Package 'SNPRelate'

May 21, 2024

Type Package

Title Parallel Computing Toolset for Relatedness and Principal Component Analysis of SNP Data

Version 1.39.0

Date 2024-04-11

Depends R (>= 2.15), gdsfmt (>= 1.8.3)

LinkingTo gdsfmt

Imports methods

Suggests parallel, Matrix, RUnit, knitr, markdown, rmarkdown, MASS, BiocGenerics

Enhances SeqArray (>= 1.12.0)

Description Genome-wide association studies (GWAS) are widely used to investigate the genetic basis of diseases and traits, but they pose many computational challenges. We developed an R package SNPRelate to provide a binary format for single-nucleotide polymorphism (SNP) data in GWAS utilizing CoreArray Genomic Data Structure (GDS) data files. The GDS format offers the efficient operations specifically designed for integers with two bits, since a SNP could occupy only two bits. SNPRelate is also designed to accelerate two key computations on SNP data using parallel computing for multi-core symmetric multiprocessing computer architectures: Principal Component Analysis (PCA) and relatedness analysis using Identity-By-Descent measures. The SNP GDS format is also used by the GWASTools package with the support of S4 classes and generic functions. The extended GDS format is implemented in the SeqArray package to support the storage of single nucleotide variations (SNVs), insertion/deletion polymorphism (indel) and structural variation calls in whole-genome and whole-exome variant data.

License GPL-3

VignetteBuilder knitr

LazyData true

URL https://github.com/zhengxwen/SNPRelate

BugReports https://github.com/zhengxwen/SNPRelate/issues

biocViews Infrastructure, Genetics, StatisticalMethod, PrincipalComponent

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

git_branch devel

git_last_commit 9da19ab

git_last_commit_date 2024-04-30

Repository Bioconductor 3.20

Date/Publication 2024-05-20

Author Xiuwen Zheng [aut, cre, cph] (<https://orcid.org/0000-0002-1390-0708>), Stephanie Gogarten [ctb], Cathy Laurie [ctb], Bruce Weir [ctb, ths] (<https://orcid.org/0000-0002-4883-1247>)

Maintainer Xiuwen Zheng <zhengx@u.washington.edu>

Contents

`SNPRelate-package` *Parallel Computing Toolset for Genome-Wide Association Studies*

Description

Genome-wide association studies are widely used to investigate the genetic basis of diseases and traits, but they pose many computational challenges. We developed SNPRelate (R package for multi-core symmetric multiprocessing computer architectures) to accelerate two key computations on SNP data: principal component analysis (PCA) and relatedness analysis using identity-by-descent measures. The kernels of our algorithms are written in C/C++ and highly optimized.

Details

Package:	SNPRelate
Type:	Package
License:	GPL version 3
Depends:	gdsfmt (>= 1.0.4)

The genotypes stored in GDS format can be analyzed by the R functions in SNPRelate, which utilize the multi-core feature of machine for a single computer.

Webpage: https://github.com/zhengxwen/SNPRelate, http://corearray.sourceforge.net/

Tutorial: http://corearray.sourceforge.net/tutorials/SNPRelate/

Author(s)

Xiuwen Zheng <zhengxwen@gmail.com>

References

Zheng X, Levine D, Shen J, Gogarten SM, Laurie C, Weir BS. A High-performance Computing Toolset for Relatedness and Principal Component Analysis of SNP Data. Bioinformatics (2012); doi: 10.1093/bioinformatics/bts610

Examples

```
######################################################################
# Convert the PLINK BED file to the GDS file
#

# PLINK BED files
bed.fn <- system.file("extdata", "plinkhapmap.bed.gz", package="SNPRelate")
fam.fn <- system.file("extdata", "plinkhapmap.fam.gz", package="SNPRelate")
bim.fn <- system.file("extdata", "plinkhapmap.bim.gz", package="SNPRelate")

# convert
snpgdsBED2GDS(bed.fn, fam.fn, bim.fn, "HapMap.gds")


######################################################################
# Principal Component Analysis
#

# open
genofile <- snpgdsOpen("HapMap.gds")

RV <- snpgdsPCA(genofile)
plot(RV$eigenvect[,2], RV$eigenvect[,1], xlab="PC 2", ylab="PC 1",
    col=rgb(0,0,150, 50, maxColorValue=255), pch=19)

# close the file
```

```
snpgdsClose(genofile)


############################################################################
# Identity-By-Descent (IBD) Analysis
#

# open
genofile <- snpgdsOpen(snpgdsExampleFileName())

RV <- snpgdsIBDMoM(genofile)
flag <- lower.tri(RV$k0)
plot(RV$k0[flag], RV$k1[flag], xlab="k0", ylab="k1",
    col=rgb(0,0,150, 50, maxColorValue=255), pch=19)
abline(1, -1, col="red", lty=4)

# close the file
snpgdsClose(genofile)


############################################################################
# Identity-By-State (IBS) Analysis
#

# open
genofile <- snpgdsOpen(snpgdsExampleFileName())

RV <- snpgdsIBS(genofile)
m <- 1 - RV$ibs
colnames(m) <- rownames(m) <- RV$sample.id
GeneticDistance <- as.dist(m[1:45, 1:45])
HC <- hclust(GeneticDistance, "ave")
plot(HC)

# close the file
snpgdsClose(genofile)


############################################################################
# Linkage Disequilibrium (LD) Analysis
#

# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

snpset <- read.gdsn(index.gdsn(genofile, "snp.id"))[1:200]
L1 <- snpgdsLDMat(genofile, snp.id=snpset, method="composite", slide=-1)

# plot
image(abs(L1$LD), col=terrain.colors(64))

# close the file
snpgdsClose(genofile)
```

hapmap_geno *SNP genotypes of HapMap samples*

Description

A list object including the following components:

`sample.id` – a vector of sample ids;

`snp.id` – a vector of SNP ids;

`snp.position` – a vector of SNP positions;

`snp.chromosome` – a vector of chromosome indices;

`snp.allele` – a character vector of "reference / non-reference";

`genotype` – a "# of SNPs" X "# of samples" genotype matrix.

Usage

```
hapmap_geno
```

Value

A list

snpgdsAdmixPlot *Plot Ancestry Proportions*

Description

Plot the admixture proportions according to their ancestries.

Usage

```
snpgdsAdmixPlot(propmat, group=NULL, col=NULL, multiplot=TRUE, showgrp=TRUE,
    shownum=TRUE, ylim=TRUE, na.rm=TRUE)
snpgdsAdmixTable(propmat, group, sort=FALSE)
```

Arguments

`propmat`	a sample-by-ancestry matrix of proportion estimates, returned from `snpgdsAdmixProp()`
`group`	a character vector of a factor according to the rows in `propmat`
`col`	specify colors; if `group` is not specified, it is a color for each sample; otherwise specify colors for the groups
`multiplot`	single plot or multiple plots
`showgrp`	show group names in the plot; applicable when `group` is used

shownum	TRUE: show the number of each group on the X-axis in the figure; applicable when group is used
ylim	TRUE: y-axis is limited to [0, 1]; FALSE: ylim <- range(propmat); a 2-length numeric vector: ylim used in plot()
na.rm	TRUE: remove the sample(s) according to the missing value(s) in group
sort	TRUE: rearranges the rows of proportion matrices into descending order

Details

The minor allele frequency and missing rate for each SNP passed in snp.id are calculated over all the samples in sample.id.

Value

snpgdsAdmixPlot(): none.

snpgdsAdmixTable(): a list of data.frame consisting of group, num, mean, sd, min, max

Author(s)

Xiuwen Zheng

References

Zheng X, Weir BS. Eigenanalysis on SNP Data with an Interpretation of Identity by Descent. Theoretical Population Biology. 2015 Oct 23. pii: S0040-5809(15)00089-1. doi: 10.1016/j.tpb.2015.09.004.

See Also

snpgdsEIGMIX, snpgdsAdmixProp

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# get population information
#   or pop_code <- scan("pop.txt", what=character())
#   if it is stored in a text file "pop.txt"
pop_code <- read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))

# get sample id
samp.id <- read.gdsn(index.gdsn(genofile, "sample.id"))

# run eigen-analysis
RV <- snpgdsEIGMIX(genofile)

# define groups
groups <- list(CEU = samp.id[pop_code == "CEU"],
    YRI = samp.id[pop_code == "YRI"],
    CHB = samp.id[is.element(pop_code, c("HCB", "JPT"))])
```

```
prop <- snpgdsAdmixProp(RV, groups=groups, bound=TRUE)

# draw
snpgdsAdmixPlot(prop, group=pop_code)

# use user-defined colors for the groups
snpgdsAdmixPlot(prop, group=pop_code, multiplot=FALSE, col=c(3,2,4))

snpgdsAdmixTable(prop, group=pop_code)

# close the genotype file
snpgdsClose(genofile)
```

---

snpgdsAdmixProp *Estimate ancestral proportions from the eigen-analysis*

Description

Estimate ancestral (admixture) proportions based on the eigen-analysis.

Usage

```
snpgdsAdmixProp(eigobj, groups, bound=FALSE)
```

Arguments

eigobj	an object of snpgdsEigMixClass from snpgdsEIGMIX, or an object of snpgdsPCAClass from snpgdsPCA
groups	a list of sample IDs, such like groups = list(CEU = c("NA0101", "NA1022", ...), YRI = c("NAxxxx", ...), Asia = c("NA1234", ...))
bound	if TRUE, the estimates are bounded in [0, 1], and the sum of proportions is one; bound=FALSE for unbiased estimates

Details

The minor allele frequency and missing rate for each SNP passed in snp.id are calculated over all the samples in sample.id.

Value

Return a matrix of ancestral proportions with rows for study individuals (rownames() is sample ID).

Author(s)

Xiuwen Zheng

References

Zheng X, Weir BS. Eigenanalysis on SNP Data with an Interpretation of Identity by Descent. Theoretical Population Biology. 2015 Oct 23. pii: S0040-5809(15)00089-1. doi: 10.1016/j.tpb.2015.09.004. [Epub ahead of print]

See Also

snpgdsEIGMIX, snpgdsPCA, snpgdsAdmixPlot

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# get population information
#   or pop_code <- scan("pop.txt", what=character())
#   if it is stored in a text file "pop.txt"
pop_code <- read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))

# get sample id
samp.id <- read.gdsn(index.gdsn(genofile, "sample.id"))

# run eigen-analysis
RV <- snpgdsEIGMIX(genofile)

# eigenvalues
RV$eigenval

# make a data.frame
tab <- data.frame(sample.id = samp.id, pop = factor(pop_code),
    EV1 = RV$eigenvect[,1],    # the first eigenvector
    EV2 = RV$eigenvect[,2],    # the second eigenvector
    stringsAsFactors = FALSE)
head(tab)

# draw
plot(tab$EV2, tab$EV1, col=as.integer(tab$pop),
    xlab="eigenvector 2", ylab="eigenvector 1")
legend("bottomleft", legend=levels(tab$pop), pch="o", col=1:4)


# define groups
groups <- list(CEU = samp.id[pop_code == "CEU"],
    YRI = samp.id[pop_code == "YRI"],
    CHB = samp.id[is.element(pop_code, c("HCB", "JPT"))])

prop <- snpgdsAdmixProp(RV, groups=groups)
head(prop)

# draw
plot(prop[, "YRI"], prop[, "CEU"], col=as.integer(tab$pop),
    xlab = "Admixture Proportion from YRI",
```

```
    ylab = "Admixture Proportion from CEU")
abline(v=0, col="gray25", lty=2)
abline(h=0, col="gray25", lty=2)
abline(a=1, b=-1, col="gray25", lty=2)
legend("topright", legend=levels(tab$pop), pch="o", col=1:4)


# draw
snpgdsAdmixPlot(prop, group=pop_code)


# close the genotype file
snpgdsClose(genofile)
```

`snpgdsAlleleSwitch` *Allele-switching*

Description

Switch alleles according to the reference if needed.

Usage

```
snpgdsAlleleSwitch(gdsobj, A.allele, verbose=TRUE)
```

Arguments

`gdsobj`	an object of class `SNPGDSFileClass`, a SNP GDS file
`A.allele`	characters, referring to A allele
`verbose`	if TRUE, show information

Value

A logical vector with `TRUE` indicating allele-switching and `NA` when it is unable to determine. `NA` occurs when `A.allele = NA` or `A.allele` is not in the list of alleles.

Author(s)

Xiuwen Zheng

Examples

```
# the file name of SNP GDS
(fn <- snpgdsExampleFileName())

# copy the file
file.copy(fn, "test.gds", overwrite=TRUE)

# open the SNP GDS file
```

```
genofile <- snpgdsOpen("test.gds", readonly=FALSE)

# allelic information
allele <- read.gdsn(index.gdsn(genofile, "snp.allele"))
allele.list <- strsplit(allele, "/")

A.allele <- sapply(allele.list, function(x) { x[1] })
B.allele <- sapply(allele.list, function(x) { x[2] })

set.seed(1000)
flag <- rep(FALSE, length(A.allele))
flag[sample.int(length(A.allele), 50, replace=TRUE)] <- TRUE

A.allele[flag] <- B.allele[flag]
A.allele[sample.int(length(A.allele), 10, replace=TRUE)] <- NA
table(A.allele, exclude=NULL)


# allele switching
z <- snpgdsAlleleSwitch(genofile, A.allele)

table(z, exclude=NULL)


# close the file
snpgdsClose(genofile)


# delete the temporary file
unlink("test.gds", force=TRUE)
```

`snpgdsApartSelection` *Select SNPs with a basepair distance*

Description

Randomly selects SNPs for which each pair is at least as far apart as the specified basepair distance.

Usage

```
snpgdsApartSelection(chromosome, position, min.dist=100000,
    max.n.snp.perchr=-1, verbose=TRUE)
```

Arguments

`chromosome`	chromosome codes
`position`	SNP positions in base pair
`min.dist`	A numeric value to specify minimum distance required (in basepairs)

max.n.snp.perchr
: A numeric value specifying the maximum number of SNPs to return per chromosome, "-1" means no number limit

verbose
: if TRUE, show information

Value

A logical vector indicating which SNPs were selected.

Author(s)

Xiuwen Zheng

See Also

snpgdsLDpruning

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())
genofile

chr <- read.gdsn(index.gdsn(genofile, "snp.chromosome"))
pos <- read.gdsn(index.gdsn(genofile, "snp.position"))

set.seed(1000)
flag <- snpgdsApartSelection(chr, pos, min.dist=250000, verbose=TRUE)
table(flag)

# close the genotype file
snpgdsClose(genofile)
```

---

snpgdsBED2GDS *Conversion from PLINK BED to GDS*

Description

Convert a PLINK binary ped file to a GDS file.

Usage

```
snpgdsBED2GDS(bed.fn, fam.fn, bim.fn, out.gdsfn, family=FALSE,
    snpfirstdim=NA, compress.annotation="LZMA_RA", compress.geno="",
    option=NULL, cvt.chr=c("int", "char"), cvt.snpid=c("auto", "int"),
    verbose=TRUE)
```

Arguments

`bed.fn`	the file name of binary file, genotype information
`fam.fn`	the file name of first six columns of `".ped"`; if it is missing, ".fam" is added to `bed.fn`
`bim.fn`	the file name of extended MAP file: two extra columns = allele names; if it is missing, ".bim" is added to `bim.fn`
`out.gdsfn`	the output file name of GDS file
`family`	if `TRUE`, to include family information in the sample annotation
`snpfirstdim`	if TRUE, genotypes are stored in the individual-major mode, (i.e, list all SNPs for the first individual, and then list all SNPs for the second individual, etc); `NA`, the dimension is determined by the BED file
`compress.annotation`	the compression method for the GDS variables, except "genotype"; optional values are defined in the function `add.gdsn`
`compress.geno`	the compression method for "genotype"; optional values are defined in the function `add.gdsn`
`option`	`NULL` or an object from `snpgdsOption`, see details
`cvt.chr`	`"int"` – chromosome code in the GDS file is integer; `"char"` – chromosome code in the GDS file is character
`cvt.snpid`	`"int"` – to create an integer `snp.id` starting from 1; `"auto"` – if SNP IDs in the PLINK file are not unique, to create an an integer `snp.id`, otherwise to use SNP IDs for `snp.id`
`verbose`	if TRUE, show information

Details

GDS – Genomic Data Structures, the extended file name used for storing genetic data, and the file format is used in the gdsfmt package.

BED – the PLINK binary ped format.

The user could use `option` to specify the range of code for autosomes. For humans there are 22 autosomes (from 1 to 22), but dogs have 38 autosomes. Note that the default settings are used for humans. The user could call `option = snpgdsOption(autosome.end=38)` for importing the BED file of dog. It also allow define new chromosome coding, e.g., `option = snpgdsOption(Z=27)`.

Value

Return the file name of GDS format with an absolute path.

Author(s)

Xiuwen Zheng

References

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ & Sham PC. 2007. PLINK: a toolset for whole-genome association and population-based linkage analysis. American Journal of Human Genetics, 81.

See Also

snpgdsOption, snpgdsPED2GDS, snpgdsGDS2PED

Examples

```
# PLINK BED files
bed.fn <- system.file("extdata", "plinkhapmap.bed.gz", package="SNPRelate")
fam.fn <- system.file("extdata", "plinkhapmap.fam.gz", package="SNPRelate")
bim.fn <- system.file("extdata", "plinkhapmap.bim.gz", package="SNPRelate")

# convert
snpgdsBED2GDS(bed.fn, fam.fn, bim.fn, "HapMap.gds")

# open
genofile <- snpgdsOpen("HapMap.gds")
genofile

# close
snpgdsClose(genofile)


# delete the temporary file
unlink("HapMap.gds", force=TRUE)
```

---

snpgdsClose *Close the SNP GDS File*

Description

Close the SNP GDS file

Usage

```
snpgdsClose(gdsobj)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file

Details

It is suggested to call snpgdsClose instead of closefn.gds.

Value

None.

