Package 'SNPRelate'

April 10, 2015

Type Package

Title Parallel computing toolset for relatedness and principal component analysis of SNP data

Version 1.0.1

Date 2015-01-14

Depends R (>= 2.14), gdsfmt (>= 1.2.2)

LinkingTo gdsfmt

Suggests parallel, RUnit, lattice, BiocStyle, BiocGenerics, knitr

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 format in this package is also used by the GWASTools package with the support of S4 classes and generic functions.

License GPL-3

VignetteBuilder knitr

URL http://github.com/zhengxwen/SNPRelate,
http://corearray.sourceforge.net/tutorials/SNPRelate/

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

biocViews Infrastructure, Genetics, StatisticalMethod, PrincipalComponent

Author Xiuwen Zheng [aut, cre],
Stephanie Gogarten [ctb],
Cathy Laurie [ctb],
Bruce Weir [ctb, ths]

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

R topics documented:

SNPRelate-package . . . 3
hapmap_geno . . . 5
snpgdsAdmixProp . . . 6
snpgdsAlleleSwitch . . . 8
snpgdsApartSelection . . . 9
snpgdsBED2GDS . . . 10
snpgdsClose . . . 12
snpgdsCombineGeno . . . 13
snpgdsCreateGeno . . . 14
snpgdsCreateGenoSet . . . 16
snpgdsCutTree . . . 17
snpgdsDiss . . . 20
snpgdsDrawTree . . . 22
snpgdsEIGMIX . . . 23
snpgdsErrMsg . . . 26
snpgdsExampleFileName . . . 26
SNPGDSFileClass . . . 27
snpgdsFst . . . 28
snpgdsGDS2BED . . . 29
snpgdsGDS2Eigen . . . 31
snpgdsGDS2PED . . . 32
snpgdsGEN2GDS . . . 33
snpgdsGetGeno . . . 35
snpgdsGRM . . . 36
snpgdsHCluster . . . 37
snpgdsIBDKING . . . 39
snpgdsIBDMLE . . . 41
snpgdsIBDMLELogLik . . . 44
snpgdsIBDMoM . . . 46
snpgdsIBDSelection . . . 49
snpgdsIBS . . . 50
snpgdsIBSNum . . . 52
snpgdsIndInb . . . 53
snpgdsIndInbCoef . . . 54
snpgdsLDMat . . . 56
snpgdsLDpair . . . 57
snpgdsLDpruning . . . 59
snpgdsOpen . . . 61
snpgdsOption . . . 62

snpgdsPairIBD 63
snpgdsPairIBDMLELogLik 65
snpgdsPCA 68
snpgdsPCACorr 71
snpgdsPCASampLoading 72
snpgdsPCASNPLoading 74
snpgdsSampMissRate 75
snpgdsSelectSNP 76
snpgdsSlidingWindow 77
snpgdsSNPList 79
snpgdsSNPListClass 80
snpgdsSNPListIntersect 81
snpgdsSNPListStrand 82
snpgdsSNPRateFreq 83
snpgdsSummary 84
snpgdsTranspose 85
snpgdsVCF2GDS 86
snpgdsVCF2GDS_R 89

Index **92**

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

Tutorial: `http://corearray.sourceforge.net/materials/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", package="SNPRelate")
bim.fn <- system.file("extdata", "plinkhapmap.bim", package="SNPRelate")
fam.fn <- system.file("extdata", "plinkhapmap.fam", 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

`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("`
`bound`	if `TRUE`, the estimates are bounded so that no component < 0 or > 1, and the sum of proportions is one

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 `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"
`ibdmat`	the IBD matrix

Author(s)

Xiuwen Zheng

References

Zheng X, Weir BS. Eigenanalysis on SNP Data with an Interpretation of Identity by Descent. 2014. Submitted.

