Package 'snpStats'

November 12, 2026

Title SnpMatrix and XSnpMatrix classes and methods

Version 1.62.0

Date 2025-04-28

Author David Clayton <dc208@cam.ac.uk>

Description Classes and statistical methods for large SNP association studies. This extends the earlier snpMatrix package, allowing for uncertainty in genotypes.

Maintainer David Clayton <dc208@cam.ac.uk>

Depends R(>= 2.10.0), survival, Matrix, methods

Imports graphics, grDevices, stats, utils, BiocGenerics

Suggests hexbin

License GPL-3

Collate ss.R contingency.table.R convert.R compare.R glm-test.R imputation.R indata.R long.R misc.R ld.R mvtests.R pedfile.R outdata.R plink.R qc.R qq-chisq.R single.R tdt-single.R structure.R xstuff.R

LazyLoad yes

biocViews Microarray, SNP, GeneticVariability

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

git_branch RELEASE_3_23

git_last_commit 1919258

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

snpStats-package 3
chi.squared 3
convert.snpMatrix 5
example-new 5

families . . . 6
filter.rules . . . 7
for.exercise . . . 8
Fst . . . 9
glm.test.control . . . 10
GlmEstimates-class . . . 11
GlmTests-class . . . 12
ibsCount . . . 13
ibsDist . . . 14
imputation.maf . . . 15
ImputationRules-class . . . 16
impute.snps . . . 17
ld . . . 18
ld.example . . . 19
mean2g . . . 20
misinherits . . . 21
mvtests . . . 22
plotUncertainty . . . 23
pool . . . 24
pool2 . . . 25
pp . . . 26
qq.chisq . . . 26
random.snps . . . 28
read.beagle . . . 29
read.impute . . . 30
read.long . . . 31
read.mach . . . 33
read.pedfile . . . 34
read.plink . . . 35
read.snps.long . . . 37
row.summary . . . 39
sample.ped.gz . . . 41
single.snp.tests . . . 42
SingleSnpTests-class . . . 44
sm.compare . . . 45
snp.cor . . . 46
snp.imputation . . . 48
snp.lhs.estimates . . . 50
snp.lhs.tests . . . 52
snp.pre.multiply . . . 54
snp.rhs.estimates . . . 55
snp.rhs.tests . . . 57
SnpMatrix-class . . . 59
switch.alleles . . . 61
tdt.snp . . . 62
test.allele.switch . . . 64
testdata . . . 66
write.plink . . . 67

write.SnpMatrix 69
XSnpMatrix-class 70
xxt 71

Index **74**

`snpStats-package` *SnpMatrix and XSnpMatrix classes and methods*

Description

Classes and statistical methods for large SNP association studies. This extends the earlier snpMatrix package, allowing for uncertainty in genotypes.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

Author(s)

David Clayton <dc208@cam.ac.uk>

Maintainer: David Clayton <dc208@cam.ac.uk>

`chi.squared` *Extract test statistics and p-values*

Description

Generic functions to extract values from the SNP association test objects returned by various testing functions

Usage

```
chi.squared(x, df)
deg.freedom(x)
effect.sign(x, simplify)
p.value(x, df)
sample.size(x)
effective.sample.size(x)
```

Arguments

x	An object of class "SingleSnpTests", "SingleSnpTestsScore", or "GlmTests"
df	Either the numeric value 1 or 2 (not used when x is of class "GlmTests")
simplify	This switch is relevant when x is of class "GlmTests" and plays the same role as it does in sapply. If simplify=TRUE, where possible the output is returned as a simple numeric vector rather than as a list

Details

These functions operate on objects created by single.snp.tests, snp.lhs.tests, and snp.lhs.tests.

The functions chi.squared and p.value return the chi-squared statistic and the corresponding *p*-value. The argument df is only used for output from single.snp.tests, since this function calculates both 1 df and 2 df tests for each SNP. The functions snp.lhs.tests and snp.rhs.tests potentially calculate chi-squared tests on varying degrees of freedom, which can be extracted with deg.freedom. The function effect.sign indicates the direction of associations. When applied to an output object from snp.single.tests, it returns +1 if the association, as measured by the 1 df test, is positive and -1 if the association is negative. Each test calculated by GlmTests are potentially tests of several parameters so that the effect sign can be a vector. Thus effect.sign returns a list of sign vectors unless, if simplify=TRUE, and it can be simplified as a single vector with one sign for each test. The function sample.size returns the number of observations actually used in the test, after exclusions due to missing data have been applied, and effective.sample.size returns the effective sample size which is less than the true sample size for tests on imperfectly imputed SNPs.

Value

A numeric vector containing the chi-squared test statistics or p-values. The output vector has a names attribute.

Note

The df and simplify arguments are not always required (or legal). See above

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

single.snp.tests, snp.lhs.tests, snp.rhs.tests, SingleSnpTests-class, SingleSnpTestsScore-class, GlmTests-class

Examples

```
data(testdata)
tests <- single.snp.tests(cc, stratum=region, data=subject.data,
   snp.data=Autosomes, snp.subset=1:10)
chi.squared(tests, 1)
p.value(tests, 1)
```

convert.snpMatrix *Convert* snpMatrix *objects to* snpStats *objects*

Description

These functions convert snpMatrix objects to snpStats objects. convert.snpMatrix converts a single object, while convert.snpMatrix.dir converts all stored elements in a specified directory. They really only change the class names since most of the classes in snpStats are backwards-compatible with snpMatrix. The exception is the ImputationRules class; imputation.rules objects will need to be regenerated.

Usage

```
convert.snpMatrix(object)

convert.snpMatrix.dir(dir = ".", ext = "RData")
```

Arguments

object	Object to be converted
dir	A directory containing saved snpMatrix objects
ext	The file extension for files containing such objects

Value

convert.snpMatrix returns the converted object. convert.snpMatrix.dir rewrites the files in place.