Author(s)

Xiuwen Zheng

See Also

snpgdsOpen

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

genofile

# close the file
snpgdsClose(genofile)
```

---

snpgdsCombineGeno *Merge SNP datasets*

Description

To merge GDS files of SNP genotypes into a single GDS file

Usage

```
snpgdsCombineGeno(gds.fn, out.fn, method=c("position", "exact"),
    compress.annotation="ZIP_RA.MAX", compress.geno="ZIP_RA",
    same.strand=FALSE, snpfirstdim=FALSE, verbose=TRUE)
```

Arguments

gds.fn	a character vector of GDS file names to be merged
out.fn	the name of output GDS file
method	"exact": matching by all snp.id, chromosomes, positions and alleles; "position": matching by chromosomes and positions
compress.annotation	the compression method for the variables except genotype
compress.geno	the compression method for the variable genotype
same.strand	if TRUE, assuming the alleles on the same strand
snpfirstdim	if TRUE, genotypes are stored in the individual-major mode, (i.e, list all SNPs for the first individual, and then list all SNPs for the second individual, etc)
verbose	if TRUE, show information

Details

This function calls snpgdsSNPListIntersect internally to determine the common SNPs. Allele definitions are taken from the first GDS file.

Value

None.

Author(s)

Xiuwen Zheng

See Also

snpgdsCreateGeno, snpgdsCreateGenoSet, snpgdsSNPList, snpgdsSNPListIntersect

Examples

```
# get the file name of a gds file
fn <- snpgdsExampleFileName()

f <- snpgdsOpen(fn)
samp_id <- read.gdsn(index.gdsn(f, "sample.id"))
snp_id <- read.gdsn(index.gdsn(f, "snp.id"))
geno <- read.gdsn(index.gdsn(f, "genotype"), start=c(1,1), count=c(-1, 3000))
snpgdsClose(f)


# split the GDS file with different samples
snpgdsCreateGenoSet(fn, "t1.gds", sample.id=samp_id[1:10],
    snp.id=snp_id[1:3000])
snpgdsCreateGenoSet(fn, "t2.gds", sample.id=samp_id[11:30],
    snp.id=snp_id[1:3000])

# combine with different samples
snpgdsCombineGeno(c("t1.gds", "t2.gds"), "test.gds", same.strand=TRUE)
f <- snpgdsOpen("test.gds")
g <- read.gdsn(index.gdsn(f, "genotype"))
snpgdsClose(f)

identical(geno[1:30, ], g)  # TRUE


# split the GDS file with different SNPs
snpgdsCreateGenoSet(fn, "t1.gds", snp.id=snp_id[1:100])
snpgdsCreateGenoSet(fn, "t2.gds", snp.id=snp_id[101:300])

# combine with different SNPs
snpgdsCombineGeno(c("t1.gds", "t2.gds"), "test.gds")
f <- snpgdsOpen("test.gds")
g <- read.gdsn(index.gdsn(f, "genotype"))
snpgdsClose(f)
```

```
identical(geno[, 1:300], g)  # TRUE


# delete the temporary files
unlink(c("t1.gds", "t2.gds", "t3.gds", "t4.gds", "test.gds"), force=TRUE)
```

---

snpgdsCreateGeno *Create a SNP genotype dataset from a matrix*

Description

To create a GDS file of genotypes from a matrix.

Usage

```
snpgdsCreateGeno(gds.fn, genmat, sample.id=NULL, snp.id=NULL, snp.rs.id=NULL,
    snp.chromosome=NULL, snp.position=NULL, snp.allele=NULL, snpfirstdim=TRUE,
    compress.annotation="ZIP_RA.max", compress.geno="", other.vars=NULL)
```

Arguments

gds.fn	the file name of gds
genmat	a matrix of genotypes
sample.id	the sample ids, which should be unique
snp.id	the SNP ids, which should be unique
snp.rs.id	the rs ids for SNPs, which can be not unique
snp.chromosome	the chromosome indices
snp.position	the SNP positions in basepair
snp.allele	the reference/non-reference alleles
snpfirstdim	if TRUE, genotypes are stored in the individual-major mode, (i.e, list all SNPs for the first individual, and then list all SNPs for the second individual, etc)
compress.annotation	the compression method for the variables except genotype
compress.geno	the compression method for the variable genotype
other.vars	a list object storing other variables

Details

There are possible values stored in the variable genmat: 0, 1, 2 and other values. "0" indicates two B alleles, "1" indicates one A allele and one B allele, "2" indicates two A alleles, and other values indicate a missing genotype.

If snpfirstdim is TRUE, then genmat should be "# of SNPs X # of samples"; if snpfirstdim is FALSE, then genmat should be "# of samples X # of SNPs".

The typical variables specified in other.vars are "sample.annot" and "snp.annot", which are data.frame objects.

Value

None.

Author(s)

Xiuwen Zheng

See Also

snpgdsCreateGenoSet, snpgdsCombineGeno

Examples

```
# load data
data(hapmap_geno)

# create a gds file
with(hapmap_geno, snpgdsCreateGeno("test.gds", genmat=genotype,
    sample.id=sample.id, snp.id=snp.id, snp.chromosome=snp.chromosome,
    snp.position=snp.position, snp.allele=snp.allele, snpfirstdim=TRUE))

# open the gds file
genofile <- snpgdsOpen("test.gds")

RV <- snpgdsPCA(genofile)
plot(RV$eigenvect[,2], RV$eigenvect[,1], xlab="PC 2", ylab="PC 1")

# close the file
snpgdsClose(genofile)
```

---

snpgdsCreateGenoSet *Create a SNP genotype dataset from a GDS file*

Description

To create a GDS file of genotypes from a specified GDS file.

Usage

```
snpgdsCreateGenoSet(src.fn, dest.fn, sample.id=NULL, snp.id=NULL,
    snpfirstdim=NULL, compress.annotation="ZIP_RA.max", compress.geno="",
    verbose=TRUE)
```

Arguments

src.fn	the file name of a specified GDS file
dest.fn	the file name of output GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
snpfirstdim	if TRUE, genotypes are stored in the individual-major mode, (i.e, list all SNPs for the first individual, and then list all SNPs for the second individual, etc)
compress.annotation	the compression method for the variables except genotype
compress.geno	the compression method for the variable genotype
verbose	if TRUE, show information

Value

None.

Author(s)

Xiuwen Zheng

See Also

snpgdsCreateGeno, snpgdsCombineGeno

Examples

```
# open an example dataset (HapMap)
(genofile <- snpgdsOpen(snpgdsExampleFileName()))
# +    [  ] *
# |--+ sample.id   { VStr8 279 ZIP(29.9%), 679B }
# |--+ snp.id   { Int32 9088 ZIP(34.8%), 12.3K }
# |--+ snp.rs.id   { VStr8 9088 ZIP(40.1%), 36.2K }
# |--+ snp.position   { Int32 9088 ZIP(94.7%), 33.6K }
# |--+ snp.chromosome   { UInt8 9088 ZIP(0.94%), 85B } *
# |--+ snp.allele   { VStr8 9088 ZIP(11.3%), 4.0K }
# |--+ genotype   { Bit2 279x9088, 619.0K } *
# \--+ sample.annot   [ data.frame ] *
#    |--+ family.id   { VStr8 279 ZIP(34.4%), 514B }
#    |--+ father.id   { VStr8 279 ZIP(31.5%), 220B }
#    |--+ mother.id   { VStr8 279 ZIP(30.9%), 214B }
#    |--+ sex   { VStr8 279 ZIP(17.0%), 95B }
#    \--+ pop.group   { VStr8 279 ZIP(6.18%), 69B }

set.seed(1000)
snpset <- unlist(snpgdsLDpruning(genofile))
length(snpset)
# 6547

# close the file
```

```
snpgdsClose(genofile)

snpgdsCreateGenoSet(snpgdsExampleFileName(), "test.gds", snp.id=snpset)

##################################################
# check

(gfile <- snpgdsOpen("test.gds"))
# +    [  ] *
# |--+ sample.id   { Str8 279 ZIP_ra(31.2%), 715B }
# |--+ snp.id   { Int32 6547 ZIP_ra(34.9%), 8.9K }
# |--+ snp.rs.id   { Str8 6547 ZIP_ra(41.5%), 27.1K }
# |--+ snp.position   { Int32 6547 ZIP_ra(94.9%), 24.3K }
# |--+ snp.chromosome   { Int32 6547 ZIP_ra(0.45%), 124B }
# |--+ snp.allele   { Str8 6547 ZIP_ra(11.5%), 3.0K }
# \--+ genotype   { Bit2 279x6547, 446.0K } *

# close the file
snpgdsClose(gfile)


unlink("test.gds", force=TRUE)
```

`snpgdsCutTree` *Determine clusters of individuals*

Description

To determine sub groups of individuals using a specified dendrogram from hierarchical cluster analysis

Usage

```
snpgdsCutTree(hc, z.threshold=15, outlier.n=5, n.perm = 5000, samp.group=NULL,
    col.outlier="red", col.list=NULL, pch.outlier=4, pch.list=NULL,
    label.H=FALSE, label.Z=TRUE, verbose=TRUE)
```

Arguments

`hc`	an object of `snpgdsHCluster`
`z.threshold`	the threshold of Z score to determine whether split the node or not
`outlier.n`	the cluster with size less than or equal to `outlier.n` is considered as outliers
`n.perm`	the times for permutation
`samp.group`	if `NULL`, determine groups by Z score; if a vector of factor, assign each individual in dendrogram with respect to `samp.group`
`col.outlier`	the color of outlier
`col.list`	the list of colors for different clusters

pch.outlier	plotting 'character' for outliers
pch.list	plotting 'character' for different clusters
label.H	if TRUE, plotting heights in a dendrogram
label.Z	if TRUE, plotting Z scores in a dendrogram
verbose	if TRUE, show information

Details

The details will be described in future.

Value

Return a list:

sample.id	the sample ids used in the analysis
z.threshold	the threshold of Z score to determine whether split the node or not
outlier.n	the cluster with size less than or equal to outlier.n is considered as outliers
samp.order	the order of samples in the dendrogram
samp.group	a vector of factor, indicating the group of each individual
dmat	a matrix of pairwise group dissimilarity
dendrogram	the dendrogram of individuals
merge	a data.frame of (z, n1, n2) describing each combination: z, the Z score; n1, the size of the first cluster; n2, the size of the second cluster
clust.count	the counts for clusters

Author(s)

Xiuwen Zheng

See Also

snpgdsHCluster, snpgdsDrawTree, snpgdsIBS, snpgdsDiss

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

pop.group <- as.factor(read.gdsn(index.gdsn(
    genofile, "sample.annot/pop.group")))
pop.level <- levels(pop.group)

diss <- snpgdsDiss(genofile)
hc <- snpgdsHCluster(diss)

# close the genotype file
snpgdsClose(genofile)
```

```
#############################################################
# cluster individuals
#

set.seed(100)
rv <- snpgdsCutTree(hc, label.H=TRUE, label.Z=TRUE)

# the distribution of Z scores
snpgdsDrawTree(rv, type="z-score", main="HapMap Phase II")

# draw dendrogram
snpgdsDrawTree(rv, main="HapMap Phase II",
    edgePar=list(col=rgb(0.5,0.5,0.5, 0.75), t.col="black"))


#############################################################
# or cluster individuals by ethnic information
#

rv2 <- snpgdsCutTree(hc, samp.group=pop.group)

# cluster individuals by Z score, specifying 'clust.count'
snpgdsDrawTree(rv2, rv$clust.count, main="HapMap Phase II",
    edgePar = list(col=rgb(0.5,0.5,0.5, 0.75), t.col="black"),
    labels = c("YRI", "CHB/JPT", "CEU"), y.label=0.1)
legend("bottomleft", legend=levels(pop.group), col=1:nlevels(pop.group),
    pch=19, ncol=4, bg="white")



#############################################################
# zoom in ...
#

snpgdsDrawTree(rv2, rv$clust.count, dend.idx = c(1),
    main="HapMap Phase II -- YRI",
    edgePar=list(col=rgb(0.5,0.5,0.5, 0.75), t.col="black"),
    y.label.kinship=TRUE)

snpgdsDrawTree(rv2, rv$clust.count, dend.idx = c(2,2),
    main="HapMap Phase II -- CEU",
    edgePar=list(col=rgb(0.5,0.5,0.5, 0.75), t.col="black"),
    y.label.kinship=TRUE)

snpgdsDrawTree(rv2, rv$clust.count, dend.idx = c(2,1),
    main="HapMap Phase II -- CHB/JPT",
    edgePar=list(col=rgb(0.5,0.5,0.5, 0.75), t.col="black"),
    y.label.kinship=TRUE)
```

snpgdsDiss *Individual dissimilarity analysis*

Description

Calculate the individual dissimilarities for each pair of individuals.

Usage

```
snpgdsDiss(gdsobj, sample.id=NULL, snp.id=NULL, autosome.only=TRUE,
    remove.monosnp=TRUE, maf=NaN, missing.rate=NaN, num.thread=1, verbose=TRUE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
autosome.only	if TRUE, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
remove.monosnp	if TRUE, remove monomorphic SNPs
maf	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
missing.rate	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
num.thread	the number of (CPU) cores used; if NA, detect the number of cores automatically
verbose	if TRUE, show information

Details

The minor allele frequency and missing rate for each SNP passed in snp.id are calculated over all the samples in sample.id.

snpgdsDiss() returns 1 - beta_ij which is formally described in Weir&Goudet (2017).

Value

Return a class "snpgdsDissClass":

sample.id	the sample ids used in the analysis
snp.id	the SNP ids used in the analysis
diss	a matrix of individual dissimilarity

Author(s)

Xiuwen Zheng

References

Zheng, Xiuwen. 2013. Statistical Prediction of HLA Alleles and Relatedness Analysis in Genome-Wide Association Studies. PhD dissertation, the department of Biostatistics, University of Washington.

Weir BS, Zheng X. SNPs and SNVs in Forensic Science. 2015. Forensic Science International: Genetics Supplement Series.

Weir BS, Goudet J. A Unified Characterization of Population Structure and Relatedness. Genetics. 2017 Aug;206(4):2085-2103. doi: 10.1534/genetics.116.198424.

See Also

snpgdsHCluster

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

pop.group <- as.factor(read.gdsn(index.gdsn(
    genofile, "sample.annot/pop.group")))
pop.level <- levels(pop.group)

diss <- snpgdsDiss(genofile)
diss
hc <- snpgdsHCluster(diss)
names(hc)
plot(hc$dendrogram)

# close the genotype file
snpgdsClose(genofile)


# split
set.seed(100)
rv <- snpgdsCutTree(hc, label.H=TRUE, label.Z=TRUE)

# draw dendrogram
snpgdsDrawTree(rv, main="HapMap Phase II",
    edgePar=list(col=rgb(0.5,0.5,0.5, 0.75), t.col="black"))
```

---

snpgdsDrawTree *Draw a dendrogram*

Description

To draw a dendrogram or the distribution of Z scores

Usage

```
snpgdsDrawTree(obj, clust.count=NULL, dend.idx=NULL,
    type=c("dendrogram", "z-score"), yaxis.height=TRUE, yaxis.kinship=TRUE,
    y.kinship.baseline=NaN, y.label.kinship=FALSE, outlier.n=NULL,
    shadow.col=c(rgb(0.5, 0.5, 0.5, 0.25), rgb(0.5, 0.5, 0.5, 0.05)),
    outlier.col=rgb(1, 0.50, 0.50, 0.5), leaflab="none",
    labels=NULL, y.label=0.2, ...)
```

Arguments

obj	an object returned by snpgdsCutTree
clust.count	the counts for clusters, drawing shadows
dend.idx	the index of sub tree, plot obj$dendrogram[[dend.idx]], or NULL for the whole tree
type	"dendrogram", draw a dendrogram; or "z-score", draw the distribution of Z score
yaxis.height	if TRUE, draw the left Y axis: height of tree
yaxis.kinship	if TRUE, draw the right Y axis: kinship coefficient
y.kinship.baseline	the baseline value of kinship; if NaN, it is the height of the first split from top in a dendrogram; only works when yaxis.kinship = TRUE
y.label.kinship	if TRUE, show 'PO/FS' etc on the right axis
outlier.n	the cluster with size less than or equal to outlier.n is considered as outliers; if NULL, let outlier.n = obj$outlier.n
shadow.col	two colors for shadow
outlier.col	the colors for outliers
leaflab	a string specifying how leaves are labeled. The default "perpendicular" write text vertically (by default). "textlike" writes text horizontally (in a rectangle), and "none" suppresses leaf labels.
labels	the legend for different regions
y.label	y positions of labels
...	Arguments to be passed to the method "plot(, ...)", such as graphical parameters.

Details

The details will be described in future.

Value

None.

Author(s)

Xiuwen Zheng

See Also

snpgdsCutTree

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

pop.group <- as.factor(read.gdsn(index.gdsn(
    genofile, "sample.annot/pop.group")))
pop.level <- levels(pop.group)

diss <- snpgdsDiss(genofile)
hc <- snpgdsHCluster(diss)

# close the genotype file
snpgdsClose(genofile)


# split
set.seed(100)
rv <- snpgdsCutTree(hc, label.H=TRUE, label.Z=TRUE)

# draw dendrogram
snpgdsDrawTree(rv, main="HapMap Phase II",
    edgePar=list(col=rgb(0.5,0.5,0.5, 0.75), t.col="black"))
```

---

snpgdsEIGMIX *Eigen-analysis on SNP genotype data*

Description

Eigen-analysis on IBD matrix based SNP genotypes.

Usage

```
snpgdsEIGMIX(gdsobj, sample.id=NULL, snp.id=NULL, autosome.only=TRUE,
    remove.monosnp=TRUE, maf=NaN, missing.rate=NaN, num.thread=1L,
    eigen.cnt=32L, diagadj=TRUE, ibdmat=FALSE, verbose=TRUE)
## S3 method for class 'snpgdsEigMixClass'
plot(x, eig=c(1L,2L), ...)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used

autosome.only	if TRUE, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
remove.monosnp	if TRUE, remove monomorphic SNPs
maf	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
missing.rate	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
num.thread	the number of (CPU) cores used; if NA, detect the number of cores automatically
eigen.cnt	output the number of eigenvectors; if eigen.cnt < 0, returns all eigenvectors; if eigen.cnt==0, no eigen calculation
diagadj	TRUE for diagonal adjustment by default
ibdmat	if TRUE, returns the IBD matrix
verbose	if TRUE, show information
x	a snpgdsEigMixClass object
eig	indices of eigenvectors, like 1:2 or 1:4
...	the arguments passed to or from other methods, like pch, col

Value

Return a snpgdsEigMixClass object, and it is a list:

sample.id	the sample ids used in the analysis
snp.id	the SNP ids used in the analysis
eigenval	eigenvalues
eigenvect	eigenvactors, "# of samples" x "eigen.cnt"
afreq	allele frequencies
ibd	the IBD matrix when ibdmat=TRUE
diagadj	the argument diagadj

Author(s)

Xiuwen Zheng

References

Zheng X, Weir BS. Eigenanalysis on SNP Data with an Interpretation of Identity by Descent. Theoretical Population Biology. 2016 Feb;107:65-76. doi: 10.1016/j.tpb.2015.09.004

See Also

snpgdsAdmixProp, snpgdsAdmixPlot, snpgdsPCA, snpgdsPCASNPLoading, snpgdsPCASampLoading

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# get population information
#   or pop_code <- scan("pop.txt", what=character())
#   if it is stored in a text file "pop.txt"
pop_code <- read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))

# get sample id
samp.id <- read.gdsn(index.gdsn(genofile, "sample.id"))

# run eigen-analysis
RV <- snpgdsEIGMIX(genofile)
RV

# eigenvalues
RV$eigenval

# make a data.frame
tab <- data.frame(sample.id = samp.id, pop = factor(pop_code),
    EV1 = RV$eigenvect[,1],    # the first eigenvector
    EV2 = RV$eigenvect[,2],    # the second eigenvector
    stringsAsFactors = FALSE)
head(tab)

# draw
plot(tab$EV2, tab$EV1, col=as.integer(tab$pop),
    xlab="eigenvector 2", ylab="eigenvector 1")
legend("topleft", legend=levels(tab$pop), pch="o", col=1:4)


# define groups
groups <- list(CEU = samp.id[pop_code == "CEU"],
    YRI = samp.id[pop_code == "YRI"],
    CHB = samp.id[is.element(pop_code, c("HCB", "JPT"))])

prop <- snpgdsAdmixProp(RV, groups=groups)