See Also

snpgdsEIGMIX, snpgdsPCA

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$pop),
    xlab="eigenvector 2", ylab="eigenvector 1")
legend("topleft", legend=levels(tab$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)



# run eigen-analysis
RV <- snpgdsEIGMIX(genofile, sample.id=samp.id[pop_code=="JPT"])
z <- RV$ibdmat

mean(c(z))
mean(diag(z))


# 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,
    compress.annotation="ZIP_RA.max", option=NULL, 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
bim.fn	the file name of extended MAP file: two extra cols = allele names
out.gdsfn	the output GDS file
family	if TRUE, to include family information in the sample annotation
compress.annotation	the compression mode of the nodes stored, except "genotype"; optional value are defined in the function of add.gdsn
option	NULL or an object from snpgdsOption, see details
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.

http://corearray.sourceforge.net/

See Also

snpgdsOption, snpgdsBED2GDS, snpgdsGDS2PED

Examples

```
# PLINK BED files
bed.fn <- system.file("extdata", "plinkhapmap.bed", package="SNPRelate")
fam.fn <- system.file("extdata", "plinkhapmap.fam", package="SNPRelate")
bim.fn <- system.file("extdata", "plinkhapmap.bim", 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, sample.id=NULL, snpobj=NULL,
    name.prefix=NULL, snpfirstdim=TRUE, compress.annotation="ZIP.MAX",
    compress.geno="", other.vars=NULL, verbose=TRUE)
```

Arguments

gds.fn	a list of SNP GDS files to be merged
out.fn	the name of output GDS file
sample.id	NULL, or a list. If it is a list, specify sample ids for each SNP GDS file
snpobj	specify a snpgdsSNPListClass object, used for strand switch; if NULL, the strand information of the first SNP GDS file is used
name.prefix	NULL, a character vector (added to sample ids for each GDS file)
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
verbose	if TRUE, show information

Details

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

`snpgdsCreateGeno`, `snpgdsCreateGenoSet`

Examples

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

# combine
snpgdsCombineGeno(c(fn, fn), "test.gds")

snpgdsSummary("test.gds")
```

---

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.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.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    { FStr8 279 ZIP(23.10%) }
# |--+ snp.id   { Int32 9088 ZIP(34.76%) }
# |--+ snp.rs.id    { FStr8 9088 ZIP(42.66%) }
```

```
# |--+ snp.position { Int32 9088 ZIP(94.73%) }
# |--+ snp.chromosome   { UInt8 9088 ZIP(0.94%) } *
# |--+ snp.allele   { FStr8 9088 ZIP(14.45%) }
# |--+ genotype { Bit2 9088x279 } *
# |--+ sample.annot [ data.frame ] *
# |  |--+ sample.id { FStr8 279 ZIP(23.10%) }
# |  |--+ family.id { FStr8 279 ZIP(28.37%) }
# |  |--+ geneva.id { Int32 279 ZIP(80.29%) }
# |  |--+ father.id { FStr8 279 ZIP(12.98%) }
# |  |--+ mother.id { FStr8 279 ZIP(12.86%) }
# |  |--+ plate.id  { FStr8 279 ZIP(1.29%) }
# |  |--+ sex   { FStr8 279 ZIP(28.32%) }
# |  |--+ pop.group { FStr8 279 ZIP(7.89%) }

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    { VStr8 279 ZIP(29.89%) }
# |--+ snp.id   { Int32 6547 ZIP(34.89%) }
# |--+ snp.rs.id    { VStr8 6547 ZIP(40.52%) }
# |--+ snp.position { Int32 6547 ZIP(94.85%) }
# |--+ snp.chromosome   { Int32 6547 ZIP(0.41%) }
# |--+ snp.allele   { VStr8 6547 ZIP(11.51%) }
# |--+ genotype { Bit2 6547x279 } *

# 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`.

The details will be described in future.

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.

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)
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"))
```

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=32, need.ibdmat=FALSE, ibdmat.only=FALSE,
    method=c("Coancestry", "GRM"), 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
eigen.cnt	output the number of eigenvectors; if eigen.cnt <= 0, then return all eigenvectors
need.ibdmat	if TRUE, return the IBD matrix
ibdmat.only	return the IBD matrix only, do not compute the eigenvalues and eigenvectors
method	"Coancestry" – EIGMIX coancestry matrix (by default); "GRM" – genetic relationship matrix
verbose	if TRUE, show information

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"
ibdmat	the IBD matrix

Author(s)

Xiuwen Zheng

References

Zheng X, Weir BS. Eigenanalysis on SNP Data with an Interpretation of Identity by Descent. 2014. Submitted.