Author(s)

David Clayton <dc208@cam.ac.uk>

example-new *An example of intensity data for SNP genotyping*

Description

The file example-new.txt contains some signal intensity data for testing and comparing genotype scoring algorithms

Format

This is a text file containing data on 99 SNPs for 1550 DNA samples. One line of data appears for each SNP, starting with the SNP name and followed by 1550 pairs of intensity values. There is a header line containing variable names, with intensities labelled as xxxxA and xxxxB, where xxxx is the sample name.

Details

See the package vignette "Comparing clustering algorithms".

Source

These data were originally distributed with the "Illuminus" genotype scoring software from the Wellcome Trust Sanger Institute: http://www.sanger.ac.uk/resources/software/illuminus/

`families` *Test data for family association tests*

Description

These data started life as real data derived from an affected sibling pair study of type 1 diabetes. However, original subject and SNP identidiers have been replaced by randomly chosen ones.

Usage

```
data(families)
```

Format

There are two objects in the loaded data file:

- `genotypes`: An object of class `"snp.matrix"` containing the SNP genotype data for both parents and affected offspring
- `pedData`: A data frame containing the standard six fields for a *LINKAGE* pedfile. The are named `familyid`, `member`, `father`, `mother` `sex`, and `affected`

The two objects are linked by common row names.

Details

Coding in the `pedData` frame is as in the *LINKAGE* package, except that missing data are coded `NA` rather than zero

Examples

```
data(families)
summary(genotypes)
summary(pedData)
```

`filter.rules` *Filter a set of imputation rules*

Description

Determine which imputation rules are broken by removal of some SNPs from a study. This function is needed because, when if it emerges that genotyping of some SNPs is not reliable, necessitating their removal from study, we would also wish to remove any SNPs imputed on the basis of these unreliable SNPs.

Usage

```
filter.rules(rules, snps.excluded, exclusions = TRUE)
```

Arguments

`rules`	An object of class `"ImputationRules"` containing a set of imputation rules
`snps.excluded`	The names of the SNPs whose removal is to be investigated
`exclusions`	If `TRUE`, the names of the imputed SNPs which would be lost by removal of the SNPs listed in `snps.excluded`. If `FALSE`, the names of the imputed SNPs which would *not* be lost are returned

Value

A character vector containing the names of imputed SNPs to be removed

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

`ImputationRules-class`, `snp.imputation`

Examples

```
# No example yet
```

`for.exercise` *Data for exercise in use of the snpStats package*

Description

These data have been created artificially from publicly available datasets. The SNPs have been selected from those genotyped by the International HapMap Project (http://www.hapmap.org) to represent the typical density found on a whole genome association chip, (the Affymetrix 500K platform, http://www.affymetrix.com/support/technical/sample_data/500k_hapmap_genotype_data.affx for a moderately sized chromosome (chromosome 10). A study of 500 cases and 500 controls has been simulated allowing for recombination using beta software from Su and Marchini (http://www.stats.ox.ac.uk/~marchini/software/gwas/hapgen.html). Re-sampling of cases was weighted in such a way as to simulate three "causal" locus on this chromosome, with multiplicative effects of 1.3, 1.4 and 1.5 for each copy of the risk allele.

Usage

```
data(for.exercise)
```

Format

There are three data objects in the dataset:

- `snps.10`: An object of class "`SnpMatrix`" containing a matrix of SNP genotype calls. Rows of the matrix correspond to subjects and columns correspond to SNPs.
- `snp.support`: A conventional R data frame containing information about the SNPs typed (the chromosome position and the nucleotides corresponding to the two alleles of the SNP).
- `subject.support`: A conventional R dataframe containing information about the study subjects. There are two variables; `cc` gives case/control status (1=case), and `stratum` gives ethnicity.

Source

The data were obtained from the diabetes and inflammation laboratory (see http://www-gene.cimr.cam.ac.uk)

Examples

```
data(for.exercise)
snps.10
summary(snps.10)
summary(snp.support)
summary(subject.support)
```

`Fst` *Calculate fixation indices*

Description

This function calculates the fixation index Fst for each SNP, together with its weight in the overall estimate (as used by the Internation HapMap Consortium).

Usage

```
Fst(snps, group, pairwise=FALSE)
```

Arguments

`snps`	an object of class `SnpMatrix` or `XSnpMatrix` containing the SNP data
`group`	a factor (or object than can be coerced into a factor), of length equal to the number of rows of `snps`, giving the grouping or rows for which the Fst is to be calculated
`pairwise`	if `TRUE`, the within-group variances are weighted according to the number of possible within-group pairwise comparisons of chromosomes. If `FALSE`, the default value, weights are simply the number of chromosomes in each group.

Details

See vignette.

Value

A list:

Fst Fst values for each SNP

weight The weights for combining these into a single index

Note

Uncertain genotypes are treated as missing

Author(s)

David Clayton <dc208@cam.ac.uk>

Examples

```
## Analysis of some HapMap data

data(for.exercise)
f <- Fst(snps.10, subject.support$stratum)
weighted.mean(f$Fst, f$weight)
```

`glm.test.control` *Set up control object for GLM computations*

Description

Several commands depend on fitting a generalized linear model (GLM), using the standard iteratively reweighted least squares (IRLS) algorithm. This function sets various control parameters for this.