# draw
plot(prop[, "YRI"], prop[, "CEU"], col=as.integer(tab$pop),
    xlab = "Admixture Proportion from YRI",
    ylab = "Admixture Proportion from CEU")
abline(v=0, col="gray25", lty=2)
abline(h=0, col="gray25", lty=2)
abline(a=1, b=-1, col="gray25", lty=2)
legend("topright", legend=levels(tab$pop), pch="o", col=1:4)


# close the genotype file
snpgdsClose(genofile)
```

snpgdsErrMsg *Get the last error information*

Description

Return the last error message.

Usage

```
snpgdsErrMsg()
```

Value

Characters

Author(s)

Xiuwen Zheng

Examples

```
snpgdsErrMsg()
```

snpgdsExampleFileName *Example GDS file*

Description

Return the file name of example data

Usage

```
snpgdsExampleFileName()
```

Details

A GDS genotype file was created from a subset of HapMap Phase II dataset consisting of 270 individuals and duplicates.

Value

Characters

Author(s)

Xiuwen Zheng

Examples

```
snpgdsExampleFileName()
```

SNPGDSFileClass *SNPGDSFileClass*

Description

A SNPGDSFileClass object provides access to a GDS file containing genome-wide SNP data. It extends the class gds.class in the gdsfmt package.

Author(s)

Xiuwen Zheng

See Also

snpgdsOpen, snpgdsClose

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())
genofile

class(genofile)
# "SNPGDSFileClass" "gds.class"

# close the file
snpgdsClose(genofile)
```

---

snpgdsFst *F-statistics (fixation indices)*

Description

Calculate relatedness measures F-statistics (also known as fixation indices) for given populations

Usage

```
snpgdsFst(gdsobj, population, method=c("W&C84", "W&H02"), sample.id=NULL,
    snp.id=NULL, autosome.only=TRUE, remove.monosnp=TRUE, maf=NaN,
    missing.rate=NaN, with.id=FALSE, verbose=TRUE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
population	a factor, indicating population information for each individual
method	"W&C84" – Fst estimator in Weir & Cockerham 1984 (by default), "W&H02" – relative beta estimator in Weir & Hill 2002, see details
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
autosome.only	if TRUE, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
remove.monosnp	if TRUE, remove monomorphic SNPs
maf	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
missing.rate	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
with.id	if TRUE, the returned value with sample.id and sample.id
verbose	if TRUE, show information

Details

The minor allele frequency and missing rate for each SNP passed in snp.id are calculated over all the samples in sample.id.

The "W&H02" option implements the calculation in Buckleton et. al. 2016.

Value

Return a list:

sample.id	the sample ids used in the analysis
snp.id	the SNP ids used in the analysis
Fst	weighted Fst estimate
MeanFst	the average of Fst estimates across SNPs
FstSNP	a vector of Fst for each SNP
Beta	Beta matrix

Author(s)

Xiuwen Zheng

References

Weir, BS. & Cockerham, CC. Estimating F-statistics for the analysis of population structure. (1984).

Weir, BS. & Hill, WG. Estimating F-statistics. Annual review of genetics 36, 721-50 (2002).

Population-specific FST values for forensic STR markers: A worldwide survey. Buckleton J, Curran J, Goudet J, Taylor D, Thiery A, Weir BS. Forensic Sci Int Genet. 2016 Jul;23:91-100. doi: 10.1016/j.fsigen.2016.03.004.

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

group <- as.factor(read.gdsn(index.gdsn(
    genofile, "sample.annot/pop.group")))

# Fst estimation
v <- snpgdsFst(genofile, population=group, method="W&C84")
v$Fst
v$MeanFst
summary(v$FstSNP)

# or
v <- snpgdsFst(genofile, population=group, method="W&H02")
v$Fst
v$MeanFst
v$Beta
summary(v$FstSNP)

# close the genotype file
snpgdsClose(genofile)
```

---

snpgdsGDS2BED *Conversion from GDS to PLINK BED*

Description

Convert a GDS file to a PLINK binary ped (BED) file.

Usage

```
snpgdsGDS2BED(gdsobj, bed.fn, sample.id=NULL, snp.id=NULL, snpfirstdim=NULL,
    verbose=TRUE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file; or characters, the file name of GDS
bed.fn	the file name of output, without the filename extension ".bed"
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
snpfirstdim	if TRUE, genotypes are stored in the individual-major mode, (i.e, list all SNPs for the first individual, and then list all SNPs for the second individual, etc); if NULL, determine automatically
verbose	if TRUE, show information

Details

GDS – Genomic Data Structures, the extended file name used for storing genetic data, and the file format used in the gdsfmt package.

BED – the PLINK binary ped format.

Value

None.

Author(s)

Xiuwen Zheng

References

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ & Sham PC. 2007. PLINK: a toolset for whole-genome association and population-based linkage analysis. American Journal of Human Genetics, 81.

http://corearray.sourceforge.net/

See Also

snpgdsBED2GDS, snpgdsGDS2PED

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

snpset <- snpgdsSelectSNP(genofile, missing.rate=0.95)
snpgdsGDS2BED(genofile, bed.fn="test", snp.id=snpset)

# close the genotype file
snpgdsClose(genofile)


# delete the temporary files
unlink(c("test.bed", "test.bim", "test.fam"), force=TRUE)
```

---

snpgdsGDS2Eigen *Conversion from GDS to Eigen (EIGENSTRAT)*

Description

Convert a GDS file to an EIGENSTRAT file.

Usage

```
snpgdsGDS2Eigen(gdsobj, eigen.fn, sample.id=NULL, snp.id=NULL, verbose=TRUE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
eigen.fn	the file name of EIGENSTRAT
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
verbose	if TRUE, show information

Details

GDS – Genomic Data Structures, the extended file name used for storing genetic data, and the file format used in the gdsfmt package.

Eigen – the text format used in EIGENSTRAT.

Value

None.

Author(s)

Xiuwen Zheng

References

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

Price AL, Patterson NJ, Plenge RM, Weinblatt ME, Shadick NA, Reich D (2006) Principal components analysis corrects for stratification in genome-wide association studies. Nat Genet. 38, 904-909.

http://corearray.sourceforge.net/

See Also

snpgdsGDS2PED

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

snpset <- snpgdsSelectSNP(genofile, missing.rate=0.95)
snpgdsGDS2Eigen(genofile, eigen.fn="tmpeigen", snp.id=snpset)

# close the genotype file
snpgdsClose(genofile)
```

```
# delete the temporary files
unlink(c("tmpeigen.eigenstratgeno", "tmpeigen.ind", "tmpeigen.snp"), force=TRUE)
```

---

snpgdsGDS2PED *Conversion from GDS to PED*

Description

Convert a GDS file to a PLINK text ped file.

Usage

```
snpgdsGDS2PED(gdsobj, ped.fn, sample.id=NULL, snp.id=NULL, use.snp.rsid=TRUE,
    format=c("A/G/C/T", "A/B", "1/2"), verbose=TRUE)
```

Arguments

gdsobj	a GDS file object (gds.class)
ped.fn	the file name of output
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
use.snp.rsid	if TRUE, use "snp.rs.id" instead of "snp.id" if available
format	specify the coding: "A/G/C/T" – allelic codes stored in "snp.allele" of the GDS file; "A/B" – A and B codes; "1/2" – 1 and 2 codes
verbose	if TRUE, show information

Details

GDS – Genomic Data Structures, the extended file name used for storing genetic data, and the file format used in the gdsfmt package.

PED – the PLINK text ped format.

Value

None.

Author(s)

Xiuwen Zheng

References

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ & Sham PC. 2007. PLINK: a toolset for whole-genome association and population-based linkage analysis. American Journal of Human Genetics, 81.

http://corearray.sourceforge.net/

See Also

snpgdsGDS2BED

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# GDS ==> PED
snpgdsGDS2PED(genofile, ped.fn="tmp")

# close the GDS file
snpgdsClose(genofile)
```

---

snpgdsGEN2GDS *Conversion from Oxford GEN format to GDS*

Description

Convert an Oxford GEN file (text format) to a GDS file.

Usage

```
snpgdsGEN2GDS(gen.fn, sample.fn, out.fn, chr.code=NULL,
    call.threshold=0.9, version=c(">=2.0", "<=1.1.5"),
    snpfirstdim=FALSE, compress.annotation="ZIP_RA.max", compress.geno="",
    verbose=TRUE)
```

Arguments

gen.fn	the file name of Oxford GEN text file(s), it could be a vector indicate merging all files
sample.fn	the file name of sample annotation
out.fn	the output GDS file
chr.code	a vector of chromosome code according to gen.fn, indicating chromosomes. It could be either numeric or character-type
call.threshold	the threshold to determine missing genotypes
version	either ">=2.0" or "<=1.1.5", see details
snpfirstdim	if TRUE, genotypes are stored in the individual-major mode, (i.e, list all SNPs for the first individual, and then list all SNPs for the second individual, etc)
compress.annotation	the compression method for the GDS variables, except "genotype"; optional values are defined in the function add.gdsn
compress.geno	the compression method for "genotype"; optional values are defined in the function add.gdsn
verbose	if TRUE, show information

Details

GDS – Genomic Data Structures, the extended file name used for storing genetic data, and the file format is used in the gdsfmt package.

NOTE : the sample file format (`sample.fn`) has changed with the release of SNPTEST v2. Specifically, the way in which covariates and phenotypes are coded on the second line of the header file has changed. `version` has to be specified, and the function uses ">=2.0" by default.

Value

Return the file name of GDS format with an absolute path.

Author(s)

Xiuwen Zheng

References

https://code.enkre.net/bgen

See Also

snpgdsBED2GDS, snpgdsVCF2GDS

Examples

```
cat("running snpgdsGEN2GDS ...\n")
## Not run:
snpgdsGEN2GDS("test.gen", "test.sample", "output.gds", chr.code=1)

## End(Not run)
```

---

snpgdsGetGeno *To get a genotype matrix*

Description

To get a genotype matrix from a specified GDS file

Usage

```
snpgdsGetGeno(gdsobj, sample.id=NULL, snp.id=NULL, snpfirstdim=NA,
    .snpread=NA, with.id=FALSE, verbose=TRUE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file; or characters to specify the file name of SNP GDS
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
snpfirstdim	if TRUE, genotypes are stored in the individual-major mode, (i.e, list all SNPs for the first individual, and then list all SNPs for the second individual, etc); FALSE for snp-major mode; if NA, determine automatically
.snpread	internal use
with.id	if TRUE, return sample.id and snp.id
verbose	if TRUE, show information

Value

The function returns an integer matrix with values 0, 1, 2 or NA representing the number of reference allele when with.id=FALSE; or list(genotype, sample.id, snp.id) when with.id=TRUE. The orders of sample and SNP IDs in the genotype matrix are actually consistent with sample.id and snp.id in the GDS file, which may not be as the same as the arguments sampel.id and snp.id specified by users.

Author(s)

Xiuwen Zheng

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

set.seed(1000)
snpset <- sample(read.gdsn(index.gdsn(genofile, "snp.id")), 1000)

mat1 <- snpgdsGetGeno(genofile, snp.id=snpset, snpfirstdim=TRUE)
dim(mat1)
# 1000  279
table(c(mat1), exclude=NULL)

mat2 <- snpgdsGetGeno(genofile, snp.id=snpset, snpfirstdim=FALSE)
dim(mat2)
# 279 1000
table(c(mat2), exclude=NULL)

identical(t(mat1), mat2)
# TRUE

# close the file
snpgdsClose(genofile)
```

`snpgdsGRM` *Genetic Relationship Matrix (GRM) for SNP genotype data*

Description

Calculate Genetic Relationship Matrix (GRM) using SNP genotype data.

Usage

```
snpgdsGRM(gdsobj, sample.id=NULL, snp.id=NULL,
    autosome.only=TRUE, remove.monosnp=TRUE, maf=NaN, missing.rate=NaN,
    method=c("GCTA", "Eigenstrat", "EIGMIX", "Weighted", "Corr", "IndivBeta"),
    num.thread=1L, useMatrix=FALSE, out.fn=NULL, out.prec=c("double", "single"),
    out.compress="LZMA_RA", with.id=TRUE, verbose=TRUE)
```

Arguments

`gdsobj`	an object of class `SNPGDSFileClass`, a SNP GDS file
`sample.id`	a vector of sample id specifying selected samples; if NULL, all samples are used
`snp.id`	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
`autosome.only`	if `TRUE`, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
`remove.monosnp`	if TRUE, remove monomorphic SNPs
`maf`	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
`missing.rate`	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
`method`	"GCTA" – genetic relationship matrix defined in CGTA; "Eigenstrat" – genetic covariance matrix in EIGENSTRAT; "EIGMIX" – two times coancestry matrix defined in Zheng&Weir (2016), "Weighted" – weighted GCTA, as the same as "EIGMIX", "Corr" – Scaled GCTA GRM (dividing each i,j element by the product of the square root of the i,i and j,j elements), "IndivBeta" – two times individual beta estimate relative to the minimum of beta; see details
`num.thread`	the number of (CPU) cores used; if `NA`, detect the number of cores automatically
`useMatrix`	if `TRUE`, use `Matrix::dspMatrix` to store the output square matrix to save memory
`out.fn`	NULL for no GDS output, or a file name
`out.prec`	double or single precision for storage
`out.compress`	the compression method for storing the GRM matrix in the GDS file
`with.id`	if `TRUE`, the returned value with `sample.id` and `sample.id`
`verbose`	if `TRUE`, show information

Details

"GCTA": the genetic relationship matrix in GCTA is defined as $G_ij = avg_l [(g_il - 2*p_l)*(g_jl - 2*p_l) / 2*p_l*(1 - p_l)]$ for individuals i,j and locus l;

"Eigenstrat": the genetic covariance matrix in EIGENSTRAT $G_ij = avg_l [(g_il - 2*p_l)*(g_jl - 2*p_l) / 2*p_l*(1 - p_l)]$ for individuals i,j and locus l; the missing genotype is imputed by the dosage mean of that locus.

"EIGMIX" / "Weighted": it is the same as '2 * snpgdsEIGMIX(, ibdmat=TRUE, diagadj=FALSE)$ibd': $G_ij = [sum_l (g_il - 2*p_l)*(g_jl - 2*p_l)] / [sum_l 2*p_l*(1 - p_l)]$ for individuals i,j and locus l;

"IndivBeta": 'beta = snpgdsIndivBeta(, inbreeding=TRUE)' (Weir&Goudet, 2017), and beta-based GRM is $grm_ij = 2 * (beta_ij - beta_min) / (1 - beta_min)$ for $i!=j$, $grm_ij = 1 + (beta_i - beta_min) / (1 - beta_min)$ for $i=j$. It is relative to the minimum value of beta estimates.

Value

Return a list if `with.id` = `TRUE`:

`sample.id`	the sample ids used in the analysis
`snp.id`	the SNP ids used in the analysis
`method`	characters, the method used
`grm`	the genetic relationship matrix; different methods might have different meanings and interpretation for estimates

If `with.id` = `FALSE`, this function returns the genetic relationship matrix (GRM) without sample and SNP IDs.

Author(s)

Xiuwen Zheng

References

Patterson, N., Price, A. L. & Reich, D. Population structure and eigenanalysis. PLoS Genet. 2, e190 (2006).

Yang, J., Lee, S. H., Goddard, M. E. & Visscher, P. M. GCTA: a tool for genome-wide complex trait analysis. American journal of human genetics 88, 76-82 (2011).

Zheng X, Weir BS. Eigenanalysis on SNP Data with an Interpretation of Identity by Descent. Theoretical Population Biology. 2016 Feb;107:65-76. doi: 10.1016/j.tpb.2015.09.004

Weir BS, Zheng X. SNPs and SNVs in Forensic Science. Forensic Science International: Genetics Supplement Series. 2015. doi:10.1016/j.fsigss.2015.09.106

Weir BS, Goudet J. A Unified Characterization of Population Structure and Relatedness. Genetics. 2017 Aug;206(4):2085-2103. doi: 10.1534/genetics.116.198424.

See Also

`snpgdsPCA`, `snpgdsEIGMIX`, `snpgdsIndivBeta`, `snpgdsIndInb`, `snpgdsFst`, `snpgdsMergeGRM`

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

rv <- snpgdsGRM(genofile, method="GCTA")
eig <- eigen(rv$grm)  # Eigen-decomposition

# output to a GDS file
snpgdsGRM(genofile, method="GCTA", out.fn="test.gds")


pop <- factor(read.gdsn(index.gdsn(genofile, "sample.annot/pop.group")))
plot(eig$vectors[,1], eig$vectors[,2], col=pop)
legend("topleft", legend=levels(pop), pch=19, col=1:4)


# close the file
snpgdsClose(genofile)

# delete the temporary file
unlink("test.gds", force=TRUE)
```

---

snpgdsHCluster *Hierarchical cluster analysis*

Description

Perform hierarchical cluster analysis on the dissimilarity matrix.

Usage

```
snpgdsHCluster(dist, sample.id=NULL, need.mat=TRUE, hang=0.25)
```

Arguments

dist	an object of "snpgdsDissClass" from snpgdsDiss, an object of "snpgdsIBSClass" from snpgdsIBS, or a square matrix for dissimilarity
sample.id	to specify sample id, only work if dist is a matrix
need.mat	if TRUE, store the dissimilarity matrix in the result
hang	The fraction of the plot height by which labels should hang below the rest of the plot. A negative value will cause the labels to hang down from 0.

Details

Call the function hclust to perform hierarchical cluster analysis, using method="average".

Value

Return a list (class "snpgdsHCClass"):

`sample.id`	the sample ids used in the analysis
`hclust`	an object returned from `hclust`
`dendrogram`	
`dist`	the dissimilarity matrix, if `need.mat = TRUE`

Author(s)

Xiuwen Zheng

See Also

`snpgdsIBS`, `snpgdsDiss`, `snpgdsCutTree`

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

pop.group <- read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))
pop.group <- as.factor(pop.group)
pop.level <- levels(pop.group)

diss <- snpgdsDiss(genofile)
hc <- snpgdsHCluster(diss)
rv <- snpgdsCutTree(hc)
rv

# call 'plot' to draw a dendrogram
plot(rv$dendrogram, leaflab="none", main="HapMap Phase II")


# the distribution of Z scores
snpgdsDrawTree(rv, type="z-score", main="HapMap Phase II")

# draw dendrogram
snpgdsDrawTree(rv, main="HapMap Phase II",
    edgePar=list(col=rgb(0.5,0.5,0.5, 0.75), t.col="black"))


# close the file
snpgdsClose(genofile)
```

snpgdsHWE *Statistical test of Hardy-Weinberg Equilibrium*

Description

Calculate the p-values for the exact SNP test of Hardy-Weinberg Equilibrium.

Usage

```
snpgdsHWE(gdsobj, sample.id=NULL, snp.id=NULL, with.id=FALSE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples will be used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs will be used
with.id	if TRUE, the returned value with sample and SNP IDs

Value

If with.id=FALSE, return a vector of numeric values (p-value); otherwise, return a list with three components "pvalue", "sample.id" and "snp.id".

Author(s)

Xiuwen Zheng, Janis E. Wigginton

References

Wigginton, J. E., Cutler, D. J. & Abecasis, G. R. A note on exact tests of Hardy-Weinberg equilibrium. Am. J. Hum. Genet. 76, 887-93 (2005).