See Also

snpgdsEIGMIX, snpgdsPCA

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$pop),
    xlab="eigenvector 2", ylab="eigenvector 1")
legend("topleft", legend=levels(tab$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)
```

```
# run eigen-analysis
RV <- snpgdsEIGMIX(genofile, sample.id=samp.id[pop_code=="JPT"])
z <- RV$ibdmat

mean(c(z))
mean(diag(z))


# 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 *Relatedness estimation – Fst*

Description

Calculate relatedness measures (like Fst) for specified populations

Usage

```
snpgdsFst(gdsobj, population, method=c("W&B02", "W&C84"), 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&B02" – relative beta estimator in Weir&Hill 2002, "W&C84" – Fst estimator in Weir&Cockerham 1984
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.

Value

Return a list:

sample.id	the sample ids used in the analysis
snp.id	the SNP ids used in the analysis
Fst	Fst estimator
Beta	Beta matrix

Author(s)

Xiuwen Zheng

References

Weir, B. S. & Cockerham, C. C. Estimating F-statistics for the analysis of population structure. (1984).

Weir, B. S. & Hill, W. G. Estimating F-statistics. Annual review of genetics 36, 721-50 (2002).

Examples

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

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

# Fst estimation
snpgdsFst(genofile, population=group, method="W&B02")

# or
snpgdsFst(genofile, population=group, method="W&C84")


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

`snpgdsGDS2BED` *Conversion from GDS to PLINK BED*

Description

Convert a GDS file to a PLINK binary ped 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
`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)
```

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)
```

---

snpgdsGDS2PED *Conversion from GDS to PED*

Description

Convert a GDS file to a PLINK 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())

snpset <- snpgdsSelectSNP(genofile, missing.rate=0.95)
snpgdsGDS2PED(genofile, ped.fn="tmp", snp.id=snpset)

# close the genotype 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 mode of the nodes stored, except "genotype"; optional value are defined in the function of `add.gdsn`

`compress.geno` the data compression mode for the variable "genotype", optional values are defined in the function of `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

http://www.stats.ox.ac.uk/~marchini/software/gwas/file_format.html

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=NULL,
    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); if `NULL`, determine automatically
`verbose`	if TRUE, show information

Value

Return an integer matrix with values 0, 1, 2 or NA representing the number of reference allele.

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)
```

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

---

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

Description

Eigen-analysis on the Genetic Relationship Matrix (GRM) for SNP genotype data.

Usage

```
snpgdsGRM(gdsobj, sample.id=NULL, snp.id=NULL, autosome.only=TRUE,
    remove.monosnp=TRUE, maf=NaN, missing.rate=NaN, 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
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
with.id	if TRUE, the returned value with sample.id and sample.id
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
GRM	the genetic relationship matrix

If with.id = FALSE, return the genetic relationship matrix without sample and SNP IDs.

Author(s)

Xiuwen Zheng

References

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).

See Also

snpgdsIndInb, snpgdsFst

Examples

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

# Genetic relationship matrix
grm <- snpgdsGRM(genofile)

# close the file
snpgdsClose(genofile)
```

---

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 = "complete"`.

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)
```

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=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
type	"KING-robust" – relationship inference in the presence of population stratification; "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
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	IBD coefficient, the probability of sharing zero IBD
k1	IBD coefficient, the probability of sharing one IBD
IBS0	proportion of SNPs with zero IBS
kinship	the estimated kinship coefficients, if the parameter kinship=TRUE

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, family.id=NULL)
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)")
```

```
####  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)")


# 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"`
`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)
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)
```

`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=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
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
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 = 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.

The values of the IBS matrix range from ZERO to ONE.

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 = 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.

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"), 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.

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())

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)
```

snpgdsLDMat *Linkage Disequilibrium (LD) analysis*

Description

Return a LD matrix for SNP pairs.

Usage

```
snpgdsLDMat(gdsobj, sample.id=NULL, snp.id=NULL, slide=250,
    method=c("composite", "r", "dprime", "corr"), 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
slide	# of SNPs, the size of sliding window, see details
method	"composite", "r", "dprime", "corr", see details
num.thread	the number of (CPU) cores used; if NA, detect the number of cores automatically
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, output a n-by-n LD matrix; otherwise, output a m-by-n LD matrix, where n is the total number of SNPs, m is the size of sliding window. For n-by-n matrix, the value of i row and j column is LD of i and j SNPs. For m-by-n matrix, 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

```
library(lattice)

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

# chromosome 15
snpset <- read.gdsn(index.gdsn(genofile, "snp.id"))[
    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
levelplot(t(ld.noslide$LD^2), col.regions = terrain.colors)

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


# 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 = NaN, missing.rate = NaN,
    method = c("composite", "r", "dprime", "corr"), slide.max.bp = 500000,
    slide.max.n = NA, ld.threshold = 0.2, 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
`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
`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`.

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, 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)
names(snpset)
#  [1] "chr1"  "chr2"  "chr3"  "chr4"  "chr5"  "chr6"  "chr7"  "chr8"  "chr9"
# [10] "chr10" "chr11" "chr12" "chr13" "chr14" "chr15" "chr16" "chr17" "chr18"
# ......
head(snpset$chr1)
# [1] 1 2 3 4 5 6

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

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

---

snpgdsOpen *Open a SNP GDS File*

Description

Open a SNP GDS file

Usage

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

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"), kinship.constraint=FALSE,
    max.niter=1000, 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", or "MoM", 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.

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)


# 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)
```