Usage

```
glm.test.control(maxit = 20, epsilon = 1.e-5, R2Max = 0.99)
```

Arguments

`maxit`	Maximum number of IRLS steps
`epsilon`	Convergence threshold for IRLS algorithm
`R2Max`	R-squared limit for aliasing of new terms

Details

Sometimes (although not always), an iterative scheme is necessary to fit a generalized linear model (GLM). The `maxit` parameter sets the maximum number of iterations to be carried out, while the `epsilon` parameter sets the criterion for determining convergence. Variables which are judged to be "aliased" are dropped. A variable is judged to be aliased if RSS/TSS is less than (1-R2Max), where

- RSS is the residual (weighted) sum of squares from the regression of that variable on the variables which precede it in the model formula (and any stratification defined in a strata() call in th emodel formula), and
- TSS is the total (weighted) sum of squared deviations of this variable from its mean (or, when a strata() call is present, from its stratum-specific means).

The weights used in this calculation are the "working" weights of the IRLS algorithm.

Value

Returns the parameters as a list in the expected order

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

`snp.lhs.tests`, `snp.rhs.tests`

GlmEstimates-class *Class "GlmEstimates"*

Description

A simple class to hold output from snp.lhs.estimates and snp.rhs.estimates. Its main purpose is to provide a show method

Objects from the Class

Objects from this class are simple lists. Each element of the list is a list giving the results of a generalized linear model fit, with elements:

Y.var Name of the Y variable

beta The vector or parameter estimates (with their names)

Var.beta The upper triangle of the variance-covariance matrix of estimates, stored as a simple vector

N The number of "units" used in the model fit

Extends

Class "list", from data part. Class "vector", by class "list", distance 2.

Methods

[signature(x = "GlmEstimates", i = "ANY", j = "missing", drop = "missing"): Subset

coerce signature(from = "GlmEstimates", to = "GlmTests"): Calculate Wald tests

show signature(object = "GlmEstimates"): Display

Note

Wald tests calculated by coercing an object of class GlmEstimates to class GlmTests are *asymptotically* equivalent to the "score" tests calculated by snp.lhs.tests and snp.rhs.tests, and both types of test are *asymptotically* correct. But the asymptotic properties of Wald tests are not as good as those of score tests and, in circumstances such as low sample size or low minor allele frequency, Wald and score tests may differ substantially. In general score tests are to be preferred, but Wald tests are provided in snpStats because they are widely used.

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

snp.lhs.estimates, snp.rhs.estimates

Examples

```
showClass("GlmEstimates")
```

`GlmTests-class` *Classes "GlmTests" and "GlmTestsScore"*

Description

Classes of objects created by `snp.lhs.tests` and `snp.rhs.tests`. The class `"GlmTestsScore"` extends the class `"GlmTests"` and is invoked by setting the argument `score=TRUE` when calling testing functions in order to save the scores and their variances (and covariances)

Objects from the Class

Objects of class `"GlmTests"` have four slots:

snp.names When only single SNPs are tested, a character vector of SNP names. Otherwise a list of such vectors (one for each test)

var.names A character vector containing names of variables tested against SNPs

chisq A numerical vector of chi-squared test values

df An integer vector of degrees of freedom for the tests

N A integer vector of the number of samples contributing to each test

The `"GlmTestsScore"` class extends this, adding a slot `score` containing a list with elements which are themselves lists with two elements:

U The vector (or matrix) of efficient scores

V The upper triangle of the variance-covariance matrix of `U`, stored as a vector

Methods

[ ] `signature(x = "GlmTests", i = "ANY", j = "missing", drop = "missing")`: Subsetting operator

coerce `signature(from = "GlmTests", to = "data.frame")`: Simplify object

chi.squared `signature(x = "GlmTests", df = "missing")`: Extract chi-squared test values

deg.freedom `signature(x = "GlmTests")`: Extract degrees of freedom for tests

names `signature(x="GlmTests")`: Extract (or generate) a name for each test

p.value `signature(x = "GlmTests", df = "missing")`: Extract *p*-values

sample.size `signature(object = "GlmTests")`: Extract sample sizes for tests

show `signature(object = "GlmTests")`: Show method

summary `signature(object = "GlmTests")`: Summary method

[ ] `signature(x = "GlmTestsScore", i = "ANY", j = "missing", drop = "missing")`: Subsetting operator

effect.sign `signature(x = "GlmTestsScore", simplify = "logical")`: Extract signs of associations. If `simpify` is `TRUE` then a simple vector is returned if all tests are on 1df

pool2 `signature(x = "GlmTestsScore", y = "GlmTestsScore", score = "logical")`: Combine results from two sets of tests

switch.alleles `signature(x = "GlmTestsScore", snps = "character")`: Emulate, in the score vector and its (co)variances, the effect of switching of the alleles of specified SNPs

Note

Most of the methods for this class are shared with the `SingleSnpTests` and `SingleSnpTestsScore` classes

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

`snp.lhs.tests`,`snp.rhs.tests`, `SingleSnpTests`, `SingleSnpTestsScore`

Examples

```
showClass("GlmTests")
```

---

ibsCount *Count alleles identical by state*

Description

This function counts, for all pairs of subjects and across all SNPs, the total number of alleles which are identical by state (IBS)

Usage

```
ibsCount(snps, uncertain = FALSE)
```

Arguments

`snps`	An input object of class `"SnpMatrix"` or `"XSnpMatrix"`
`uncertain`	If `FALSE`, uncertain genotypes are ignored. Otherwise contributions are weighted by posterior probabilities

Details

For each pair of subjects the function counts the total number of alleles which are IBS. For autosomal SNPs, each locus contributes 4 comparisons, since each subject carries two copies. For SNPs on the X chromosome, the number of comparisons is also 4 for female:female comparisons, but is 2 for female:male and 1 for male:male comparisons.

Value

If there are N rows in the input matrix, the function returns an $N*N$ matrix. The lower triangle contains the total number of comparisons and the upper triangle contains the number of these which are IBS. The diagonal contains the number of valid calls for each subject.