See Also

snpgdsSNPRateFreq

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# Japanese samples
sample.id <- read.gdsn(index.gdsn(genofile, "sample.id"))
pop <- read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))
(samp.sel <- sample.id[pop=="JPT"])
samp.sel <- samp.sel[nchar(samp.sel) == 7]
```

```
# chromosome 1
snp.id <- snpgdsSelectSNP(genofile, sample.id=samp.sel, autosome.only=1L)

# HWE test
p <- snpgdsHWE(genofile, sample.id=samp.sel, snp.id=snp.id)
summary(p)

# QQ plot
plot(-log10((1:length(p))/length(p)), -log10(p[order(p)]),
xlab="-log10(expected P)", ylab="-log10(observed P)", main="QQ plot")
abline(a=0, b=1, col="blue")

# close the genotype file
snpgdsClose(genofile)
```

`snpgdsIBDKING` *KING method of moment for the identity-by-descent (IBD) analysis*

Description

Calculate IBD coefficients by KING method of moment.

Usage

```
snpgdsIBDKING(gdsobj, sample.id=NULL, snp.id=NULL, autosome.only=TRUE,
    remove.monosnp=TRUE, maf=NaN, missing.rate=NaN,
    type=c("KING-robust", "KING-homo"), family.id=NULL, num.thread=1L,
    useMatrix=FALSE, verbose=TRUE)
```

Arguments

`gdsobj`	an object of class `SNPGDSFileClass`, a SNP GDS file
`sample.id`	a vector of sample id specifying selected samples; if NULL, all samples are used
`snp.id`	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
`autosome.only`	if `TRUE`, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
`remove.monosnp`	if TRUE, remove monomorphic SNPs
`maf`	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
`missing.rate`	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
`type`	`"KING-robust"` – relationship inference in the presence of population stratification (by default); `"KING-homo"` – relationship inference in a homogeneous population
`family.id`	if `NULL`, all individuals are treated as singletons; if family id is given, within- and between-family relationship are estimated differently. If `sample.id=NULL`, `family.id` should have the same length as "sample.id" in the GDS file, otherwise `family.id` should have the same length and order as the argument `sample.id`

num.thread	the number of (CPU) cores used; if NA, detect the number of cores automatically
useMatrix	if TRUE, use Matrix::dspMatrix to store the output square matrix to save memory
verbose	if TRUE, show information

Details

KING IBD estimator is a moment estimator, and it is computationally efficient relative to MLE method. The approaches include "KING-robust" – robust relationship inference within or across families in the presence of population substructure, and "KING-homo" – relationship inference in a homogeneous population.

With "KING-robust", the function would return the proportion of SNPs with zero IBS (IBS0) and kinship coefficient (kinship). With "KING-homo" it would return the probability of sharing one IBD (k1) and the probability of sharing zero IBD (k0).

The minor allele frequency and missing rate for each SNP passed in snp.id are calculated over all the samples in sample.id.

Value

Return a list:

sample.id	the sample ids used in the analysis
snp.id	the SNP ids used in the analysis
k0	a matrix for IBD coefficients, the probability of sharing zero IBD, if type="KING-homo"
k1	a matrix for IBD coefficients, the probability of sharing one IBD, if type="KING-homo"
IBS0	a matrix for the proportions of SNPs with zero IBS, if type="KING-robust"
kinship	a matrix for the estimated kinship coefficients, if type="KING-robust"

Author(s)

Xiuwen Zheng

References

Manichaikul A, Mychaleckyj JC, Rich SS, Daly K, Sale M, Chen WM. Robust relationship inference in genome-wide association studies. Bioinformatics. 2010 Nov 15;26(22):2867-73.

See Also

snpgdsIBDMLE, snpgdsIBDMoM

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# CEU population
samp.id <- read.gdsn(index.gdsn(genofile, "sample.id"))
```

```
CEU.id <- samp.id[
    read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))=="CEU"]



####  KING-robust:
####  relationship inference in the presence of population stratification
####               robust relationship inference across family

ibd.robust <- snpgdsIBDKING(genofile, sample.id=CEU.id)
names(ibd.robust)
# [1] "sample.id" "snp.id"    "afreq"     "IBS0"      "kinship"

# select a set of pairs of individuals
dat <- snpgdsIBDSelection(ibd.robust, 1/32)
head(dat)

plot(dat$IBS0, dat$kinship, xlab="Proportion of Zero IBS",
    ylab="Estimated Kinship Coefficient (KING-robust)")


# using Matrix
ibd.robust <- snpgdsIBDKING(genofile, sample.id=CEU.id, useMatrix=TRUE)
is(ibd.robust$IBS0)  # dspMatrix
is(ibd.robust$kinship)  # dspMatrix



####  KING-robust:
####  relationship inference in the presence of population stratification
####               within- and between-family relationship inference

# incorporate with pedigree information
family.id <- read.gdsn(index.gdsn(genofile, "sample.annot/family.id"))
family.id <- family.id[match(CEU.id, samp.id)]

ibd.robust2 <- snpgdsIBDKING(genofile, sample.id=CEU.id, family.id=family.id)
names(ibd.robust2)

# select a set of pairs of individuals
dat <- snpgdsIBDSelection(ibd.robust2, 1/32)
head(dat)

plot(dat$IBS0, dat$kinship, xlab="Proportion of Zero IBS",
    ylab="Estimated Kinship Coefficient (KING-robust)")



####  KING-homo: relationship inference in a homogeneous population

ibd.homo <- snpgdsIBDKING(genofile, sample.id=CEU.id, type="KING-homo")
names(ibd.homo)
# "sample.id" "snp.id"    "afreq"     "k0"        "k1"
```

```
# select a subset of pairs of individuals
dat <- snpgdsIBDSelection(ibd.homo, 1/32)
head(dat)

plot(dat$k0, dat$kinship, xlab="Pr(IBD=0)",
    ylab="Estimated Kinship Coefficient (KING-homo)")


# using Matrix
ibd.homo <- snpgdsIBDKING(genofile, sample.id=CEU.id, type="KING-homo",
    useMatrix=TRUE)
is(ibd.homo$k0)  # dspMatrix
is(ibd.homo$k1)  # dspMatrix


# close the genotype file
snpgdsClose(genofile)
```

`snpgdsIBDMLE` *Maximum likelihood estimation (MLE) for the Identity-By-Descent (IBD) Analysis*

Description

Calculate the three IBD coefficients (k0, k1, k2) for non-inbred individual pairs by Maximum Likelihood Estimation.

Usage

```
snpgdsIBDMLE(gdsobj, sample.id=NULL, snp.id=NULL, autosome.only=TRUE,
    remove.monosnp=TRUE, maf=NaN, missing.rate=NaN, kinship=FALSE,
    kinship.constraint=FALSE, allele.freq=NULL,
    method=c("EM", "downhill.simplex", "Jacquard"), max.niter=1000L,
    reltol=sqrt(.Machine$double.eps), coeff.correct=TRUE,
    out.num.iter=TRUE, num.thread=1, verbose=TRUE)
```

Arguments

`gdsobj`	an object of class `SNPGDSFileClass`, a SNP GDS file
`sample.id`	a vector of sample id specifying selected samples; if NULL, all samples are used
`snp.id`	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
`autosome.only`	if `TRUE`, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
`remove.monosnp`	if TRUE, remove monomorphic SNPs
`maf`	to use the SNPs with ">= maf" only; if NaN, no any MAF threshold
`missing.rate`	to use the SNPs with "<= missing.rate" only; if NaN, no any missing threshold

kinship	if TRUE, output the estimated kinship coefficients
kinship.constraint	if TRUE, constrict IBD coefficients (k_0,k_1,k_2) in the geneloical region ($2 k_0 k_1 >= k_2^2$)
allele.freq	to specify the allele frequencies; if NULL, determine the allele frequencies from gdsobj using the specified samples; if snp.id is specified, allele.freq should have the same order as snp.id
method	"EM", "downhill.simplex", "Jacquard", see details
max.niter	the maximum number of iterations
reltol	relative convergence tolerance; the algorithm stops if it is unable to reduce the value of log likelihood by a factor of $reltol * (abs(log likelihood with the initial parameters) + reltol)$ at a step.
coeff.correct	TRUE by default, see details
out.num.iter	if TRUE, output the numbers of iterations
num.thread	the number of (CPU) cores used; if NA, detect the number of cores automatically
verbose	if TRUE, show information

Details

The minor allele frequency and missing rate for each SNP passed in snp.id are calculated over all the samples in sample.id.

The PLINK moment estimates are used as the initial values in the algorithm of searching maximum value of log likelihood function. Two numeric approaches can be used: one is Expectation-Maximization (EM) algorithm, and the other is Nelder-Mead method or downhill simplex method. Generally, EM algorithm is more robust than downhill simplex method. "Jacquard" refers to the estimation of nine Jacquard's coefficients.

If coeff.correct is TRUE, the final point that is found by searching algorithm (EM or downhill simplex) is used to compare the six points (fullsib, offspring, halfsib, cousin, unrelated), since any numeric approach might not reach the maximum position after a finit number of steps. If any of these six points has a higher value of log likelihood, the final point will be replaced by the best one.

Although MLE estimates are more reliable than MoM, MLE is much more computationally intensive than MoM, and might not be feasible to estimate pairwise relatedness for a large dataset.

Value

Return a snpgdsIBDClass object, which is a list:

sample.id	the sample ids used in the analysis
snp.id	the SNP ids used in the analysis
afreq	the allele frequencies used in the analysis
k0	IBD coefficient, the probability of sharing ZERO IBD, if method="EM" or "downhill.simplex"
k1	IBD coefficient, the probability of sharing ONE IBD, if method="EM" or "downhill.simplex"
D1, ..., D8	Jacquard's coefficients, if method="Jacquard", D9 = 1 - D1 - ... - D8
kinship	the estimated kinship coefficients, if the parameter kinship=TRUE

Author(s)

Xiuwen Zheng

References

Milligan BG. 2003. Maximum-likelihood estimation of relatedness. Genetics 163:1153-1167.

Weir BS, Anderson AD, Hepler AB. 2006. Genetic relatedness analysis: modern data and new challenges. Nat Rev Genet. 7(10):771-80.

Choi Y, Wijsman EM, Weir BS. 2009. Case-control association testing in the presence of unknown relationships. Genet Epidemiol 33(8):668-78.

Jacquard, A. Structures Genetiques des Populations (Masson & Cie, Paris, 1970); English translation available in Charlesworth, D. & Chalesworth, B. Genetics of Human Populations (Springer, New York, 1974).

See Also

snpgdsIBDMLELogLik, snpgdsIBDMoM

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

YRI.id <- read.gdsn(index.gdsn(genofile, "sample.id"))[
    read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))=="YRI"]
YRI.id <- YRI.id[1:30]

# SNP pruning
set.seed(10)
snpset <- snpgdsLDpruning(genofile, sample.id=YRI.id, maf=0.05,
    missing.rate=0.05)
snpset <- sample(unlist(snpset), 250)
mibd <- snpgdsIBDMLE(genofile, sample.id=YRI.id, snp.id=snpset)
mibd

# select a set of pairs of individuals
d <- snpgdsIBDSelection(mibd, kinship.cutoff=1/8)
head(d)


# log likelihood

loglik <- snpgdsIBDMLELogLik(genofile, mibd)
loglik0 <- snpgdsIBDMLELogLik(genofile, mibd, relatedness="unrelated")

# likelihood ratio test
p.value <- pchisq(loglik - loglik0, 1, lower.tail=FALSE)


flag <- lower.tri(mibd$k0)
```

```
plot(NaN, xlim=c(0,1), ylim=c(0,1), xlab="k0", ylab="k1")
lines(c(0,1), c(1,0), col="red", lty=3)
points(mibd$k0[flag], mibd$k1[flag])

# specify the allele frequencies
afreq <- snpgdsSNPRateFreq(genofile, sample.id=YRI.id,
    snp.id=snpset)$AlleleFreq
subibd <- snpgdsIBDMLE(genofile, sample.id=YRI.id[1:25], snp.id=snpset,
    allele.freq=afreq)
summary(c(subibd$k0 - mibd$k0[1:25, 1:25]))
# ZERO
summary(c(subibd$k1 - mibd$k1[1:25, 1:25]))
# ZERO


# close the genotype file
snpgdsClose(genofile)
```

`snpgdsIBDMLELogLik` *Log likelihood for MLE method in the Identity-By-Descent (IBD) Analysis*

Description

Calculate the log likelihood values from maximum likelihood estimation.

Usage

```
snpgdsIBDMLELogLik(gdsobj, ibdobj, k0 = NaN, k1 = NaN,
    relatedness=c("", "self", "fullsib", "offspring",
    "halfsib", "cousin", "unrelated"))
```

Arguments

`gdsobj`	an object of class `SNPGDSFileClass`, a SNP GDS file
`ibdobj`	the `snpgdsIBDClass` object returned from snpgdsIBDMLE
`k0`	specified IBD coefficient
`k1`	specified IBD coefficient
`relatedness`	specify a relatedness, otherwise use the values of k0 and k1

Details

If (`relatedness` == "") and (k0 == NaN or k1 == NaN), then return the log likelihood values for each (k0, k1) stored in ibdobj. \ If (`relatedness` == "") and (k0 != NaN) and (k1 != NaN), then return the log likelihood values for a specific IBD coefficient (k0, k1). \ If `relatedness` is: "self", then k0 = 0, k1 = 0; "fullsib", then k0 = 0.25, k1 = 0.5; "offspring", then k0 = 0, k1 = 1; "halfsib", then k0 = 0.5, k1 = 0.5; "cousin", then k0 = 0.75, k1 = 0.25; "unrelated", then k0 = 1, k1 = 0.

Value

Return a n-by-n matrix of log likelihood values, where n is the number of samples.

Author(s)

Xiuwen Zheng

References

Milligan BG. 2003. Maximum-likelihood estimation of relatedness. Genetics 163:1153-1167.

Weir BS, Anderson AD, Hepler AB. 2006. Genetic relatedness analysis: modern data and new challenges. Nat Rev Genet. 7(10):771-80.

Choi Y, Wijsman EM, Weir BS. 2009. Case-control association testing in the presence of unknown relationships. Genet Epidemiol 33(8):668-78.

See Also

snpgdsIBDMLE, snpgdsIBDMoM

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

YRI.id <- read.gdsn(index.gdsn(genofile, "sample.id"))[
    read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))=="YRI"]
YRI.id <- YRI.id[1:30]

# SNP pruning
set.seed(10)
snpset <- snpgdsLDpruning(genofile, sample.id=YRI.id, maf=0.05,
    missing.rate=0.05)
snpset <- sample(unlist(snpset), 250)
mibd <- snpgdsIBDMLE(genofile, sample.id=YRI.id, snp.id=snpset)
names(mibd)

# select a set of pairs of individuals
d <- snpgdsIBDSelection(mibd, kinship.cutoff=1/8)
head(d)


# log likelihood

loglik <- snpgdsIBDMLELogLik(genofile, mibd)
loglik0 <- snpgdsIBDMLELogLik(genofile, mibd, relatedness="unrelated")

# likelihood ratio test
p.value <- pchisq(loglik - loglik0, 1, lower.tail=FALSE)


flag <- lower.tri(mibd$k0)
```

```
plot(NaN, xlim=c(0,1), ylim=c(0,1), xlab="k0", ylab="k1")
lines(c(0,1), c(1,0), col="red", lty=3)
points(mibd$k0[flag], mibd$k1[flag])

# specify the allele frequencies
afreq <- snpgdsSNPRateFreq(genofile, sample.id=YRI.id,
    snp.id=snpset)$AlleleFreq
subibd <- snpgdsIBDMLE(genofile, sample.id=YRI.id[1:25], snp.id=snpset,
    allele.freq=afreq)
summary(c(subibd$k0 - mibd$k0[1:25, 1:25]))
# ZERO
summary(c(subibd$k1 - mibd$k1[1:25, 1:25]))
# ZERO


# close the genotype file
snpgdsClose(genofile)
```

`snpgdsIBDMoM` *PLINK method of moment (MoM) for the Identity-By-Descent (IBD) Analysis*

Description

Calculate three IBD coefficients for non-inbred individual pairs by PLINK method of moment (MoM).

Usage

```
snpgdsIBDMoM(gdsobj, sample.id=NULL, snp.id=NULL, autosome.only=TRUE,
    remove.monosnp=TRUE, maf=NaN, missing.rate=NaN, allele.freq=NULL,
    kinship=FALSE, kinship.constraint=FALSE, num.thread=1L, useMatrix=FALSE,
    verbose=TRUE)
```

Arguments

`gdsobj`	an object of class `SNPGDSFileClass`, a SNP GDS file
`sample.id`	a vector of sample id specifying selected samples; if `NULL`, all samples are used
`snp.id`	a vector of snp id specifying selected SNPs; if `NULL`, all SNPs are used
`autosome.only`	if `TRUE`, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
`remove.monosnp`	if `TRUE`, remove monomorphic SNPs
`maf`	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
`missing.rate`	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
`allele.freq`	to specify the allele frequencies; if NULL, determine the allele frequencies from `gdsobj` using the specified samples; if `snp.id` is specified, `allele.freq` should have the same order as `snp.id`

kinship	if TRUE, output the estimated kinship coefficients
kinship.constraint	if TRUE, constrict IBD coefficients (k_0,k_1,k_2) in the geneloical region ($2 k_0 k_1 >= k_2^2$)
num.thread	the number of (CPU) cores used; if NA, detect the number of cores automatically
useMatrix	if TRUE, use Matrix::dspMatrix to store the output square matrix to save memory
verbose	if TRUE, show information

Details

PLINK IBD estimator is a moment estimator, and it is computationally efficient relative to MLE method. In the PLINK method of moment, a correction factor based on allele counts is used to adjust for sampling. However, if allele frequencies are specified, no correction factor is conducted since the specified allele frequencies are assumed to be known without sampling.

The minor allele frequency and missing rate for each SNP passed in snp.id are calculated over all the samples in sample.id.

Value

Return a list:

sample.id	the sample ids used in the analysis
snp.id	the SNP ids used in the analysis
k0	IBD coefficient, the probability of sharing ZERO IBD
k1	IBD coefficient, the probability of sharing ONE IBD
kinship	the estimated kinship coefficients, if the parameter kinship=TRUE

Author(s)

Xiuwen Zheng

References

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ & Sham PC. 2007. PLINK: a toolset for whole-genome association and population-based linkage analysis. American Journal of Human Genetics, 81.

