SeqVarGDSClass-method (pca), 22
pedigree, 23
plot, *24*

refChar (allele-methods), 3
refChar,SeqVarGDSClass-method (allele-methods), 3
refDosage (getGenotype), 10
refDosage,SeqVarGDSClass-method (getGenotype), 10
refFrac, 24
refFrac,SeqVarGDSClass-method (refFrac), 24
refFracOverHets (refFrac), 24
refFracOverHets,SeqVarGDSClass-method (refFrac), 24
refFracPlot (refFrac), 24
refFracPlot,SeqVarGDSClass-method (refFrac), 24

sampleData (SeqVarData), 25
sampleData,SeqVarData-method (SeqVarData), 25
sampleData<- (SeqVarData), 25
sampleData<-,SeqVarData,AnnotatedDataFrame-method (SeqVarData), 25
seqApply, *19*, *26*
SeqArray, *2*
seqClose, *26*
seqGetData, *11, 12*, *26*
seqOpen, *26*
seqSetFilter, *5*, *7, 8*, *11*, *25, 26*
SeqVarData, 25
SeqVarData-class (SeqVarData), 25
SeqVarGDSClass, *3–12*, *14–28*
SeqVarTools (SeqVarTools-package), 2
SeqVarTools-package, 2
seqVCF2GDS, *26*, *27*
setVariantID, 26

titv, 27
titv,SeqVarGDSClass-method (titv), 27