Note

In genome-wide studies, the SNP data will usually be held as a series of objects (of class `"SnpMatrix"` or `"XSnpMatrix"`), one per chromosome. Note that the matrices produced by applying the `ibsCount` function to each object in turn can be added to yield the genome-wide result.

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

`ibsDist` which calculates a distance matrix based on proportion of alleles which are IBS

Examples

```
data(testdata)
ibs.A <- ibsCount(Autosomes[,1:100])
ibs.X <- ibsCount(Xchromosome)
```

---

ibsDist *Distance matrix based on identity by state (IBS)*

Description

Expresses a matrix of IBS counts (see `ibsCount`) as a distance matrix. The distance between two samples is returned as the proportion of allele comparisons which are *not* IBS.

Usage

```
ibsDist(counts)
```

Arguments

`counts`	A matrix of IBS counts as produced by the function `ibsCount`

Value

An object of class `"dist"` (see `dist`)

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

ibsCount, dist

Examples

```
data(testdata)
ibs <- ibsCount(Xchromosome)
distance <- ibsDist(ibs)
```

`imputation.maf` *Extract statistics from imputation rules*

Description

These functions extract key characteristics of regression-based imputation rules stored as an object of class `"ImputationRules"`. `imputation.maf` extracts the minor allele frequencies of the imputed SNPs and `imputation.r2` extracts the prediction R^2.

Usage

```
can.impute(rules)
imputation.maf(rules)
imputation.r2(rules)
imputation.nsnp(rules)
```

Arguments

`rules`	An object of class `"ImputationRules"`

Details

`can.impute` returns a logical vector identifying which rules allow a valid imputation. `imputation.maf` and `imputation.r2` extract the minor allele frequencies of the imputed SNPs and the R^2 for prediction achieved when building each rule. `imputation.nsnp` returns the numbers of SNPs used in each imputation

Value

Either a logical vector, or a numeric vector containing the extracted values

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

ImputationRules-class, snp.imputation

Examples

```
# These functions are currently defined as
function (rules) sapply(rules, function(x) x$maf)
function (rules) sapply(rules, function(x) x$r2)
```

`ImputationRules-class` *Class "ImputationRules"*

Description

A class defining a list "rules" for imputation of SNPs. Rules are estimated population haplotype probabilities for a target SNP and one or more predictor SNPs

Objects from the Class

Objects are lists of *rules*. Rules are named list elements each describing imputation of a SNP by a linear regression equation. Each element is itself a list with the following elements:

maf The minor allele frequency of the imputed SNP

r.squared The squared Pearson correlation coefficient between observed and predicted SNP duration derivation of the rule.

snps The names of the SNPs to be included in the regression.

hap.probs A numerical array containing estimated probabilities for haplotypes of the SNP to be imputed and all the predictor SNPs

If any target SNP is monomorphic, the corresponding rule is returned as `NULL`. An object of class `ImputationRules` has an attribute, `Max.predictors`, which gives the maximum number of predictors used for any imputation.

Methods

show `signature(object = "ImputationRules")`: prints an abreviated listing of the rules

summary `signature(object = "ImputationRules")`: returns a table which shows the distribution of r-squared values achieved against the number of snps used for imputation

plot `signature(x="ImputationRules", y="missing")`: plots the distribution of r-squared values as a stacked bar chart

[ ] `signature(x = "ImputationRules", i = "ANY")`: subset operations

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

`snp.imputation`, `impute.snps`, `single.snp.tests`

Examples

```
showClass("ImputationRules")
```

impute.snps *Impute snps*

Description

Given SNPs stored in an object of class "SnpMatrix" or "XSnpMatrix" and a set of imputation rules in an object of class "ImputationRules", this function calculates imputed values.

Usage

```
impute.snps(rules, snps, subset = NULL, as.numeric = TRUE)
```

Arguments

rules	The imputation rules; an object of class "ImputationRules"
snps	The object of class "SnpMatrix" or "XSnpMatrix" containing the observed SNPs
subset	A vector describing the subset of subjects to be used. If NULL (default), then use all subjects
as.numeric	If TRUE, the output is a numeric matrix containing posterior expectations of the imputed SNPs. Otherwise the output matrix is of the same class as snps and contains uncertain genotype calls

Value

A matrix with imputed SNPs as columns. The imputed values are the estimated expected values of each SNP when coded 0, 1 or 2.

Author(s)

David Clayton <dc208@cam.ac.uk>

References

Wallace, C. et al. (2010) *Nature Genetics*, **42**:68-71

See Also

snp.imputation

Examples

```
# Remove 5 SNPs from a datset and derive imputation rules for them
data(for.exercise)
sel <- c(20, 1000, 2000, 3000, 5000)
to.impute <- snps.10[,sel]
impute.from <- snps.10[,-sel]
pos.to <- snp.support$position[sel]
```

```
pos.fr <- snp.support$position[-sel]
imp <- snp.imputation(impute.from, to.impute, pos.fr, pos.to)
# Now calculate the imputed values
imputed <- impute.snps(imp, impute.from)
```

---

ld *Pairwise linkage disequilibrium measures*

Description

This function calculates measures of linkage disequilibrium between pairs of SNPs. The two SNPs in each pair may both come from the same `SnpMatrix` object, or from two different `SnpMatrix` objects. Statistics which can be calculated are the log likelihood ratio, odds ratio, Yule's Q, covariance, D-prime, R-squared, and R.