See Also

snpgdsIBDMLE, snpgdsIBDMLELogLik

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

############################################################
# CEU population

CEU.id <- read.gdsn(index.gdsn(genofile, "sample.id"))[
    read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))=="CEU"]
pibd <- snpgdsIBDMoM(genofile, sample.id=CEU.id)
names(pibd)

flag <- lower.tri(pibd$k0)
plot(NaN, xlim=c(0,1), ylim=c(0,1), xlab="k0", ylab="k1")
lines(c(0,1), c(1,0), col="red", lty=3)
points(pibd$k0[flag], pibd$k1[flag])

# select a set of pairs of individuals
d <- snpgdsIBDSelection(pibd, kinship.cutoff=1/8)
head(d)


############################################################
# YRI population

YRI.id <- read.gdsn(index.gdsn(genofile, "sample.id"))[
    read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))=="YRI"]
pibd <- snpgdsIBDMoM(genofile, sample.id=YRI.id)
flag <- lower.tri(pibd$k0)
plot(NaN, xlim=c(0,1), ylim=c(0,1), xlab="k0", ylab="k1")
lines(c(0,1), c(1,0), col="red", lty=3)
points(pibd$k0[flag], pibd$k1[flag])


# specify the allele frequencies
afreq <- snpgdsSNPRateFreq(genofile, sample.id=YRI.id)$AlleleFreq
aibd <- snpgdsIBDMoM(genofile, sample.id=YRI.id, allele.freq=afreq)
flag <- lower.tri(aibd$k0)
plot(NaN, xlim=c(0,1), ylim=c(0,1), xlab="k0", ylab="k1")
lines(c(0,1), c(1,0), col="red", lty=3)
points(aibd$k0[flag], aibd$k1[flag])

# analysis on a subset
subibd <- snpgdsIBDMoM(genofile, sample.id=YRI.id[1:25], allele.freq=afreq)
summary(c(subibd$k0 - aibd$k0[1:25, 1:25]))
# ZERO
summary(c(subibd$k1 - aibd$k1[1:25, 1:25]))
# ZERO


# close the genotype file
snpgdsClose(genofile)
```

snpgdsIBDSelection *Get a table of IBD coefficients*

Description

Return a data frame with IBD coefficients.

Usage

```
snpgdsIBDSelection(ibdobj, kinship.cutoff=NaN, samp.sel=NULL)
```

Arguments

ibdobj	an object of snpgdsIBDClass returned by snpgdsIBDMLE or snpgdsIBDMoM
kinship.cutoff	select the individual pairs with kinship coefficients >= kinship.cutoff; no filter if kinship.cutoff = NaN
samp.sel	a logical vector or integer vector to specify selection of samples

Value

Return a data.frame:

ID1	the id of the first individual
ID2	the id of the second individual
k0	the probability of sharing ZERO alleles
k1	the probability of sharing ONE alleles
kinship	kinship coefficient

Author(s)

Xiuwen Zheng

See Also

snpgdsIBDMLE, snpgdsIBDMoM, snpgdsIBDKING

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# YRI population
YRI.id <- read.gdsn(index.gdsn(genofile, "sample.id"))[
    read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))=="YRI"]
pibd <- snpgdsIBDMoM(genofile, sample.id=YRI.id)
flag <- lower.tri(pibd$k0)
plot(NaN, xlim=c(0,1), ylim=c(0,1), xlab="k0", ylab="k1")
```

```
lines(c(0,1), c(1,0), col="red", lty=3)
points(pibd$k0[flag], pibd$k1[flag])

# close the genotype file
snpgdsClose(genofile)

# IBD coefficients
dat <- snpgdsIBDSelection(pibd, 1/32)
head(dat)
#       ID1     ID2          k0        k1    kinship
# 1 NA19152 NA19154 0.010749154 0.9892508 0.24731271
# 2 NA19152 NA19093 0.848207777 0.1517922 0.03794806
# 3 NA19139 NA19138 0.010788047 0.9770181 0.25035144
# 4 NA19139 NA19137 0.012900661 0.9870993 0.24677483
# 5 NA18912 NA18914 0.008633077 0.9913669 0.24784173
# 6 NA19160 NA19161 0.008635754 0.9847777 0.24948770
```

`snpgdsIBS` *Identity-By-State (IBS) proportion*

Description

Calculate the fraction of identity by state for each pair of samples

Usage

```
snpgdsIBS(gdsobj, sample.id=NULL, snp.id=NULL, autosome.only=TRUE,
    remove.monosnp=TRUE, maf=NaN, missing.rate=NaN, num.thread=1L,
    useMatrix=FALSE, verbose=TRUE)
```

Arguments

`gdsobj`	an object of class `SNPGDSFileClass`, a SNP GDS file
`sample.id`	a vector of sample id specifying selected samples; if NULL, all samples are used
`snp.id`	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
`autosome.only`	if `TRUE`, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
`remove.monosnp`	if TRUE, remove monomorphic SNPs
`maf`	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
`missing.rate`	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
`num.thread`	the number of (CPU) cores used; if `NA`, detect the number of cores automatically
`useMatrix`	if `TRUE`, use `Matrix::dspMatrix` to store the output square matrix to save memory
`verbose`	if TRUE, show information

Details

The minor allele frequency and missing rate for each SNP passed in `snp.id` are calculated over all the samples in `sample.id`.

The values of the IBS matrix range from ZERO to ONE, and it is defined as the average of `1 - | g_{1,i} - g_{2,i} | / 2` across the genome for the first and second individuals and SNP i.

Value

Return a list (class "snpgdsIBSClass"):

`sample.id`	the sample ids used in the analysis
`snp.id`	the SNP ids used in the analysis
`ibs`	a matrix of IBS proportion, "# of samples" x "# of samples"

Author(s)

Xiuwen Zheng

See Also

`snpgdsIBSNum`

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# perform identity-by-state calculations
ibs <- snpgdsIBS(genofile)

# perform multidimensional scaling analysis on
# the genome-wide IBS pairwise distances:
loc <- cmdscale(1 - ibs$ibs, k = 2)
x <- loc[, 1]; y <- loc[, 2]
race <- as.factor(read.gdsn(index.gdsn(genofile, "sample.annot/pop.group")))
plot(x, y, col=race, xlab = "", ylab = "", main = "cmdscale(IBS Distance)")
legend("topleft", legend=levels(race), text.col=1:nlevels(race))

# close the file
snpgdsClose(genofile)
```

snpgdsIBSNum *Identity-By-State (IBS)*

Description

Calculate the number of SNPs for identity by state for each pair of samples.

Usage

```
snpgdsIBSNum(gdsobj, sample.id = NULL, snp.id = NULL, autosome.only = TRUE,
    remove.monosnp = TRUE, maf = NaN, missing.rate = NaN, num.thread = 1L,
    verbose = TRUE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
autosome.only	if TRUE, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
remove.monosnp	if TRUE, remove monomorphic SNPs
maf	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
missing.rate	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
num.thread	the number of (CPU) cores used; if NA, detect the number of cores automatically
verbose	if TRUE, show information

Details

The minor allele frequency and missing rate for each SNP passed in snp.id are calculated over all the samples in sample.id.

Value

Return a list (n is the number of samples):

sample.id	the sample ids used in the analysis
snp.id	the SNP ids used in the analysis
ibs0	a n-by-n matrix, the number of SNPs sharing 0 IBS
ibs1	a n-by-n matrix, the number of SNPs sharing 1 IBS
ibs2	a n-by-n matrix, the number of SNPs sharing 2 IBS

Author(s)

Xiuwen Zheng

See Also

snpgdsIBS

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

RV <- snpgdsIBSNum(genofile)
pop <- read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))
L <- order(pop)
image(RV$ibs0[L, L]/length(RV$snp.id))

# close the genotype file
snpgdsClose(genofile)
```

---

snpgdsIndInb *Individual Inbreeding Coefficients*

Description

To calculate individual inbreeding coefficients using SNP genotype data

Usage

```
snpgdsIndInb(gdsobj, sample.id=NULL, snp.id=NULL,
    autosome.only=TRUE, remove.monosnp=TRUE, maf=NaN, missing.rate=NaN,
    method=c("mom.weir", "mom.visscher", "mle", "gcta1", "gcta2", "gcta3"),
    allele.freq=NULL, out.num.iter=TRUE, reltol=.Machine$double.eps^0.75,
    verbose=TRUE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
autosome.only	if TRUE, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
remove.monosnp	if TRUE, remove monomorphic SNPs
maf	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
missing.rate	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
method	see details
allele.freq	to specify the allele frequencies; if NULL, the allele frequencies are estimated from the given samples
out.num.iter	output the numbers of iterations

reltol	relative convergence tolerance used in MLE; the algorithm stops if it is unable to reduce the value of log likelihood by a factor of $reltol * (abs(log likelihood with the initial parameters) + reltol)$ at a step.
verbose	if TRUE, show information

Details

The method can be: "mom.weir": a modified Visscher's estimator, proposed by Bruce Weir; "mom.visscher": Visscher's estimator described in Yang et al. (2010); "mle": the maximum likelihood estimation; "gcta1": F^I in GCTA, avg [(g_i - 2p_i)^2 / (2*p_i*(1-p_i)) - 1]; "gcta2": F^II in GCTA, avg [1 - g_i*(2 - g_i) / (2*p_i*(1-p_i))]; "gcta3": F^III in GCTA, the same as "mom.visscher", avg [g_i^2 - (1 + 2p_i)*g_i + 2*p_i^2] / (2*p_i*(1-p_i)).

Value

Return estimated inbreeding coefficient.

Author(s)

Xiuwen Zheng

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.

Yang, J., Lee, S. H., Goddard, M. E. & Visscher, P. M. GCTA: a tool for genome-wide complex trait analysis. American journal of human genetics 88, 76-82 (2011).

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

rv <- snpgdsIndInb(genofile, method="mom.visscher")
head(rv$inbreeding)
summary(rv$inbreeding)

# close the genotype file
snpgdsClose(genofile)
```

---

snpgdsIndInbCoef *Individual Inbreeding Coefficient*

Description

To calculate an individual inbreeding coefficient using SNP genotype data

Usage

```
snpgdsIndInbCoef(x, p, method = c("mom.weir", "mom.visscher", "mle"),
    reltol=.Machine$double.eps^0.75)
```

Arguments

x	SNP genotypes
p	allele frequencies
method	see details
reltol	relative convergence tolerance used in MLE; the algorithm stops if it is unable to reduce the value of log likelihood by a factor of $reltol * (abs(log likelihood with the initial parameters) + reltol)$ at a step.

Details

The method can be: "mom.weir": a modified Visscher's estimator, proposed by Bruce Weir; "mom.visscher": Visscher's estimator described in Yang et al. (2010); "mle": the maximum likelihood estimation.

Value

Return estimated inbreeding coefficient.

Author(s)

Xiuwen Zheng

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.

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

chr1 <- read.gdsn(index.gdsn(genofile, "snp.id"))[
    read.gdsn(index.gdsn(genofile, "snp.chromosome"))==1]
chr1idx <- match(chr1, read.gdsn(index.gdsn(genofile, "snp.id")))

AF <- snpgdsSNPRateFreq(genofile)
g <- read.gdsn(index.gdsn(genofile, "genotype"), start=c(1,1), count=c(-1,1))

snpgdsIndInbCoef(g[chr1idx], AF$AlleleFreq[chr1idx], method="mom.weir")
snpgdsIndInbCoef(g[chr1idx], AF$AlleleFreq[chr1idx], method="mom.visscher")
snpgdsIndInbCoef(g[chr1idx], AF$AlleleFreq[chr1idx], method="mle")

# close the genotype file
```

```
snpgdsClose(genofile)
```

`snpgdsIndivBeta` *Individual inbreeding and relatedness estimation (beta estimator)*

Description

Calculate individual inbreeding and relatedness estimation (beta estimator) using SNP genotype data.

Usage

```
snpgdsIndivBeta(gdsobj, sample.id=NULL, snp.id=NULL, autosome.only=TRUE,
    remove.monosnp=TRUE, maf=NaN, missing.rate=NaN, method=c("weighted"),
    inbreeding=TRUE, num.thread=1L, with.id=TRUE, useMatrix=FALSE, verbose=TRUE)
snpgdsIndivBetaRel(beta, beta_rel, verbose=TRUE)
```

Arguments

`gdsobj`	an object of class `SNPGDSFileClass`, a SNP GDS file
`sample.id`	a vector of sample id specifying selected samples; if NULL, all samples are used
`snp.id`	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
`autosome.only`	if `TRUE`, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
`remove.monosnp`	if TRUE, remove monomorphic SNPs
`maf`	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
`missing.rate`	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
`method`	"weighted" estimator
`inbreeding`	`TRUE`, the diagonal is a vector of inbreeding coefficients; otherwise, individual variance estimates
`num.thread`	the number of (CPU) cores used; if `NA`, detect the number of cores automatically
`with.id`	if `TRUE`, the returned value with `sample.id` and `sample.id`
`useMatrix`	if `TRUE`, use `Matrix::dspMatrix` to store the output square matrix to save memory
`beta`	the object returned from `snpgdsIndivBeta()`
`beta_rel`	the beta-based matrix is generated relative to `beta_rel`
`verbose`	if `TRUE`, show information

Value

Return a list if `with.id = TRUE`:

`sample.id`	the sample ids used in the analysis
`snp.id`	the SNP ids used in the analysis
`inbreeding`	a logical value; `TRUE`, the diagonal is a vector of inbreeding coefficients; otherwise, individual variance estimates
`beta`	beta estimates
`avg_val`	the average of M_B among all loci, it could be used to calculate each M_ij

If `with.id = FALSE`, this function returns the genetic relationship matrix without sample and SNP IDs.

Author(s)

Xiuwen Zheng

References

Weir BS, Zheng X. SNPs and SNVs in Forensic Science. Forensic Science International: Genetics Supplement Series. 2015. doi:10.1016/j.fsigss.2015.09.106

Weir BS, Goudet J. A Unified Characterization of Population Structure and Relatedness. Genetics. 2017 Aug;206(4):2085-2103. doi: 10.1534/genetics.116.198424.

See Also

`snpgdsGRM`, `snpgdsIndInb`, `snpgdsFst`

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

b <- snpgdsIndivBeta(genofile, inbreeding=FALSE)
b$beta[1:10, 1:10]

z <- snpgdsIndivBetaRel(b, min(b$beta))

# close the file
snpgdsClose(genofile)
```

`snpgdsLDMat` *Linkage Disequilibrium (LD) analysis*

Description

Return a LD matrix for SNP pairs.

Usage

```
snpgdsLDMat(gdsobj, sample.id=NULL, snp.id=NULL, slide=250L,
    method=c("composite", "r", "dprime", "corr", "cov"), mat.trim=FALSE,
    num.thread=1L, with.id=TRUE, verbose=TRUE)
```

Arguments

`gdsobj`	an object of class `SNPGDSFileClass`, a SNP GDS file
`sample.id`	a vector of sample id specifying selected samples; if NULL, all samples are used
`snp.id`	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
`slide`	# of SNPs, the size of sliding window; if `slide < 0`, return a full LD matrix; see details
`method`	"composite", "r", "dprime", "corr", "cov", see details
`mat.trim`	if `TRUE`, trim the matrix when `slide > 0`: the function returns a `"num_slide x (n_snp - slide)"` matrix
`num.thread`	the number of (CPU) cores used; if `NA`, detect the number of cores automatically
`with.id`	if `TRUE`, the returned value with `sample.id` and `sample.id`
`verbose`	if TRUE, show information

Details

Four methods can be used to calculate linkage disequilibrium values: "composite" for LD composite measure, "r" for R coefficient (by EM algorithm assuming HWE, it could be negative), "dprime" for D', and "corr" for correlation coefficient. The method "corr" is equivalent to "composite", when SNP genotypes are coded as: 0 – BB, 1 – AB, 2 – AA.

If `slide <= 0`, the function returns a n-by-n LD matrix where the value of i row and j column is LD of i and j SNPs. If `slide > 0`, it returns a m-by-n LD matrix where n is the number of SNPs, m is the size of sliding window, and the value of i row and j column is LD of j and j+i SNPs.

Value

Return a list:

`sample.id`	the sample ids used in the analysis
`snp.id`	the SNP ids used in the analysis
`LD`	a matrix of LD values
`slide`	the size of sliding window

Author(s)

Xiuwen Zheng

References

Weir B: Inferences about linkage disequilibrium. Biometrics 1979; 35: 235-254.

Weir B: Genetic Data Analysis II. Sunderland, MA: Sinauer Associates, 1996.

Weir BS, Cockerham CC: Complete characterization of disequilibrium at two loci; in Feldman MW (ed): Mathematical Evolutionary Theory. Princeton, NJ: Princeton University Press, 1989.

See Also

snpgdsLDpair, snpgdsLDpruning

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# missing proportion and MAF
ff <- snpgdsSNPRateFreq(genofile)

# chromosome 15
snpset <- read.gdsn(index.gdsn(genofile, "snp.id"))[
    ff$MissingRate==0 & ff$MinorFreq>0 &
    read.gdsn(index.gdsn(genofile, "snp.chromosome"))==15]
length(snpset)


# LD matrix without sliding window
ld.noslide <- snpgdsLDMat(genofile, snp.id=snpset, slide=-1, method="composite")
# plot
image(t(ld.noslide$LD^2), col=terrain.colors(16))

# LD matrix with a sliding window
ld.slide <- snpgdsLDMat(genofile, snp.id=snpset, method="composite")
# plot
image(t(ld.slide$LD^2), col=terrain.colors(16))


# close the genotype file
snpgdsClose(genofile)
```

---

snpgdsLDpair *Linkage Disequilibrium (LD)*

Description

Return a LD value between snp1 and snp2.

Usage

```
snpgdsLDpair(snp1, snp2, method = c("composite", "r", "dprime", "corr"))
```

Arguments

snp1	a vector of SNP genotypes (0 – BB, 1 – AB, 2 – AA)
snp2	a vector of SNP genotypes (0 – BB, 1 – AB, 2 – AA)
method	"composite", "r", "dprime", "corr", see details

Details

Four methods can be used to calculate linkage disequilibrium values: "composite" for LD composite measure, "r" for R coefficient (by EM algorithm assuming HWE, it could be negative), "dprime" for D', and "corr" for correlation coefficient. The method "corr" is equivalent to "composite", when SNP genotypes are coded as: 0 – BB, 1 – AB, 2 – AA.

Value

Return a numeric vector:

ld	a measure of linkage disequilibrium

if method = "r" or "dprime",

pA_A	haplotype frequency of AA, the first locus is A and the second locus is A
pA_B	haplotype frequency of AB, the first locus is A and the second locus is B
pB_A	haplotype frequency of BA, the first locus is B and the second locus is A
pB_B	haplotype frequency of BB, the first locus is B and the second locus is B

Author(s)

Xiuwen Zheng

References

Weir B: Inferences about linkage disequilibrium. Biometrics 1979; 35: 235-254.

Weir B: Genetic Data Analysis II. Sunderland, MA: Sinauer Associates, 1996.

Weir BS, Cockerham CC: Complete characterization of disequilibrium at two loci; in Feldman MW (ed): Mathematical Evolutionary Theory. Princeton, NJ: Princeton University Press, 1989.