`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)
```

---

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, eigen.cnt=32,
    num.thread=1L, bayesian=FALSE, need.genmat=FALSE, genmat.only=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
eigen.cnt	output the number of eigenvectors; if eigen.cnt <= 0, then return all eigenvectors
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
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 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 PC
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 (2006) Population structure and eigenanalysis. PLoS Genetics 2:e190.

See Also

snpgdsPCACorr, snpgdsPCASampLoading, snpgdsPCASNPLoading

Examples

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

# run PCA
RV <- snpgdsPCA(genofile)

# eigenvalues
RV$eigenval

# variance proportion (%)
```

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


####  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$pop),
    xlab="eigenvector 2", ylab="eigenvector 1")
legend("topleft", legend=levels(tab$pop), pch="o", col=1:4)


# close the file
snpgdsClose(genofile)
```

`snpgdsPCACorr` *PCA-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,
    verbose=TRUE)
```

Arguments

`pcaobj`	the `snpgdsPCAClass` object returned from the function snpgdsPCA
`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
`verbose`	if TRUE, show information

Value

Return a list:

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

# close the file
snpgdsClose(genofile)
```

---

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	the snpgdsPCASNPLoadingClass 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

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"
TraceXTX	the trace of the genetic covariance matrix
Bayesian	whether use bayerisan normalization

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"))

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

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

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

# 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	the snpgdsPCAClass object returned from the function snpgdsPCA
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

Return a snpgdsPCASNPLoading object, 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	the SNP loadings, or SNP eigenvectors
TraceXTX	the trace of the genetic covariance matrix
Bayesian	whether use bayerisan normalization
avefreq	the allele frequency used in snpgdsPCA
scale	internal parameter

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, 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"   "avefreq"    "scale"
dim(SnpLoad$snploading)
# [1]     8 8722

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

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

---

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"),
    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	the user-defined function
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 double
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

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:

rs.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, snpgdsSNPListStrand

Examples

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

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

# 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:

`rs.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 two SNP list objects*

Description

Get a common SNP list by comparing their rs id, chromosome indices and positions.

Usage

```
snpgdsSNPListIntersect(snplist1, snplist2)
```

Arguments

snplist1	the first SNP list object snpgdsSNPListClass
snplist2	the second SNP list object snpgdsSNPListClass

Value

Return an object of snpgdsSNPListClass including the following components:

rs.id	SNP id
chromosome	SNP chromosome index
position	SNP physical position in basepair
allele	reference / non-ref alleles from the first SNP list object
afreq	allele frequency from the first SNP list object

Author(s)

Xiuwen Zheng

See Also

snpgdsSNPList, snpgdsSNPListStrand

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
snplist <- snpgdsSNPListIntersect(snplist1, snplist2)
```

```
summary(snplist$afreq)

# close the file
snpgdsClose(genofile)
```

---

snpgdsSNPListStrand *Switch allele strand.*

Description

To get a logical vector, indicating whether allele references of snplist2 need to be switched, with respect to snplist1.

Usage

```
snpgdsSNPListStrand(snplist1, snplist2, same.strand=FALSE)
```

Arguments

snplist1	the first SNP list object "snpgdsSNPListClass"
snplist2	the second SNP list object "snpgdsSNPListClass"
same.strand	TRUE assuming alleles are on the same strand (e.g., forward strand); otherwise, FALSE not assuming whether on the same strand or not

Value

a logical vector, where TRUE indiciates the allele references need to be switched, and NA indiciates that locus is not in the common snp list.