Usage

```
ld(x, y = NULL, depth = NULL, stats, symmetric = FALSE)
```

Arguments

`x`	An object of class `SnpMatrix` or `XSnpMatrix`
`y`	(Optional) Another object of the same class as `x`. If `y` is supplied, LD statistics are calculated between each column of `x` and each column of `y`. Otherwise, they are calculated between columns of `x`
`depth`	When `y` is not supplied, this parameter is mandatory and controls the maximum lag between columns of `x` considered. Thus, LD statistics are calculated between `x[,i]` and `x[,j]` only if `i` and `j` differ by no more than `depth`
`stats`	A character vector specifying the linkage disequilibrium measures to be calculated. This should contain one or more of the strings: `"LLR"`, `"OR"`, `"Q"`, `"Covar"`, `"D.prime"`, `"R.squared"`, ad `"R"`
`symmetric`	When no `y` argument is supplied this argument controls the format of the output band matrices. If `TRUE`, symmetric matrices are returned and, otherwise, an upper triangular matrices are returned

Details

For each pair of SNPs, phased haplotype frequencies are first estimated by maximum likelihood using the method described by Clayton and Leung (2007). The arrays of chosen LD statistics are then calculated and returned, either as band matrices (when `y` is not supplied), or as conventional rectangular matrices (when `y` is supplied). Band matrices are stored in compressed form as objects of class `dsCMatrix` (symmetric) or `dgCMatrix` (upper triangular). These classes are defined in the `"Matrix"` package)

Value

If only one LD statistic is requested, the function returns either a matrix or a compressed band matrix. If more than one LD statistic is requested, a list of such objects is returned

Author(s)

David Clayton <dc208@cam.ac.uk>

References

Clayton and Leung (2007) *Human Heredity*, **64**:45-51, (this paper is included in package documentation)

See Also

"Matrix-class"

Examples

```
data(testdata)
ld1 <- ld(Autosomes[, 1:50], depth=10, stats=c("D.prime", "R.squared"))
ld2 <- ld(Autosomes[, 1:20], Autosomes[, 21:25], stats="R.squared")
```

---

ld.example *Datasets to illustrate calculation of linkage disequilibrium statistics*

Description

This R data file contains data from the International HapMap project, concerning 603 SNPs spanning a one megabase region on chromosome 22, in a sample of Europeans and a sample of Africans

Format

There are three objects in the file:

- ceph.1m: A snpMatrix object containing the European genotype data
- yri.1m: A snpMatrix object containing the African genotype data
- support.ld: A dataframe containing details (chromosome position etc.

of the 603 SNPs

Source

http://hapmap.ncbi.nlm.nih.gov

References

The International HapMap Consortium. The International HapMap Project. *Nature* **426**:789-796 (2003)

mean2g *Raw coding of posterior probabilities of SNP genotype*

Description

An uncertain SNP genotype call is represented by the three posterior probabilities of the three possible calls. In the class SnpMatrix, an approximation to these is packed into a single 1-byte variable of type raw. These functions carry out this coding (and decoding).

Usage

```
post2g(p, transpose = FALSE)
mean2g(m, maxE = FALSE)
g2post(g, transpose = FALSE)
```

Arguments

p	A matrix of posterior probabilities. If transpose is FALSE this is Nx3, otherwise 3xN
m	A vector of posterior means
g	A raw vector of genotype codes
transpose	A logical flag indicating transposition of the matric of posterior probabilities (see Description)
maxE	A logical flag selecting the maximum entropy option in mean2g

Details

post2g and g2post convert from posterior probabilities to raw code and back respectively. If only the posterior expectation of the genotype (when numerically coded 0, 1, or 2) is available, no unique soultion exists in general and the behaviour of the function is determined by the value of maxE. If TRUE, then the maximum entropy solutions are returned while, if FALSE, an attempt is made to return the least uncertain solution, by setting the posterior probability of the BB genotype to zero when the posterior mean is less than 1.0 and the probability of AA to zero when the mean is greater than 1.0.

Value

post2g and mean2g return a vector of type raw. g2post returns a numeric matrix.

Note

These functions are provided mainly for users wishing to write their own data input functions.

Author(s)

David Clayton <dc208@cam.ac.uk>

Examples

```
data(testdata)
g <- Autosomes[1:10, 20] ## A vector of codes
p <- g2post(g)           ## Transform to probabilities ...
pg <- post2g(p)          ## ... and back to codes
m <- p[,2]+2*p[,3]       ## Posterior expectations
mg <- mean2g(m)          ## Transform to codes ...
pmg <- g2post(mg)        ## ... and transform to probabilities
## Write everything out
print(cbind(as(g, "numeric"), p, as.numeric(pg), m, as.numeric(mg), pmg))
```

`misinherits` *Find non-Mendelian inheritances in family data*

Description

For SNP data in families, this function locates all subjects whose parents are in the dataset and tests each SNP for non-Mendelian inheritances in these trios.

Usage

```
misinherits(ped, id, father, mother, data = sys.parent(), snp.data)
```

Arguments

`ped`	Pedigree identifiers
`id`	Subject identifiers
`father`	Identifiers for subjects' fathers
`mother`	Identifiers for subjects' mothers
`data`	A data frame in which to evaluate the previous four arguments
`snp.data`	An object of class `"SnpMatrix"` containing the SNP genotypes to be tested

Details

The first four arguments are usually derived from a "pedfile". If a data frame is supplied for the `data` argument, the first four arguments will be evaluated in this frame. Otherwise they will be evaluated in the calling environment. If the arguments are missing, they will be assumed to be in their usual positions in the pedfile data frame i.e. in columns one to four. If the pedfile data are obtained from a dataframe, the row names of the `data` and `snp.data` files will be used to align the pedfile and SNP data. Otherwise, these vectors will be assumed to be in the same order as the rows of `snp.data`.

Value

A logical matrix. Rows are subjects with any non-Mendelian inheritances and columns are SNPs with any non-Mendelian inheritances. The body of the matrix details whether each subject has non-Mendelian inheritance at each SNP. If a subject has no recorded genotype for a specific SNP, the corresponding element of the output matrix is set to NA.