See Also

snpgdsLDMat, snpgdsLDpruning

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

snp1 <- read.gdsn(index.gdsn(genofile, "genotype"), start=c(1,1), count=c(1,-1))
snp2 <- read.gdsn(index.gdsn(genofile, "genotype"), start=c(2,1), count=c(1,-1))

snpgdsLDpair(snp1, snp2, method = "composite")
snpgdsLDpair(snp1, snp2, method = "r")
snpgdsLDpair(snp1, snp2, method = "dprime")
snpgdsLDpair(snp1, snp2, method = "corr")

# close the genotype file
snpgdsClose(genofile)
```

---

snpgdsLDpruning *Linkage Disequilibrium (LD) based SNP pruning*

Description

Recursively removes SNPs within a sliding window

Usage

```
snpgdsLDpruning(gdsobj, sample.id=NULL, snp.id=NULL, autosome.only=TRUE,
    remove.monosnp=TRUE, maf=0.005, missing.rate=0.05,
    method=c("composite", "r", "dprime", "corr"), slide.max.bp=500000L,
    slide.max.n=NA, ld.threshold=0.2,
    start.pos=c("random.f500", "random", "first", "last"),
    num.thread=1L, autosave=NULL, verbose=TRUE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
autosome.only	if TRUE, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
remove.monosnp	if TRUE, remove monomorphic SNPs
maf	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
missing.rate	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
method	"composite", "r", "dprime", "corr", see details
slide.max.bp	the maximum basepairs in the sliding window
slide.max.n	the maximum number of SNPs in the sliding window
ld.threshold	the LD threshold

start.pos	"random.f500", a starting postion randomly selected from the first 500 markers (by default); "random": a random starting position; "first": start from the first position; "last": start from the last position. "random.f500" is applicable for >= v1.37.2
num.thread	the number of (CPU) cores used; if NA, detect the number of cores automatically
autosave	NULL or a file name for autosaving a single R object (saving via saveRDS)
verbose	if TRUE, show information

Details

The minor allele frequency and missing rate for each SNP passed in snp.id are calculated over all the samples in sample.id.

Four methods can be used to calculate linkage disequilibrium values: "composite" for LD composite measure, "r" for R coefficient (by EM algorithm assuming HWE, it could be negative), "dprime" for D', and "corr" for correlation coefficient. The method "corr" is equivalent to "composite", when SNP genotypes are coded as: 0 – BB, 1 – AB, 2 – AA. The argument ld.threshold is the absolute value of measurement.

It is useful to generate a pruned subset of SNPs that are in approximate linkage equilibrium with each other. The function snpgdsLDpruning recursively removes SNPs within a sliding window based on the pairwise genotypic correlation. SNP pruning is conducted chromosome by chromosome, since SNPs in a chromosome can be considered to be independent with the other chromosomes.

The pruning algorithm on a chromosome is described as follows (n is the total number of SNPs on that chromosome):

1) Randomly select a starting position i (start.pos="random"), i=1 if start.pos="first", or i=last if start.pos="last"; and let the current SNP set S={ i };

2) For each right position j from i+1 to n: if any LD between j and k is greater than ld.threshold, where k belongs to S, and both of j and k are in the sliding window, then skip j; otherwise, let S be S + { j };

3) For each left position j from i-1 to 1: if any LD between j and k is greater than ld.threshold, where k belongs to S, and both of j and k are in the sliding window, then skip j; otherwise, let S be S + { j };

4) Output S, the final selection of SNPs.

Value

Return a list of SNP IDs stratified by chromosomes.

Author(s)

Xiuwen Zheng

References

Weir B: Inferences about linkage disequilibrium. Biometrics 1979; 35: 235-254.

Weir B: Genetic Data Analysis II. Sunderland, MA: Sinauer Associates, 1996.

Weir BS, Cockerham CC: Complete characterization of disequilibrium at two loci; in Feldman MW (ed): Mathematical Evolutionary Theory. Princeton, NJ: Princeton University Press, 1989.

See Also

snpgdsLDMat, snpgdsLDpair

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

set.seed(1000)
snpset <- snpgdsLDpruning(genofile)
str(snpset)
names(snpset)
#  [1] "chr1"  "chr2"  "chr3"  "chr4"  "chr5"  "chr6"  "chr7"  "chr8"  "chr9"
# [10] "chr10" "chr11" "chr12" "chr13" "chr14" "chr15" "chr16" "chr17" "chr18"
# ......

# get SNP ids
snp.id <- unlist(unname(snpset))

# close the genotype file
snpgdsClose(genofile)
```

---

snpgdsMergeGRM *Merge Multiple Genetic Relationship Matrices (GRM)*

Description

Combine multiple genetic relationship matrices with weighted averaging.

Usage

```
snpgdsMergeGRM(filelist, out.fn=NULL, out.prec=c("double", "single"),
    out.compress="LZMA_RA", weight=NULL, verbose=TRUE)
```

Arguments

filelist	a character vector, list of GDS file names
out.fn	NULL, return a GRM object; or characters, the output GDS file name
out.prec	double or single precision for storage
out.compress	the compression method for storing the GRM matrix in the GDS file
weight	NULL, weights proportional to the numbers of SNPs; a numeric vector, or a logical vector (FALSE for excluding some GRMs with a negative weight, weights proportional to the numbers of SNPs)
verbose	if TRUE, show information

Details

The final GRM is the weighted averaged matrix combining multiple GRMs. The merged GRM may not be identical to the GRM calculated using full SNPs, due to missing genotypes or the internal weighting strategy of the specified GRM calculation.

Value

None or a GRM object if `out.fn=NULL`.

Author(s)

Xiuwen Zheng

See Also

snpgdsGRM

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

snpid <- read.gdsn(index.gdsn(genofile, "snp.id"))
snpid <- snpid[snpgdsSNPRateFreq(genofile)$MissingRate == 0]

# there is no missing genotype
grm <- snpgdsGRM(genofile, snp.id=snpid, method="GCTA")


# save two GRMs
set1 <- grm$snp.id[1:(length(grm$snp.id)/2)]
set2 <- setdiff(grm$snp.id, set1)
snpgdsGRM(genofile, method="GCTA", snp.id=set1, out.fn="tmp1.gds")
snpgdsGRM(genofile, method="GCTA", snp.id=set2, out.fn="tmp2.gds")

# merge GRMs and export to a new GDS file
snpgdsMergeGRM(c("tmp1.gds", "tmp2.gds"), "tmp.gds")

# return the GRM
grm2 <- snpgdsMergeGRM(c("tmp1.gds", "tmp2.gds"))


# check
f <- openfn.gds("tmp.gds")
m <- read.gdsn(index.gdsn(f, "grm"))
closefn.gds(f)

summary(c(m - grm$grm))  # ~zero
summary(c(m - grm2$grm))  # zero


# close the file
```

```
snpgdsClose(genofile)

# delete the temporary file
unlink(c("tmp1.gds", "tmp2.gds", "tmp.gds"), force=TRUE)
```

---

snpgdsOpen *Open a SNP GDS File*

Description

Open a SNP GDS file

Usage

```
snpgdsOpen(filename, readonly=TRUE, allow.duplicate=FALSE, allow.fork=TRUE)
```

Arguments

filename	the file name
readonly	whether read-only or not
allow.duplicate	if TRUE, it is allowed to open a GDS file with read-only mode when it has been opened in the same R session, see openfn.gds
allow.fork	TRUE for parallel environment using forking, see openfn.gds

Details

It is strongly suggested to call snpgdsOpen instead of openfn.gds, since snpgdsOpen will perform internal checking for data integrality.

Value

Return an object of class SNPGDSFileClass.

Author(s)

Xiuwen Zheng

See Also

snpgdsClose

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

genofile

# close the file
snpgdsClose(genofile)
```

---

snpgdsOption *Option settings: chromosome coding, etc*

Description

Return an option list used by the SNPRelate package or a GDS file

Usage

```
snpgdsOption(gdsobj=NULL, autosome.start=1L, autosome.end=22L, ...)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
autosome.start	the starting index of autosome
autosome.end	the ending index of autosome
...	optional arguments for new chromosome coding

Value

A list

Author(s)

Xiuwen Zheng

Examples

```
# define the new chromosomes 'Z' and 'W'
snpgdsOption(Z=27L, W=28L)


# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

snpgdsOption(genofile)

# close the genotype file
snpgdsClose(genofile)
```

snpgdsPairIBD *Calculate Identity-By-Descent (IBD) Coefficients*

Description

Calculate the three IBD coefficients (k0, k1, k2) for non-inbred individual pairs by Maximum Likelihood Estimation (MLE) or PLINK Method of Moment (MoM).

Usage

```
snpgdsPairIBD(geno1, geno2, allele.freq,
    method=c("EM", "downhill.simplex", "MoM", "Jacquard"),
   kinship.constraint=FALSE, max.niter=1000L, reltol=sqrt(.Machine$double.eps),
    coeff.correct=TRUE, out.num.iter=TRUE, verbose=TRUE)
```

Arguments

geno1	the SNP genotypes for the first individual, 0 – BB, 1 – AB, 2 – AA, other values – missing
geno2	the SNP genotypes for the second individual, 0 – BB, 1 – AB, 2 – AA, other values – missing
allele.freq	the allele frequencies
method	"EM", "downhill.simplex", "MoM" or "Jacquard", see details
kinship.constraint	if TRUE, constrict IBD coefficients (k_0,k_1,k_2) in the genealogical region ($2 k_0 k_1 >= k_2^2$)
max.niter	the maximum number of iterations
reltol	relative convergence tolerance; the algorithm stops if it is unable to reduce the value of log likelihood by a factor of $reltol * (abs(log likelihood with the initial parameters) + reltol)$ at a step.
coeff.correct	TRUE by default, see details
out.num.iter	if TRUE, output the numbers of iterations
verbose	if TRUE, show information

Details

If method = "MoM", then PLINK Method of Moment without a allele-count-based correction factor is conducted. Otherwise, two numeric approaches for maximum likelihood estimation can be used: one is Expectation-Maximization (EM) algorithm, and the other is Nelder-Mead method or downhill simplex method. Generally, EM algorithm is more robust than downhill simplex method. "Jacquard" refers to the estimation of nine Jacquard's coefficients.

If coeff.correct is TRUE, the final point that is found by searching algorithm (EM or downhill simplex) is used to compare the six points (fullsib, offspring, halfsib, cousin, unrelated), since any numeric approach might not reach the maximum position after a finit number of steps. If any of these six points has a higher value of log likelihood, the final point will be replaced by the best one.

Value

Return a `data.frame`:

`k0`	IBD coefficient, the probability of sharing ZERO IBD
`k1`	IBD coefficient, the probability of sharing ONE IBD
`loglik`	the value of log likelihood
`niter`	the number of iterations

Author(s)

Xiuwen Zheng

References

Milligan BG. 2003. Maximum-likelihood estimation of relatedness. Genetics 163:1153-1167.

Weir BS, Anderson AD, Hepler AB. 2006. Genetic relatedness analysis: modern data and new challenges. Nat Rev Genet. 7(10):771-80.

Choi Y, Wijsman EM, Weir BS. 2009. Case-control association testing in the presence of unknown relationships. Genet Epidemiol 33(8):668-78.

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ & Sham PC. 2007. PLINK: a toolset for whole-genome association and population-based linkage analysis. American Journal of Human Genetics, 81.

See Also

`snpgdsPairIBDMLELogLik`, `snpgdsIBDMLE`, `snpgdsIBDMLELogLik`, `snpgdsIBDMoM`

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

YRI.id <- read.gdsn(index.gdsn(genofile, "sample.id"))[
    read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))=="YRI"]

# SNP pruning
set.seed(10)
snpset <- snpgdsLDpruning(genofile, sample.id=YRI.id, maf=0.05,
    missing.rate=0.05)
snpset <- unname(sample(unlist(snpset), 250))

# the number of samples
n <- 25

# specify allele frequencies
RF <- snpgdsSNPRateFreq(genofile, sample.id=YRI.id, snp.id=snpset,
    with.id=TRUE)
summary(RF$AlleleFreq)
```

```
subMLE <- snpgdsIBDMLE(genofile, sample.id=YRI.id[1:n], snp.id=RF$snp.id,
    allele.freq=RF$AlleleFreq)
subMoM <- snpgdsIBDMoM(genofile, sample.id=YRI.id[1:n], snp.id=RF$snp.id,
    allele.freq=RF$AlleleFreq)
subJac <- snpgdsIBDMLE(genofile, sample.id=YRI.id[1:n], snp.id=RF$snp.id,
    allele.freq=RF$AlleleFreq, method="Jacquard")



########################

# genotype matrix
mat <- snpgdsGetGeno(genofile, sample.id=YRI.id[1:n], snp.id=snpset,
    snpfirstdim=TRUE)

rv <- NULL
for (i in 2:n)
{
    rv <- rbind(rv, snpgdsPairIBD(mat[,1], mat[,i], RF$AlleleFreq, "EM"))
    print(snpgdsPairIBDMLELogLik(mat[,1], mat[,i], RF$AlleleFreq,
        relatedness="unrelated", verbose=TRUE))
}
rv
summary(rv$k0 - subMLE$k0[1, 2:n])
summary(rv$k1 - subMLE$k1[1, 2:n])
# ZERO

rv <- NULL
for (i in 2:n)
    rv <- rbind(rv, snpgdsPairIBD(mat[,1], mat[,i], RF$AlleleFreq, "MoM"))
rv
summary(rv$k0 - subMoM$k0[1, 2:n])
summary(rv$k1 - subMoM$k1[1, 2:n])
# ZERO

rv <- NULL
for (i in 2:n)
    rv <- rbind(rv, snpgdsPairIBD(mat[,1], mat[,i], RF$AlleleFreq, "Jacquard"))
rv
summary(rv$D1 - subJac$D1[1, 2:n])
summary(rv$D2 - subJac$D2[1, 2:n])
# ZERO

# close the genotype file
snpgdsClose(genofile)
```

---

snpgdsPairIBDMLELogLik

Log likelihood for MLE method in the Identity-By-Descent (IBD) Analysis

---

Description

Calculate the log likelihood values from maximum likelihood estimation.

Usage

```
snpgdsPairIBDMLELogLik(geno1, geno2, allele.freq, k0=NaN, k1=NaN,
    relatedness=c("", "self", "fullsib", "offspring", "halfsib",
    "cousin", "unrelated"), verbose=TRUE)
```

Arguments

geno1	the SNP genotypes for the first individual, 0 – BB, 1 – AB, 2 – AA, other values – missing
geno2	the SNP genotypes for the second individual, 0 – BB, 1 – AB, 2 – AA, other values – missing
allele.freq	the allele frequencies
k0	specified IBD coefficient
k1	specified IBD coefficient
relatedness	specify a relatedness, otherwise use the values of k0 and k1
verbose	if TRUE, show information

Details

If (relatedness == "") and (k0 == NaN or k1 == NaN), then return the log likelihood values for each (k0, k1) stored in ibdobj.

If (relatedness == "") and (k0 != NaN) and (k1 != NaN), then return the log likelihood values for a specific IBD coefficient (k0, k1).

If relatedness is: "self", then k0 = 0, k1 = 0; "fullsib", then k0 = 0.25, k1 = 0.5; "offspring", then k0 = 0, k1 = 1; "halfsib", then k0 = 0.5, k1 = 0.5; "cousin", then k0 = 0.75, k1 = 0.25; "unrelated", then k0 = 1, k1 = 0.

Value

The value of log likelihood.

Author(s)

Xiuwen Zheng

References

Milligan BG. 2003. Maximum-likelihood estimation of relatedness. Genetics 163:1153-1167.

Weir BS, Anderson AD, Hepler AB. 2006. Genetic relatedness analysis: modern data and new challenges. Nat Rev Genet. 7(10):771-80.

Choi Y, Wijsman EM, Weir BS. 2009. Case-control association testing in the presence of unknown relationships. Genet Epidemiol 33(8):668-78.

See Also

snpgdsPairIBD, snpgdsIBDMLE, snpgdsIBDMLELogLik, snpgdsIBDMoM

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

YRI.id <- read.gdsn(index.gdsn(genofile, "sample.id"))[
    read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))=="YRI"]

# SNP pruning
set.seed(10)
snpset <- snpgdsLDpruning(genofile, sample.id=YRI.id, maf=0.05,
    missing.rate=0.05)
snpset <- unname(sample(unlist(snpset), 250))

# the number of samples
n <- 25

# specify allele frequencies
RF <- snpgdsSNPRateFreq(genofile, sample.id=YRI.id, snp.id=snpset,
    with.id=TRUE)
summary(RF$AlleleFreq)

subMLE <- snpgdsIBDMLE(genofile, sample.id=YRI.id[1:n], snp.id=RF$snp.id,
    allele.freq=RF$AlleleFreq)
subMoM <- snpgdsIBDMoM(genofile, sample.id=YRI.id[1:n], snp.id=RF$snp.id,
    allele.freq=RF$AlleleFreq)


# genotype matrix
mat <- snpgdsGetGeno(genofile, sample.id=YRI.id[1:n], snp.id=snpset,
    snpfirstdim=TRUE)


########################

rv <- NULL
for (i in 2:n)
{
    rv <- rbind(rv, snpgdsPairIBD(mat[,1], mat[,i], RF$AlleleFreq, "EM"))
    print(snpgdsPairIBDMLELogLik(mat[,1], mat[,i], RF$AlleleFreq,
        relatedness="unrelated", verbose=TRUE))
}
rv
summary(rv$k0 - subMLE$k0[1, 2:n])
summary(rv$k1 - subMLE$k1[1, 2:n])
# ZERO

rv <- NULL
for (i in 2:n)
```

```
    rv <- rbind(rv, snpgdsPairIBD(mat[,1], mat[,i], RF$AlleleFreq, "MoM"))
rv
summary(rv$k0 - subMoM$k0[1, 2:n])
summary(rv$k1 - subMoM$k1[1, 2:n])
# ZERO

# close the genotype file
snpgdsClose(genofile)
```

---

snpgdsPairScore *Genotype Score for Pairs of Individuals*

Description

Calculate the genotype score for pairs of individuals based on identity-by-state (IBS) measure

Usage

```
snpgdsPairScore(gdsobj, sample1.id, sample2.id, snp.id=NULL,
    method=c("IBS", "GVH", "HVG", "GVH.major", "GVH.minor", "GVH.major.only",
    "GVH.minor.only"), type=c("per.pair", "per.snp", "matrix", "gds.file"),
    dosage=TRUE, with.id=TRUE, output=NULL, verbose=TRUE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample1.id	a vector of sample id specifying selected samples; if NULL, all samples are used
sample2.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
method	"IBS" – identity-by-state score, "GVH" or "HVG", see Details
type	"per.pair", "per.snp" or "matrix", see Value
dosage	TRUE, uses dosages 0, 1, 2; FALSE, uses 0, 1 (changing a return value of 1 or 2 to be 1)
with.id	if TRUE, returns "sample.id" and "snp.id"; see Value
output	if type="gds.file", the file name
verbose	if TRUE, show information

Details

sample1.id	sample2.id							
Patient	Donor	IBS	GVH	HVG	GVH.major	GVH.minor	GVH.major.only	GVH.minor.only
AA / 2	AA / 2	2	0	0	0	0	0	0
AA / 2	AB / 1	1	0	1	0	0	0	0
AA / 2	BB / 0	0	2	2	1	0	1	NA

AB / 1	AA / 2	1	1	0	0	1	NA	1
AB / 1	AB / 1	2	0	0	0	0	0	0
AB / 1	BB / 0	1	1	0	1	0	1	NA
BB / 0	AA / 2	0	2	2	0	1	NA	1
BB / 0	AB / 1	1	0	1	0	0	0	0
BB / 0	BB / 0	2	0	0	0	0	0	0

Value

Return a list:

`sample.id` the sample ids used in the analysis, if `with.id=TRUE`

`snp.id` the SNP ids used in the analysis, if `with.id=TRUE`

`score` a matrix of genotype score: if `type="per.pair"`, a `data.frame` with the first column for average scores, the second column for standard deviation and the third column for the valid number of SNPs; the additional columns for pairs of samples. if `type="per.snp"`, a 3-by-# `of SNPs` matrix with the first row for average scores, the second row for standard deviation and the third row for the valid number of individual pairs; if `type="matrix"`, a # `of pairs`-by-# `of SNPs` matrix with rows for pairs of individuals

Author(s)

Xiuwen Zheng

References

Warren, E. H., Zhang, X. C., Li, S., Fan, W., Storer, B. E., Chien, J. W., Boeckh, M. J., et al. (2012). Effect of MHC and non-MHC donor/recipient genetic disparity on the outcome of allogeneic HCT. Blood, 120(14), 2796-806. doi:10.1182/blood-2012-04-347286

See Also

snpgdsIBS

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# autosomal SNPs
selsnp <- snpgdsSelectSNP(genofile, autosome.only=TRUE, remove.monosnp=FALSE)

# sample ID
sample.id <- read.gdsn(index.gdsn(genofile, "sample.id"))
father.id <- read.gdsn(index.gdsn(genofile, "sample.annot/father.id"))
```

```
offspring.id <- sample.id[father.id != ""]
father.id <- father.id[father.id != ""]


# calculate average genotype scores
z1 <- snpgdsPairScore(genofile, offspring.id, father.id, snp.id=selsnp,
    method="IBS", type="per.pair")
str(z1)
head(z1$score)

# calculate average genotype scores
z1 <- snpgdsPairScore(genofile, offspring.id, father.id, snp.id=selsnp,
    method="IBS", type="per.pair", dosage=FALSE)
str(z1)
head(z1$score)


# calculate average genotype scores
z2 <- snpgdsPairScore(genofile, offspring.id, father.id, snp.id=selsnp,
    method="IBS", type="per.snp")
str(z2)
z2$score[, 1:4]
mean(z2$score["Avg",])
mean(z2$score["SD",])

plot(z2$score["Avg",], pch=20, cex=0.75, xlab="SNP Index", ylab="IBS score")


# calculate a matrix of genotype scores over samples and SNPs
z3 <- snpgdsPairScore(genofile, offspring.id, father.id, snp.id=selsnp,
    method="IBS", type="matrix")
str(z3)


# output the score matrix to a GDS file
snpgdsPairScore(genofile, offspring.id, father.id, snp.id=selsnp,
    method="IBS", type="gds.file", output="tmp.gds")
(f <- snpgdsOpen("tmp.gds"))
snpgdsClose(f)


# close the file
snpgdsClose(genofile)

unlink("tmp.gds", force=TRUE)
```

---

snpgdsPCA *Principal Component Analysis (PCA) on SNP genotype data*

Description

To calculate the eigenvectors and eigenvalues for principal component analysis in GWAS.