Author(s)

Xiuwen Zheng

See Also

snpgdsSNPList, snpgdsSNPListStrand

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
L <- snpgdsSNPListStrand(snplist1, snplist2)
```

```
table(L, 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)
```

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 sample and SNP IDs

Value

Return a list:

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

Author(s)

Xiuwen Zheng

See Also

snpgdsSampMissRate

Examples

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

RV <- snpgdsSNPRateFreq(genofile)
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, 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")

# 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="ZIP_RA.max", 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 data compression mode, except the GDS variable "genotype"; optional values are defined in the function of add.gdsn
compress.geno	the data compression mode for the variable "genotype", optional values are defined in the function of 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 name in `vcf.fn`, `snpgdsVCF2GDS` will merge all dataset together once 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"`.

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="ZIP_RA.max", 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 flag of the nodes stored, except "genotype"; the string value is defined in the function of 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

∗Topic **GDS**
snpgdsAdmixProp, 6
snpgdsAlleleSwitch, 8
snpgdsApartSelection, 9
snpgdsBED2GDS, 10
snpgdsClose, 12
snpgdsCombineGeno, 13
snpgdsCreateGeno, 14
snpgdsCreateGenoSet, 16
snpgdsCutTree, 17
snpgdsDiss, 20
snpgdsDrawTree, 22
snpgdsEIGMIX, 23
snpgdsErrMsg, 26
snpgdsExampleFileName, 26
SNPGDSFileClass, 27
snpgdsFst, 28
snpgdsGDS2BED, 29
snpgdsGDS2Eigen, 31
snpgdsGDS2PED, 32
snpgdsGEN2GDS, 33
snpgdsGetGeno, 35
snpgdsGRM, 36
snpgdsHCluster, 37
snpgdsIBDKING, 39
snpgdsIBDMLE, 41
snpgdsIBDMLELogLik, 44
snpgdsIBDMoM, 46
snpgdsIBDSelection, 49
snpgdsIBS, 50
snpgdsIBSNum, 52
snpgdsIndInb, 53
snpgdsIndInbCoef, 54
snpgdsLDMat, 56
snpgdsLDpair, 57
snpgdsLDpruning, 59
snpgdsOpen, 61
snpgdsOption, 62
snpgdsPairIBD, 63
snpgdsPairIBDMLELogLik, 66
snpgdsPCA, 68
snpgdsPCACorr, 71
snpgdsPCASampLoading, 72
snpgdsPCASNPLoading, 74
snpgdsSampMissRate, 75
snpgdsSelectSNP, 76
snpgdsSlidingWindow, 77
snpgdsSNPList, 79
snpgdsSNPListClass, 80
snpgdsSNPListIntersect, 81
snpgdsSNPListStrand, 82
snpgdsSNPRateFreq, 83
snpgdsSummary, 84
snpgdsTranspose, 85
snpgdsVCF2GDS, 86
SNPRelate-package, 3
∗Topic **GWAS**
snpgdsAdmixProp, 6
snpgdsAlleleSwitch, 8
snpgdsApartSelection, 9
snpgdsBED2GDS, 10
snpgdsClose, 12
snpgdsCombineGeno, 13
snpgdsCreateGeno, 14
snpgdsCreateGenoSet, 16
snpgdsCutTree, 17
snpgdsDiss, 20
snpgdsDrawTree, 22
snpgdsEIGMIX, 23
snpgdsErrMsg, 26
snpgdsExampleFileName, 26
SNPGDSFileClass, 27
snpgdsFst, 28
snpgdsGDS2BED, 29
snpgdsGDS2Eigen, 31
snpgdsGDS2PED, 32