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

tdt.snp

Examples

```
data(families)
misinherits(data=pedData, snp.data=genotypes)
```

---

mvtests *Multivariate SNP tests*

Description

This function calculates multivariate score tests between a multivariate (or multinomial) phenotype and sets of SNPs

Usage

```
mvtests(phenotype, sets, stratum, data = sys.parent(), snp.data, rules = NULL, complete = TRUE, uncertai
```

Arguments

phenotype	Either a factor (for a multinomial phenotype) or a matrix (for a multivariate phenotype)
sets	A list of sets of SNPs to be tested against the phenotype
stratum	(Optional) a stratifying variable
data	A data frame in which phenotype and stratum reside. If absent these are assumed to be in the parent frame and correctly aligned with the rows of snp.data
snp.data	An object of class SnpMatrix containing the SNP data
rules	(Optional) A set of imputation rules. The function then carries out tess on imputed SNPs
complete	If TRUE each test will use only subjects who have complete data for the phenotype and all SNPs in the set to be tested. If FALSE, then complete data for the phenotype is required, but tests are based upon complete pairs of SNPs
uncertain	If TRUE, uncertain genotype calls will be used in the tests (scored by their posterior expectations). Otherwise such calls are treated as missing
score	If TRUE, the score vectors and their variance-covariance matrices are saved in the output object for further processing

Details

Currently complete=FALSE is not implemented

Value

An object of class snp.tests.glm or GlmTests.score depending on whether score is set to FALSE or TRUE in the call

Note

This is an experimental version

Author(s)

David Clayton <dc208@cam.ac.uk>

Examples

```
## No example yet
```

---

plotUncertainty *Plot posterior probabilities of genotype assignment*

Description

The snpStats package allows for storage of uncertain genotype assignments in a one byte "raw" variable. The probabilities of assignment form a three-vector, subject to the linear constraint that they sum to 1.0; their possible values are grouped into 253 different classes. This function displays counts of these classes on a two-dimensional isometric plot.

Usage

```
plotUncertainty(snp, nlevels = 10, color.palette = heat.colors(nlevels))
```

Arguments

snp	One or more columns of a SnpMatrix object
nlevels	Probability cells are coloured according to frequency. This argument gives the number of colours that can be used
color.palette	The colour palette to be used

Details

The plot takes the form of an equilateral triangle in which each apex represents a certain assignment to one of the three genotypes. A point within the triangle represents, by the perpendicular distance from each side, the three probabilities. Each of the 253 probability classes is represented by a hexagonal cell, coloured according to its frequency in the data, which is also written within the cell

Author(s)

David Clayton <dc208@cam.ac.uk>

Examples

```
## No example available yet
```

---

pool *Pool test results from several studies or sub-studies*

Description

Given the same set of "score" tests carried out in several studies or in several different sub-samples within a study, this function pools the evidence by summation of the score statistics and score variances. It combines tests produced by `single.snp.tests` or by `snp.lhs.tests` and `snp.rhs.tests`.

Usage

```
pool(..., score = FALSE)
```

Arguments

`...`	Objects holding the (extended) test results. These must be of class `SingleSnpTests.score` or `snp.tests.glm`
`score`	Is extended score information to be returned in the output object? Relevant only for `SingleSnpTestsScore` objects

Details

This function works by recursive calls to the generic function `pool2` which pools the results of two studies.

Value

An object of same class as the input objects (optionally without the `.score`) extension. Tests are produced for the *union* of SNPs tested in all the input objects.

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

`pool2`, `SingleSnpTestsScore-class`, `GlmTests-class`, `single.snp.tests`, `snp.lhs.tests`, `snp.rhs.tests`

Examples

```
# An artificial example which simply doubles the size of a study
data(testdata)
sst <- single.snp.tests(snp.data=Autosomes, cc, data=subject.data,
         score=TRUE)
sst2 <- pool(sst, sst)
summary(sst2)
```

---

pool2 *Pool results of tests from two independent datasets*

Description

Generic function to pool results of tests from two independent datasets. It is not designed to be called directly, but is called recursively by pool

Usage

```
pool2(x, y, score)
```

Arguments

x, y	Objects holding the (extended) test results. These must be of class SingleSnpTests.score or snp.tests.glm
score	Is extended score information to be returned in the output object?

Value

An object of same class as the input objects (optionally without the .score) extension. Tests are produced for the *union* of SNPs tested in all the input objects.

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

pool, SingleSnpTestsScore-class, GlmTests-class, single.snp.tests, snp.lhs.tests, snp.rhs.tests

pp *Unpack posterior probabilities from one-byte codes*

Description

In snpStats, the three "posterior" probabilities corresponding to the possible values of an uncertain genotype are packed into a single byte code (with, of course, some loss in accuracy). This function, which is provided as an aid to writing new functions, unpacks the posterior probabilities from the single byte codes.

Usage

```
pp(x, transpose = FALSE)
```

Arguments

x	A vector, length N, which can be coerced into type raw
transpose	If FALSE, the result is an Nx3 matrix of posterior probabilities. If TRUE, a 3xN matrix is returned.

Value

A numeric matrix

Author(s)

David Clayton <dc208@cam.ac.uk>

Examples

```
##
## Read imputed data from a file produced by MACH
##
path <- system.file("extdata/mach1.out.mlprob.gz", package="snpStats")
mach <- read.mach(path)
pp(mach[1:50, 10])
```

qq.chisq *Quantile-quantile plot for chi-squared tests*

Description

This function plots ranked observed chi-squared test statistics against the corresponding expected order statistics. It also estimates an inflation (or deflation) factor, lambda, by the ratio of the trimmed means of observed and expected values. This is useful for inspecting the results of whole-genome association studies for overdispersion due to population substructure and other sources of bias or confounding.