Usage

```
snpgdsPCA(gdsobj, sample.id=NULL, snp.id=NULL,
    autosome.only=TRUE, remove.monosnp=TRUE, maf=NaN, missing.rate=NaN,
    algorithm=c("exact", "randomized"),
    eigen.cnt=ifelse(identical(algorithm, "randomized"), 16L, 32L),
    num.thread=1L, bayesian=FALSE, need.genmat=FALSE,
    genmat.only=FALSE, eigen.method=c("DSPEVX", "DSPEV"),
    aux.dim=eigen.cnt*2L, iter.num=10L, verbose=TRUE)
## S3 method for class 'snpgdsPCAClass'
plot(x, eig=c(1L,2L), ...)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
autosome.only	if TRUE, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
remove.monosnp	if TRUE, remove monomorphic SNPs
maf	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
missing.rate	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
eigen.cnt	output the number of eigenvectors; if eigen.cnt <= 0, then return all eigenvectors
algorithm	"exact", traditional exact calculation; "randomized", fast PCA with randomized algorithm introduced in Galinsky et al. 2016
num.thread	the number of (CPU) cores used; if NA, detect the number of cores automatically
bayesian	if TRUE, use bayesian normalization
need.genmat	if TRUE, return the genetic covariance matrix
genmat.only	return the genetic covariance matrix only, do not compute the eigenvalues and eigenvectors
eigen.method	"DSPEVX" – compute the top eigen.cnt eigenvalues and eigenvectors using LAPACK::DSPEVX; "DSPEV" – to be compatible with SNPRelate_1.1.6 or earlier, using LAPACK::DSPEV; "DSPEVX" is significantly faster than "DSPEV" if only top principal components are of interest
aux.dim	auxiliary dimension used in fast randomized algorithm
iter.num	iteration number used in fast randomized algorithm
verbose	if TRUE, show information
x	a snpgdsPCAClass object
eig	indices of eigenvectors, like 1:2 or 1:4
...	the arguments passed to or from other methods, like pch, col

Details

The minor allele frequency and missing rate for each SNP passed in snp.id are calculated over all the samples in sample.id.

Value

Return a `snpgdsPCAClass` object, and it is a list:

`sample.id`	the sample ids used in the analysis
`snp.id`	the SNP ids used in the analysis
`eigenval`	eigenvalues
`eigenvect`	eigenvactors, "# of samples" x "eigen.cnt"
`varprop`	variance proportion for each principal component
`TraceXTX`	the trace of the genetic covariance matrix
`Bayesian`	whether use bayerisan normalization
`genmat`	the genetic covariance matrix

Author(s)

Xiuwen Zheng

References

Patterson N, Price AL, Reich D. Population structure and eigenanalysis. PLoS Genet. 2006 Dec;2(12):e190.

Galinsky KJ, Bhatia G, Loh PR, Georgiev S, Mukherjee S, Patterson NJ, Price AL. Fast Principal-Component Analysis Reveals Convergent Evolution of ADH1B in Europe and East Asia. Am J Hum Genet. 2016 Mar 3;98(3):456-72. doi: 10.1016/j.ajhg.2015.12.022. Epub 2016 Feb 25.

See Also

`snpgdsPCACorr`, `snpgdsPCASNPLoading`, `snpgdsPCASampLoading`, `snpgdsAdmixProp`, `snpgdsEIGMIX`

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# run PCA
RV <- snpgdsPCA(genofile)
RV

# eigenvalues
head(RV$eigenval)

# variance proportion (%)
head(round(RV$varprop*100, 2))
# [1] 12.23  5.84  1.01  0.95  0.84  0.74

# draw
plot(RV)
plot(RV, 1:4)
```

```
####  there is no population information  ####

# make a data.frame
tab <- data.frame(sample.id = RV$sample.id,
    EV1 = RV$eigenvect[,1],    # the first eigenvector
    EV2 = RV$eigenvect[,2],    # the second eigenvector
    stringsAsFactors = FALSE)
head(tab)
#   sample.id         EV1         EV2
# 1   NA19152 -0.08411287 -0.01226860
# 2   NA19139 -0.08360644 -0.01085849
# 3   NA18912 -0.08110808 -0.01184524
# 4   NA19160 -0.08680864 -0.01447106
# 5   NA07034  0.03109761  0.07709255
# 6   NA07055  0.03228450  0.08155730

# draw
plot(tab$EV2, tab$EV1, xlab="eigenvector 2", ylab="eigenvector 1")



####  there are population information  ####

# get population information
#   or pop_code <- scan("pop.txt", what=character())
#   if it is stored in a text file "pop.txt"
pop_code <- read.gdsn(index.gdsn(genofile, "sample.annot/pop.group"))

# get sample id
samp.id <- read.gdsn(index.gdsn(genofile, "sample.id"))

# assume the order of sample IDs is as the same as population codes
cbind(samp.id, pop_code)
#        samp.id       pop_code
#   [1,] "NA19152"     "YRI"
#   [2,] "NA19139"     "YRI"
#   [3,] "NA18912"     "YRI"
#   [4,] "NA19160"     "YRI"
#   [5,] "NA07034"     "CEU"
#   ...

# make a data.frame
tab <- data.frame(sample.id = RV$sample.id,
    pop = factor(pop_code)[match(RV$sample.id, samp.id)],
    EV1 = RV$eigenvect[,1],    # the first eigenvector
    EV2 = RV$eigenvect[,2],    # the second eigenvector
    stringsAsFactors = FALSE)
head(tab)
#   sample.id pop         EV1         EV2
# 1   NA19152 YRI -0.08411287 -0.01226860
# 2   NA19139 YRI -0.08360644 -0.01085849
# 3   NA18912 YRI -0.08110808 -0.01184524
```

```
# 4   NA19160 YRI -0.08680864 -0.01447106
# 5   NA07034 CEU  0.03109761  0.07709255
# 6   NA07055 CEU  0.03228450  0.08155730

# draw
plot(tab$EV2, tab$EV1, col=as.integer(tab$pop),
    xlab="eigenvector 2", ylab="eigenvector 1")
legend("bottomright", legend=levels(tab$pop), pch="o", col=1:4)


# close the file
snpgdsClose(genofile)
```

`snpgdsPCACorr` *PC-correlated SNPs in principal component analysis*

Description

To calculate the SNP correlations between eigenvactors and SNP genotypes

Usage

```
snpgdsPCACorr(pcaobj, gdsobj, snp.id=NULL, eig.which=NULL, num.thread=1L,
    with.id=TRUE, outgds=NULL, verbose=TRUE)
```

Arguments

`pcaobj`	a `snpgdsPCAClass` object returned from the function snpgdsPCA, a `snpgdsEigMixClass` from snpgdsEIGMIX, or an eigenvector matrix with row names (sample id)
`gdsobj`	an object of class `SNPGDSFileClass`, a SNP GDS file
`snp.id`	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
`eig.which`	a vector of integers, to specify which eigenvectors to be used
`num.thread`	the number of (CPU) cores used; if `NA`, detect the number of cores automatically
`with.id`	if `TRUE`, the returned value with `sample.id` and `sample.id`
`outgds`	`NULL` or a character of file name for exporting correlations to a GDS file, see details
`verbose`	if TRUE, show information

Details

If an output file name is specified via `outgds`, "sample.id", "snp.id" and "correlation" will be stored in the GDS file. The GDS node "correlation" is a matrix of correlation coefficients, and it is stored with the format of packed real number ("packedreal16" preserving 4 digits, 0.0001 is the smallest number greater zero, see add.gdsn).

Value

Return a list if `outgds=NULL`,

`sample.id`	the sample ids used in the analysis
`snp.id`	the SNP ids used in the analysis
`snpcorr`	a matrix of correlation coefficients, "# of eigenvectors" x "# of SNPs"

Author(s)

Xiuwen Zheng

References

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

See Also

snpgdsPCA, snpgdsPCASampLoading, snpgdsPCASNPLoading

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())
# get chromosome index
chr <- read.gdsn(index.gdsn(genofile, "snp.chromosome"))

pca <- snpgdsPCA(genofile)
cr <- snpgdsPCACorr(pca, genofile, eig.which=1:4)
plot(abs(cr$snpcorr[3,]), xlab="SNP Index", ylab="PC 3", col=chr)


# output to a gds file if limited memory
snpgdsPCACorr(pca, genofile, eig.which=1:4, outgds="test.gds")

(f <- openfn.gds("test.gds"))
m <- read.gdsn(index.gdsn(f, "correlation"))
closefn.gds(f)

# check
summary(c(m - cr$snpcorr))  # should < 1e-4


# close the file
snpgdsClose(genofile)

# delete the temporary file
unlink("test.gds", force=TRUE)
```

`snpgdsPCASampLoading` *Project individuals onto existing principal component axes*

Description

To calculate the sample eigenvectors using the specified SNP loadings

Usage

```
snpgdsPCASampLoading(loadobj, gdsobj, sample.id=NULL, num.thread=1L,
    verbose=TRUE)
```

Arguments

`loadobj`	a `snpgdsPCASNPLoadingClass` or `snpgdsEigMixSNPLoadingClass` object returned from snpgdsPCASNPLoading
`gdsobj`	an object of class `SNPGDSFileClass`, a SNP GDS file
`sample.id`	a vector of sample id specifying selected samples; if NULL, all samples are used
`num.thread`	the number of CPU cores used
`verbose`	if TRUE, show information

Details

The `sample.id` are usually different from the samples used in the calculation of SNP loadings.

Value

Returns a `snpgdsPCAClass` object, and it is a list:

`sample.id`	the sample ids used in the analysis
`snp.id`	the SNP ids used in the analysis
`eigenval`	eigenvalues
`eigenvect`	eigenvactors, "# of samples" x "eigen.cnt"
`TraceXTX`	the trace of the genetic covariance matrix
`Bayesian`	whether use bayerisan normalization

Or returns a `snpgdsEigMixClass` object, and it is a list:

`sample.id`	the sample ids used in the analysis
`snp.id`	the SNP ids used in the analysis
`eigenval`	eigenvalues
`eigenvect`	eigenvactors, "# of samples" x "eigen.cnt"
`afreq`	allele frequencies

Author(s)

Xiuwen Zheng

References

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

Zhu, X., Li, S., Cooper, R. S., and Elston, R. C. (2008). A unified association analysis approach for family and unrelated samples correcting for stratification. Am J Hum Genet, 82(2), 352-365.

See Also

snpgdsPCA, snpgdsPCACorr, snpgdsPCASNPLoading

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

sample.id <- read.gdsn(index.gdsn(genofile, "sample.id"))

# first PCA
pca <- snpgdsPCA(genofile, eigen.cnt=8)
snp_load <- snpgdsPCASNPLoading(pca, genofile)

# calculate sample eigenvectors from SNP loadings
samp_load <- snpgdsPCASampLoading(snp_load, genofile, sample.id=sample.id[1:100])

diff <- pca$eigenvect[1:100,] - samp_load$eigenvect
summary(c(diff))
# ~ ZERO


# combine eigenvectors
allpca <- list(
    sample.id = c(pca$sample.id, samp_load$sample.id),
    snp.id = pca$snp.id,
    eigenval = c(pca$eigenval, samp_load$eigenval),
    eigenvect = rbind(pca$eigenvect, samp_load$eigenvect),
    varprop = c(pca$varprop, samp_load$varprop),
    TraceXTX = pca$TraceXTX
)
class(allpca) <- "snpgdsPCAClass"
allpca


# close the genotype file
snpgdsClose(genofile)
```

snpgdsPCASNPLoading *SNP loadings in principal component analysis*

Description

To calculate the SNP loadings in Principal Component Analysis

Usage

```
snpgdsPCASNPLoading(pcaobj, gdsobj, num.thread=1L, verbose=TRUE)
```

Arguments

pcaobj	a snpgdsPCAClass object returned from the function snpgdsPCA or a snpgdsEigMixClass from snpgdsEIGMIX
gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
num.thread	the number of (CPU) cores used; if NA, detect the number of cores automatically
verbose	if TRUE, show information

Details

Calculate the SNP loadings (or SNP eigenvectors) from the principal component analysis conducted in snpgdsPCA.

Value

Returns a snpgdsPCASNPLoading object if pcaobj is snpgdsPCAClass, which is a list:

sample.id	the sample ids used in the analysis
snp.id	the SNP ids used in the analysis
eigenval	eigenvalues
snploading	SNP loadings, or SNP eigenvectors
TraceXTX	the trace of the genetic covariance matrix
Bayesian	whether use bayerisan normalization
avgfreq	two times allele frequency used in snpgdsPCA
scale	internal parameter

Or returns a snpgdsEigMixSNPLoadingClass object if pcaobj is snpgdsEigMixClass, which is a list:

sample.id	the sample ids used in the analysis
snp.id	the SNP ids used in the analysis
eigenval	eigenvalues
snploading	SNP loadings, or SNP eigenvectors
afreq	allele frequency

Author(s)

Xiuwen Zheng

References

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

Price AL, Patterson NJ, Plenge RM, Weinblatt ME, Shadick NA, Reich D (2006) Principal components analysis corrects for stratification in genome-wide association studies. Nat Genet. 38, 904-909.

Zhu, X., Li, S., Cooper, R. S., and Elston, R. C. (2008). A unified association analysis approach for family and unrelated samples correcting for stratification. Am J Hum Genet, 82(2), 352-365.

See Also

snpgdsPCA, snpgdsEIGMIX, snpgdsPCASampLoading, snpgdsPCACorr

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

PCARV <- snpgdsPCA(genofile, eigen.cnt=8)
SnpLoad <- snpgdsPCASNPLoading(PCARV, genofile)

names(SnpLoad)
# [1] "sample.id"  "snp.id"     "eigenval"   "snploading" "TraceXTX"
# [6] "Bayesian"   "avgfreq"    "scale"
dim(SnpLoad$snploading)
# [1]     8 8722

plot(SnpLoad$snploading[1,], type="h", ylab="PC 1")

# close the genotype file
snpgdsClose(genofile)
```

---

snpgdsPED2GDS *Conversion from PLINK PED to GDS*

Description

Convert a PLINK PED text file to a GDS file.

Usage

```
snpgdsPED2GDS(ped.fn, map.fn, out.gdsfn, family=TRUE, snpfirstdim=FALSE,
    compress.annotation="ZIP_RA.max", compress.geno="", verbose=TRUE)
```

Arguments

ped.fn	the file name of PED file, genotype information
map.fn	the file name of MAP file
out.gdsfn	the output GDS file
family	if TRUE, to include family information in the sample annotation
snpfirstdim	if TRUE, genotypes are stored in the individual-major mode, (i.e, list all SNPs for the first individual, and then list all SNPs for the second individual, etc)
compress.annotation	the compression method for the GDS variables, except "genotype"; optional values are defined in the function add.gdsn
compress.geno	the compression method for "genotype"; optional values are defined in the function add.gdsn
verbose	if TRUE, show information

Details

GDS – Genomic Data Structures, the extended file name used for storing genetic data, and the file format is used in the gdsfmt package.

PED – PLINK PED format.

Value

None.

Author(s)

Xiuwen Zheng

References

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ & Sham PC. 2007. PLINK: a toolset for whole-genome association and population-based linkage analysis. American Journal of Human Genetics, 81.

See Also

snpgdsGDS2PED, snpgdsBED2GDS, snpgdsGDS2BED

Examples

```
# open
genofile <- snpgdsOpen(snpgdsExampleFileName())

snpgdsGDS2PED(genofile, "tmp")

# close
snpgdsClose(genofile)
```

```
# PED ==> GDS
snpgdsPED2GDS("tmp.ped", "tmp.map", "test.gds")


# delete the temporary file
unlink(c("tmp.ped", "tmp.map", "test.gds"), force=TRUE)
```

---

snpgdsSampMissRate *Missing Rate of Samples*

Description

Return the missing fraction for each sample

Usage

```
snpgdsSampMissRate(gdsobj, sample.id=NULL, snp.id=NULL, with.id=FALSE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples will be used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs will be used
with.id	if TRUE, the returned value with sample id

Value

A vector of numeric values.