snpgdsGEN2GDS, 33
snpgdsGetGeno, 35

snpgdsGRM, 36
snpgdsHCluster, 37
snpgdsIBDKING, 39
snpgdsIBDMLE, 41
snpgdsIBDMLELogLik, 44
snpgdsIBDMoM, 46
snpgdsIBDSelection, 49
snpgdsIBS, 50
snpgdsIBSNum, 52
snpgdsIndInb, 53
snpgdsIndInbCoef, 54
snpgdsLDMat, 56
snpgdsLDpair, 57
snpgdsLDpruning, 59
snpgdsOpen, 61
snpgdsOption, 62
snpgdsPairIBD, 63
snpgdsPairIBDMLELogLik, 66
snpgdsPCA, 68
snpgdsPCACorr, 71
snpgdsPCASampLoading, 72
snpgdsPCASNPLoading, 74
snpgdsSampMissRate, 75
snpgdsSelectSNP, 76
snpgdsSlidingWindow, 77
snpgdsSNPList, 79
snpgdsSNPListClass, 80
snpgdsSNPListIntersect, 81
snpgdsSNPListStrand, 82
snpgdsSNPRateFreq, 83
snpgdsSummary, 84
snpgdsTranspose, 85
snpgdsVCF2GDS, 86
snpgdsVCF2GDS_R, 89
SNPRelate-package, 3

∗Topic **PCA**
snpgdsPCA, 68
snpgdsPCACorr, 71
snpgdsPCASampLoading, 72
snpgdsPCASNPLoading, 74

∗Topic **datasets**
hapmap_geno, 5

∗Topic **gds**
snpgdsVCF2GDS_R, 89

add.gdsn, *89*

cleanup.gds, *85*
closefn.gds, *12*

gds.class, *27*, *32*
gdsfmt, *11*, *30–32*, *34*, *86*, *90*

hapmap_geno, 5
hclust, *37, 38*

openfn.gds, *61*

snpgdsAdmixProp, 6
snpgdsAlleleSwitch, 8
snpgdsApartSelection, 9
snpgdsBED2GDS, 10, *12*, *30*, *34*, *87*, *90*
snpgdsClose, 12, *27*, *62*
snpgdsCombineGeno, 13, *15, 16*
snpgdsCreateGeno, 14, *14*, *16*
snpgdsCreateGenoSet, *14, 15*, 16
snpgdsCutTree, 17, *22, 23*, *38*
snpgdsDiss, *19*, 20, *37, 38*
snpgdsDrawTree, *19*, 22
snpgdsEIGMIX, *6, 7*, 23, *25*
snpgdsErrMsg, 26
snpgdsExampleFileName, 26
SNPGDSFileClass, *8*, *12*, *20*, *24*, 27, *28, 29*, *31*, *35, 36*, *39*, *42*, *45*, *47*, *50*, *52, 53*, *56*, *59*, *61, 62*, *68*, *71, 72*, *74*, *76–79*, *83, 84*
SNPGDSFileClass-class *(*SNPGDSFileClass*)*, 27
snpgdsFst, 28, *37*
snpgdsGDS2BED, 29, *33*
snpgdsGDS2Eigen, 31
snpgdsGDS2PED, *12*, *30, 31*, 32
snpgdsGEN2GDS, 33
snpgdsGetGeno, 35
snpgdsGRM, 36
snpgdsHCluster, *18, 19*, *21*, 37
snpgdsIBDKING, 39, *50*
snpgdsIBDMLE, *40*, 41, *45*, *48–50*, *64*, *67*
snpgdsIBDMLELogLik, *43*, 44, *48*, *64*, *67*
snpgdsIBDMoM, *40*, *43*, *45*, 46, *49, 50*, *64*, *67*
snpgdsIBDSelection, 49
snpgdsIBS, *19*, *37, 38*, 50, *53*
snpgdsIBSNum, *51*, 52
snpgdsIndInb, *37*, 53
snpgdsIndInbCoef, 54
snpgdsLDMat, 56, *58*, *60*
snpgdsLDpair, 57, *57*, *60*
snpgdsLDpruning, *10*, *57, 58*, 59, *77*
snpgdsOpen, *13*, *27*, 61

snpgdsOption, *11, 12*, 62, *89, 90*
snpgdsPairIBD, 63, *67*
snpgdsPairIBDMLELogLik, *64*, 65
snpgdsPCA, *6, 7*, *25*, 68, *71–75*
snpgdsPCACorr, *69*, 71, *73*, *75*
snpgdsPCASampLoading, *69*, 72, *72*, *75*
snpgdsPCASNPLoading, *69*, *72, 73*, 74
snpgdsSampMissRate, 75, *77*, *83*
snpgdsSelectSNP, 76
snpgdsSlidingWindow, 77
snpgdsSNPList, 79, *80–82*
snpgdsSNPListClass, *13*, 80, *81*
snpgdsSNPListIntersect, *79, 80*, 81
snpgdsSNPListStrand, *79*, *81*, 82, *82*
snpgdsSNPRateFreq, *76, 77*, 83
snpgdsSummary, 84
snpgdsTranspose, 85
snpgdsVCF2GDS, *34*, 86
snpgdsVCF2GDS_R, 89, *90*
SNPRelate *(*SNPRelate-package*)*, 3
SNPRelate-package, 3