Usage

```
qq.chisq(x, df=1, x.max, main="QQ plot",
    sub=paste("Expected distribution: chi-squared (",df," df)", sep=""),
    xlab="Expected", ylab="Observed",
    conc=c(0.025, 0.975), overdisp=FALSE, trim=0.5,
    slope.one=FALSE, slope.lambda=FALSE, pvals=FALSE,
    thin=c(0.25,50), oor.pch=24, col.shade="gray", ...)
```

Arguments

x	A vector of observed chi-squared test values
df	The degreees of freedom for the tests
x.max	If present, truncate the observed value (Y) axis at abs(x.max). If x.max is negative, the y-axis will extend to abs(x.max) even if the observed data do not
main	The main heading
sub	The subheading
xlab	x-axis label (default "Expected")
ylab	y-axis label (default "Observed")
conc	Lower and upper probability bounds for concentration band for the plot. Set this to NA to suppress this
overdisp	If TRUE, an overdispersion factor, lambda, will be estimated and used in calculating concentration band
trim	Quantile point for trimmed mean calculations for estimation of lambda. Default is to trim at the median
slope.one	Is a line of slope one to be superimpsed?
slope.lambda	Is a line of slope lambda to be superimposed?
pvals	Are P-values to be indicated on an axis drawn on the right-hand side of the plot?
thin	A pair of numbers indicating how points will be thinned before plotting (see Details). If NA, no thinning will be carried out
oor.pch	Observed values greater than x.max are plotted at x.max. This argument sets the plotting symbol to be used for out-of-range observations
col.shade	The colour with which the concentration band will be filled
...	Further graphical parameter settings to be passed to points()

Details

To reduce plotting time and the size of plot files, the smallest observed and expected points are thinned so that only a reduced number of (approximately equally spaced) points are plotted. The precise behaviour is controlled by the parameter thin, whose value should be a pair of numbers. The first number must lie between 0 and 1 and sets the proportion of the X axis over which thinning is to be applied. The second number should be an integer and sets the maximum number of points to be plotted in this section.

The "concentration band" for the plot is shown in grey. This region is defined by upper and lower probability bounds for each order statistic. The default is to use the 2.5 Note that this is not a

simultaneous confidence region; the probability that the plot will stray outside the band at some point exceeds 95

When required, the dispersion factor is estimated by the ratio of the observed trimmed mean to its expected value under the chi-squared assumption.

Value

The function returns the number of tests, the number of values omitted from the plot (greater than `x.max`), and the estimated dispersion factor, lambda.

Note

All tests must have the same number of degrees of freedom. If this is not the case, I suggest transforming to p-values and then plotting -2log(p) as chi-squared on 2 df.

Author(s)

David Clayton <dc208@cam.ac.uk>

References

Devlin, B. and Roeder, K. (1999) Genomic control for association studies. *Biometrics*, **55**:997-1004

See Also

`single.snp.tests`, `snp.lhs.tests`, `snp.rhs.tests`

Examples

```
## See example the single.snp.tests() function
```

---

random.snps *Generate random SnpMatrix*

Description

This function is purely for testing purposes. It can generate SnpMatrix objects which contain more than 2^31-1 elements.

Usage

```
random.snps(nrows, ncols)
```

Arguments

`nrows`	The number of rows to be generated
`ncols`	The number of columns to be generated

Details

All SNPs should be in Hardy-Weinberg equilibrium with an allele frequency of 0.5.

Note that, although the total number of elements can exceed 2^31-1, the numbers of rows and columns are still subject to this limit.

Examples

```
x <- random.snps(100,10)
col.summary(x)
```

`read.beagle` *Read genotypes imputed by the BEAGLE program*

Description

The BEAGLE program generates, for each SNP and each subject, posterior probabilities for the three genotypes. This function reads such data as a `SnpMatrix` object, storing the posterior probabilities to as much accuracy allowed by a one-byte coding

Usage

```
read.beagle(file, rownames=NULL, nsnp = NULL, header=TRUE)
```

Arguments

`file`	The input file name. This file my be gzipped.
`rownames`	The row names (sample identifiers) for the matrix
`nsnp`	The number of SNPs to be read in. This corresponds with the number of lines in the input file. If not supplied, the function does a preliminary pass to determine the number of lines
`header`	Set this `TRUE` if the file contains a header line (it won't for older versions of BEAGLE)

Details

In later versions of BEAGLE, row names are listed on a header line. However, if the `rownames` argument is supplied, this will take precedence over the header line. If there is no header line and no row names are supplied, names are generated as `Sample1`, `Sample2` etc.

No provision is made for data for the X chromosome. Such data must be first read as a `SnpMatrix` and subsequently coerced to an `XSnpMatrix` object

Value

an object of class `SnpMatrix`

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

SnpMatrix-class

Examples

```
##---- No example available yet
```

---

read.impute *Read genotypes imputed by the IMPUTE2 program*

Description

The IMPUTE2 program generates, for each SNP and each subject, posterior probabilities for the three genotypes. This function reads such data as a SnpMatrix object, storing the posterior probabilities to as much accuracy allowed by a one-byte coding

Usage

```
read.impute(file, rownames = NULL, nsnp = NULL, snpcol = 2)
```

Arguments

file	The input file name. This file my be gzipped.
rownames	The row names for the output object. Note that these correspond to groups of three columns in the input file. If not supplied, names are generated as Sample1, Sample2 etc.
nsnp	The number of SNPs to be read in. This corresponds with the number of lines in the input file. If not supplied, the function does a preliminary pass to determine the number of lines
snpcol	Which column of the input will be used as the SNP name. Default is column 2

Details

No provision is made for data for the X chromosome. Such data must be first read as a SnpMatrix and subsequently coerced to an XSnpMatrix object

Value

an object of class SnpMatrix

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

SnpMatrix-class

Examples

```
##---- No example available yet
```

---

read.long *Read SNP genotype data in long format*

Description

This function reads SNP genotype data from a file in which each line refers to a single genotype call. Replaces the earlier function `read.snps.long`.