Author(s)

Xiuwen Zheng

See Also

snpgdsSNPRateFreq

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

RV <- snpgdsSampMissRate(genofile)
summary(RV)

# close the genotype file
snpgdsClose(genofile)
```

snpgdsSelectSNP *SNP selection*

Description

Create a list of candidate SNPs based on specified criteria

Usage

```
snpgdsSelectSNP(gdsobj, sample.id=NULL, snp.id=NULL, autosome.only=TRUE,
    remove.monosnp=TRUE, maf=NaN, missing.rate=NaN, verbose=TRUE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples will be used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs will be used
autosome.only	if TRUE, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
remove.monosnp	if TRUE, remove monomorphic SNPs
maf	to use the SNPs with ">= maf" only; if NaN, no any MAF threshold
missing.rate	to use the SNPs with "<= missing.rate" only; if NaN, no any missing threshold
verbose	if TRUE, show information

Value

Return a list of snp ids.

Author(s)

Xiuwen Zheng

See Also

snpgdsSampMissRate, snpgdsSNPRateFreq, snpgdsLDpruning

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

snpset <- snpgdsSelectSNP(genofile, maf=0.05, missing.rate=0.95)
length(snpset)
# 7502

# close the genotype file
snpgdsClose(genofile)
```

snpgdsSlidingWindow *Sliding window*

Description

Apply a user-defined function with a sliding window.

Usage

```
snpgdsSlidingWindow(gdsobj, sample.id=NULL, snp.id=NULL,
    FUN=NULL, winsize=100000L, shift=10000L, unit=c("basepair", "locus"),
    winstart=NULL, autosome.only=FALSE, remove.monosnp=TRUE, maf=NaN,
    missing.rate=NaN, as.is=c("list", "numeric", "array"),
    with.id=c("snp.id", "snp.id.in.window", "none"), num.thread=1,
    verbose=TRUE, ...)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs are used
FUN	a character or a user-defined function, see details
winsize	the size of sliding window
shift	the amount of shifting the sliding window
unit	"basepair" – winsize and shift are applied with SNP coordinate of basepair; "locus" – winsize and shift are applied according to the SNP order in the GDS file
winstart	NULL – no specific starting position; an integer – a starting position for all chromosomes; or a vector of integer – the starting positions for each chromosome
autosome.only	if TRUE, use autosomal SNPs only; if it is a numeric or character value, keep SNPs according to the specified chromosome
remove.monosnp	if TRUE, remove monomorphic SNPs
maf	to use the SNPs with ">= maf" only; if NaN, no MAF threshold
missing.rate	to use the SNPs with "<= missing.rate" only; if NaN, no missing threshold
as.is	save the value returned from FUN as "list" or "numeric"; "array" is equivalent to "numeric" except some cases, see details
with.id	"snp.id", "snp.id.in.window" or "none"
num.thread	the number of (CPU) cores used; if NA, detect the number of cores automatically
verbose	if TRUE, show information
...	optional arguments to FUN

Details

If FUN="snpgdsFst", two additional arguments "population" and "method" should be specified. "population" and "method" are defined in snpgdsFst. "as.is" could be "list" (returns a list of the values from snpgdsFst), "numeric" (population-average Fst, returns a vector) or "array" (population-average and -specific Fst, returns a '# of pop + 1'-by-'# of windows' matrix, and the first row is population-average Fst).

Value

Return a list

Author(s)

Xiuwen Zheng

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# sliding windows
rv <- snpgdsSlidingWindow(genofile, winsize=500000, shift=100000,
    FUN=function(...) NULL)

# plot
plot(rv$chr1.num, ylab="# of SNPs in the sliding window")

# close the genotype file
snpgdsClose(genofile)
```

---

snpgdsSNPList *Create a SNP list object*

Description

A list object of SNP information including rs, chr, pos, allele and allele frequency.

Usage

```
snpgdsSNPList(gdsobj, sample.id=NULL)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples are used

Value

Return an object of snpgdsSNPListClass including the following components:

snp.id	SNP id
chromosome	SNP chromosome index
position	SNP physical position in basepair
allele	reference / non-ref alleles
afreq	allele frequency

Author(s)

Xiuwen Zheng

See Also

snpgdsSNPListIntersect

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# to get a snp list object
snplist <- snpgdsSNPList(genofile)
head(snplist)

# close the file
snpgdsClose(genofile)
```

---

snpgdsSNPListClass *the class of a SNP list*

Description

the class of a SNP list, and its instance is returned from snpgdsSNPList.

Value

Return an object of "snpgdsSNPListClass" including the following components:

snp.id	SNP id
chromosome	SNP chromosome index
position	SNP physical position in basepair
allele	reference / non-ref alleles
afreq	allele frequency

Author(s)

Xiuwen Zheng

See Also

snpgdsSNPList, snpgdsSNPListIntersect

---

snpgdsSNPListIntersect

Get a common SNP list between/among SNP list objects

Description

Get a common SNP list by comparing their snp id, chromosome, positions and allele frequency if needed.

Usage

```
snpgdsSNPListIntersect(snplist1, snplist2, ..., method=c("position", "exact"),
    na.rm=TRUE, same.strand=FALSE, verbose=TRUE)
```

Arguments

snplist1	the SNP list object snpgdsSNPListClass
snplist2	the SNP list object snpgdsSNPListClass
...	the other SNP list objects
method	"exact": matching by all snp.id, chromosomes, positions and alleles; "position": matching by chromosomes and positions
na.rm	if TRUE, remove mismatched alleles
same.strand	if TRUE, assuming the alleles on the same strand
verbose	if TRUE, show information

Value

Return a list of snpgdsSNPListClass including the following components:

idx1	the indices of common SNPs in the first GDS file
idx2	the indices of common SNPs in the second GDS file
idx...	
idxn	the indices of common SNPs in the n-th GDS file
flag2	an integer vector, flip flag for each common SNP for the second GDS file (assuming a value v): bitwAnd(v, 1): 0 – no flip of allele names, 1 – flip of allele names; bitwAnd(v, 2): 0 – on the same strand, 2 – on the different strands, comparing with the first GDS file; bitwAnd(v, 4): 0 – no strand ambiguity, 4 – ambiguous allele names, determined by allele frequencies; NA – mismatched allele names (there is no NA if na.rm=TRUE)

flag...

flagn flip flag for each common SNP for the n-th GDS file

Author(s)

Xiuwen Zheng

See Also

snpgdsSNPList

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

# to get a snp list object
snplist1 <- snpgdsSNPList(genofile)
snplist2 <- snpgdsSNPList(genofile)

# a common snp list, exactly matching
v <- snpgdsSNPListIntersect(snplist1, snplist2)
names(v)
# "idx1" "idx2"

# a common snp list, matching by position
v <- snpgdsSNPListIntersect(snplist1, snplist2, method="pos")
names(v)
# "idx1"  "idx2"  "flag2"

table(v$flag2, exclude=NULL)


# close the file
snpgdsClose(genofile)
```

---

snpgdsSNPRateFreq *Allele Frequency, Minor Allele Frequency, Missing Rate of SNPs*

Description

Calculate the allele frequency, minor allele frequency and missing rate per SNP.

Usage

```
snpgdsSNPRateFreq(gdsobj, sample.id=NULL, snp.id=NULL, with.id=FALSE,
    with.sample.id=FALSE, with.snp.id=FALSE)
```

Arguments

gdsobj	an object of class SNPGDSFileClass, a SNP GDS file
sample.id	a vector of sample id specifying selected samples; if NULL, all samples will be used
snp.id	a vector of snp id specifying selected SNPs; if NULL, all SNPs will be used
with.id	if TRUE, return both sample and SNP IDs
with.sample.id	if TRUE, return sample IDs
with.snp.id	if TRUE, return SNP IDs

Value

Return a list:

AlleleFreq	allele frequencies
MinorFreq	minor allele frequencies
MissingRate	missing rates
sample.id	sample id, if with.id=TRUE or with.sample.id=TRUE
snp.id	SNP id, if with.id=TRUE or with.snp.id=TRUE

Author(s)

Xiuwen Zheng

See Also

snpgdsSampMissRate

Examples

```
# open an example dataset (HapMap)
genofile <- snpgdsOpen(snpgdsExampleFileName())

RV <- snpgdsSNPRateFreq(genofile, with.snp.id=TRUE)
head(data.frame(RV))

hist(RV$AlleleFreq, breaks=128)
summary(RV$MissingRate)

# close the file
snpgdsClose(genofile)
```

snpgdsSummary *Summary of GDS genotype file*

Description

Print the information stored in the gds object

Usage

```
snpgdsSummary(gds, show=TRUE)
```

Arguments

gds	a GDS file name, or an object of class SNPGDSFileClass
show	if TRUE, show information

Value

Return a list:

sample.id	the IDs of valid samples
snp.id	the IDs of valid SNPs

Author(s)

Xiuwen Zheng

Examples

```
snpgdsSummary(snpgdsExampleFileName())
```

snpgdsTranspose *Transpose genotypic matrix*

Description

Transpose the genotypic matrix if needed.

Usage

```
snpgdsTranspose(gds.fn, snpfirstdim=FALSE, compress=NULL, optimize=TRUE,
    verbose=TRUE)
```

Arguments

gds.fn	the file name of SNP GDS format
snpfirstdim	if TRUE, genotypes are stored in snp-by-sample; if FALSE, sample-by-snp mode; if NA, force to transpose the SNP matrix
compress	the compression mode for SNP genotypes, optional values are defined in the function of add.gdsn; if NULL, to use the compression mode
optimize	if TRUE, call cleanup.gds after transposing
verbose	if TRUE, show information

Value

None.

Author(s)

Xiuwen Zheng

Examples

```
# the file name of SNP GDS
(fn <- snpgdsExampleFileName())

# copy the file
file.copy(fn, "test.gds", overwrite=TRUE)

# summary
snpgdsSummary("test.gds")

# transpose the SNP matrix
snpgdsTranspose("test.gds", snpfirstdim=TRUE)

# summary
snpgdsSummary("test.gds")


# delete the temporary file
unlink("test.gds", force=TRUE)
```

---

snpgdsVCF2GDS *Reformat VCF file(s)*

Description

Reformat Variant Call Format (VCF) file(s)

Usage

```
snpgdsVCF2GDS(vcf.fn, out.fn, method=c("biallelic.only", "copy.num.of.ref"),
    snpfirstdim=FALSE, compress.annotation="LZMA_RA", compress.geno="",
    ref.allele=NULL, ignore.chr.prefix="chr", verbose=TRUE)
```

Arguments

`vcf.fn`	the file name of VCF format, `vcf.fn` can be a vector, see details
`out.fn`	the file name of output GDS
`method`	either "biallelic.only" by default or "copy.num.of.ref", see details
`snpfirstdim`	if TRUE, genotypes are stored in the individual-major mode, (i.e, list all SNPs for the first individual, and then list all SNPs for the second individual, etc)
`compress.annotation`	the compression method for the GDS variables, except "genotype"; optional values are defined in the function `add.gdsn`
`compress.geno`	the compression method for "genotype"; optional values are defined in the function `add.gdsn`
`ref.allele`	`NULL` or a character vector indicating reference allele (like `"A"`, `"G"`, `"T"`, `NA`, `...`) for each site where `NA` to use the original reference allele in the VCF file(s). The length of character vector should be the total number of variants in the VCF file(s).
`ignore.chr.prefix`	a vector of character, indicating the prefix of chromosome which should be ignored, like "chr"; it is not case-sensitive
`verbose`	if `TRUE`, show information

Details

GDS – Genomic Data Structures used for storing genetic array-oriented data, and the file format used in the gdsfmt package.

VCF – The Variant Call Format (VCF), which is a generic format for storing DNA polymorphism data such as SNPs, insertions, deletions and structural variants, together with rich annotations.

If there are more than one file names in `vcf.fn`, `snpgdsVCF2GDS` will merge all dataset together if they all contain the same samples. It is useful to combine genetic/genomic data together if VCF data are divided by chromosomes.

`method = "biallelic.only"`: to exact bi-allelic and polymorhpic SNP data (excluding monomorphic variants); `method = "copy.num.of.ref"`: to extract and store dosage (0, 1, 2) of the reference allele for all variant sites, including bi-allelic SNPs, multi-allelic SNPs, indels and structural variants.

Haploid and triploid calls are allowed in the transfer, the variable `snp.id` stores the original the row index of variants, and the variable `snp.rs.id` stores the rs id.

When `snp.chromosome` in the GDS file is character, SNPRelate treats a chromosome as autosome only if it can be converted to a numeric value (like "1", "22"). It uses "X" and "Y" for non-autosomes instead of numeric codes. However, some software format chromosomes in VCF

files with a prefix "chr". Users should remove that prefix when importing VCF files by setting `ignore.chr.prefix = "chr"`.

The extended GDS format is implemented in the SeqArray package to support the storage of single nucleotide variation (SNV), insertion/deletion polymorphism (indel) and structural variation calls. It is strongly suggested to use SeqArray for large-scale whole-exome and whole-genome sequencing variant data instead of SNPRelate.

Value

Return the file name of GDS format with an absolute path.

Author(s)

Xiuwen Zheng

References

The variant call format and VCFtools. Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, Handsaker RE, Lunter G, Marth GT, Sherry ST, McVean G, Durbin R; 1000 Genomes Project Analysis Group. Bioinformatics. 2011 Aug 1;27(15):2156-8. Epub 2011 Jun 7.

http://corearray.sourceforge.net/

See Also

snpgdsBED2GDS

Examples

```
# the VCF file
vcf.fn <- system.file("extdata", "sequence.vcf", package="SNPRelate")
cat(readLines(vcf.fn), sep="\n")

snpgdsVCF2GDS(vcf.fn, "test1.gds", method="biallelic.only")
snpgdsSummary("test1.gds")

snpgdsVCF2GDS(vcf.fn, "test2.gds", method="biallelic.only", snpfirstdim=TRUE)
snpgdsSummary("test2.gds")

snpgdsVCF2GDS(vcf.fn, "test3.gds", method="copy.num.of.ref", snpfirstdim=TRUE)
snpgdsSummary("test3.gds")

snpgdsVCF2GDS(vcf.fn, "test4.gds", method="copy.num.of.ref")
snpgdsSummary("test4.gds")

snpgdsVCF2GDS(vcf.fn, "test5.gds", method="copy.num.of.ref",
    ref.allele=c("A", "T", "T", "T", "A"))
snpgdsSummary("test5.gds")



# open "test1.gds"
```

```
(genofile <- snpgdsOpen("test1.gds"))

read.gdsn(index.gdsn(genofile, "sample.id"))
read.gdsn(index.gdsn(genofile, "snp.rs.id"))
read.gdsn(index.gdsn(genofile, "genotype"))

# close the file
snpgdsClose(genofile)


# open "test2.gds"
(genofile <- snpgdsOpen("test2.gds"))

read.gdsn(index.gdsn(genofile, "sample.id"))
read.gdsn(index.gdsn(genofile, "snp.rs.id"))
read.gdsn(index.gdsn(genofile, "genotype"))

# close the file
snpgdsClose(genofile)


# open "test3.gds"
(genofile <- snpgdsOpen("test3.gds"))

read.gdsn(index.gdsn(genofile, "sample.id"))
read.gdsn(index.gdsn(genofile, "snp.rs.id"))
read.gdsn(index.gdsn(genofile, "genotype"))

# close the file
snpgdsClose(genofile)


# open "test4.gds"
(genofile <- snpgdsOpen("test4.gds"))

read.gdsn(index.gdsn(genofile, "sample.id"))
read.gdsn(index.gdsn(genofile, "snp.rs.id"))
read.gdsn(index.gdsn(genofile, "snp.allele"))
read.gdsn(index.gdsn(genofile, "genotype"))

# close the file
snpgdsClose(genofile)


# open "test5.gds"
(genofile <- snpgdsOpen("test5.gds"))

read.gdsn(index.gdsn(genofile, "sample.id"))
read.gdsn(index.gdsn(genofile, "snp.rs.id"))
read.gdsn(index.gdsn(genofile, "snp.allele"))
read.gdsn(index.gdsn(genofile, "genotype"))

# close the file
```

```
snpgdsClose(genofile)


# delete the temporary files
unlink(paste("test", 1:5, ".gds", sep=""), force=TRUE)
```

---

snpgdsVCF2GDS_R *Reformat a VCF file (R implementation)*

Description

Reformat a Variant Call Format (VCF) file

Usage

```
snpgdsVCF2GDS_R(vcf.fn, out.fn, nblock=1024,
    method = c("biallelic.only", "copy.num.of.ref"),
    compress.annotation="LZMA_RA", snpfirstdim=FALSE, option = NULL,
    verbose=TRUE)
```

Arguments

vcf.fn	the file name of VCF format, vcf.fn can be a vector, see details
out.fn	the output gds file
nblock	the buffer lines
method	either "biallelic.only" by default or "copy.num.of.ref", see details
compress.annotation	the compression method for the GDS variables, except "genotype"; optional values are defined in the function add.gdsn
snpfirstdim	if TRUE, genotypes are stored in the individual-major mode, (i.e, list all SNPs for the first individual, and then list all SNPs for the second individual, etc)
option	NULL or an object from snpgdsOption, see details
verbose	if TRUE, show information

Details

GDS – Genomic Data Structures used for storing genetic array-oriented data, and the file format used in the gdsfmt package.

VCF – The Variant Call Format (VCF), which is a generic format for storing DNA polymorphism data such as SNPs, insertions, deletions and structural variants, together with rich annotations.

If there are more than one file name in vcf.fn, snpgdsVCF2GDS will merge all dataset together once they all contain the same samples. It is useful to combine genetic data if VCF data are divided by chromosomes.

method = "biallelic.only": to exact bi-allelic and polymorhpic SNP data (excluding monomorphic variants); method = "biallelic.only": to exact bi-allelic and polymorhpic SNP data; method

= "copy.num.of.ref": to extract and store dosage (0, 1, 2) of the reference allele for all variant sites, including bi-allelic SNPs, multi-allelic SNPs, indels and structural variants.

Haploid and triploid calls are allowed in the transfer, the variable snp.id stores the original the row index of variants, and the variable snp.rs.id stores the rs id.

The user could use option to specify the range of code for autosomes. For humans there are 22 autosomes (from 1 to 22), but dogs have 38 autosomes. Note that the default settings are used for humans. The user could call option = snpgdsOption(autosome.end=38) for importing the VCF file of dog. It also allows defining new chromosome coding, e.g., option = snpgdsOption(Z=27), then "Z" will be replaced by the number 27.

Value

None.

Author(s)

Xiuwen Zheng

References

The variant call format and VCFtools. Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, Handsaker RE, Lunter G, Marth GT, Sherry ST, McVean G, Durbin R; 1000 Genomes Project Analysis Group. Bioinformatics. 2011 Aug 1;27(15):2156-8. Epub 2011 Jun 7.

See Also

snpgdsVCF2GDS_R, snpgdsOption, snpgdsBED2GDS

Examples

```
# The VCF file
vcf.fn <- system.file("extdata", "sequence.vcf", package="SNPRelate")
cat(readLines(vcf.fn), sep="\n")

snpgdsVCF2GDS_R(vcf.fn, "test1.gds", method="biallelic.only")
snpgdsSummary("test1.gds")

snpgdsVCF2GDS_R(vcf.fn, "test2.gds", method="biallelic.only")
snpgdsSummary("test2.gds")

snpgdsVCF2GDS_R(vcf.fn, "test3.gds", method="copy.num.of.ref")
snpgdsSummary("test3.gds")

snpgdsVCF2GDS_R(vcf.fn, "test4.gds", method="copy.num.of.ref")
snpgdsSummary("test4.gds")
```

Index