Usage

```
read.long(file, samples, snps,
            fields = c(snp = 1, sample = 2, genotype = 3, confidence = 4,
                       allele.A = NA, allele.B = NA),
            split = "\t| +", gcodes, no.call = "", threshold = NULL,
            lex.order = FALSE, verbose = FALSE)
```

Arguments

`file`	Name(s) of file(s) to be read (can be gzipped)
`samples`	Either a vector of sample identifiers, or the number of samples to be read. If a single file is to be read and this argument is omitted, the file will be scanned initially and all samples will be included
`snps`	Either a vector of SNP identifiers, or the number of SNPs to be read. If a single file is to be read and this argument is omitted, the file will be scanned initially and all SNPs will be included
`fields`	A named vector giving the locations of the required fields. See Details below
`split`	A regular expression specifying how the input line will be split into fields. The default value specifies separation of fields by a TAB character, or by one or more blanks
`gcodes`	When the genotype is read as a single field, this argument specifies how it is handled. See Details below.
`no.call`	The string which indicates "no call" for either a genotype or (when the genotype is read as two allele fields) an allele
`threshold`	A vector of length 2 giving the lower and higher acceptable limits for the confidence score
`lex.order`	If `TRUE`, the alleles at each locus will be in lexographical order. Otherwise, ordering of alleles is arbitrary, depending on the order in which they are encountered
`verbose`	If `TRUE`, this turns on output from the function. Otherwise only error and warning messages are produced

Details

Each line on the input file represents a single call and is split into fields using the function `strsplit`. The required fields are extracted according to the `fields` argument. This *must* contain the locations of the sample and snp identifier fields and *either* the location of a genotype field *or* the locations of two allele fields.

If the `samples` and `snps` arguments contain vectors of character strings, a `SnpMatrix` is created with these row and column names and the genotype values are "cherry-picked" from the input file. If either, or both, of these arguments are specified simply as numbers, then these numbers determine the *dimensions* of the `SnpMatrix` created. In this case samples and/or SNPs are included in the `SnpMatrix` on a first-come-first-served basis. If either or both of these arguments are omitted, a preliminary scan of the input file is carried out to find the missing sample and/or SNP identifiers. In this scan, when a sample or SNP identifier differs from that in the previous line, but is identical to one previously found, then all the relevant identifiers are assumed to have been found. This implies that the file must be sorted, in some consistent order, by sample and by SNP (although either one of these may vary fastest).

If the genotype is to be read as a single field, the `genotype` element of the `fields` argument must be set to the appropriate value, and the `allele.A` and `allele.B` elements should be set to `NA`. Its handling is controlled by the `gcodes` argument. If this is missing or `NA`, then the genotype is assumed to be represented by a two-character field, the two characters representing the two alleles. If `gcodes` is a single string, then it is assumed to contain a regular expression which will split the genotype field into two allele fields. Otherwise, `gcode` must be an array of length three, specifying the three genotype codes in the order "AA", "AB", "BB".

If the two alleles of the genotype are to be read from two separate fields, the `genotype` element should be set to `NA` and the `allele.A` and `allele.B` elements set to the appropriate values. The `gcode` argument should be missing or set to `NA`.

Value

If the genotype is read as a single field matching one of three specified codes, the function returns an object of class `SnpMatrix`. Otherwise it returns a list whose first element is the `SnpMatrix` object and whose second element is a dataframe containing the allele codes, with the SNP identifiers as row names. Note that allele codes only occur in this file if they occur in a genotype which was accepted. Thus, monomorphic SNPs have `allele.B` coded as `NA`, and SNPs which never pass confidence score filters have both alleles coded as `NA`.

Note

Unlike `read.snps.long`, this function is written entirely in R and may not be particularly fast. However, it imposes no restrictions on the allele codes recognized.

Homozygous genotypes are assumed to be represented in the input file by coding both alleles to the same value. No special provision is made to read `XSnpMatrix` objects; such data should first be read as a `SnpMatrix` and then coerced to an `XSnpMatrix` using `new` or `as`.

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

SnpMatrix-class, XSnpMatrix-class

Examples

```
##
## No example supplied yet
##
```

---

read.mach *Read genotypes imputed by the MACH program*

Description

This routine reads imputed genotypes generated by the MACH program. With the `--mle` and `--mldetails` options in force this program generates a `.mlprob` output file which contains probabilities of assignments. These are stored as uncertain genotype calls in a `SnpMatrix` object

Usage

```
read.mach(file, colnames = NULL, nrow = NULL)
```

Arguments

`file`	The name of the `.mlprob` file. This may be gzipped
`colnames`	The column names. If absent, names are generated as `SNP1`, `SNP2`, etc.
`nrow`	If known the number of rows of data on the file. If not supplied, it is determined by a preliminary pass through the data

Details

No routine is explicitly available for data on chromosome X. Such data should first be read as a `SnpMatrix` and then coerced to an `XSnpMatrix` object

Value

An object of class `SnpMatrix`

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

SnpMatrix-class

Examples

```
##---- No example available yet
```

`read.pedfile` *Read a pedfile as "SnpMatrix" object*

Description

Reads diallelic data in linkage "pedfile" format, with one line of data per sample (subject) containing six mandatory fields followed by pairs of fields, one pair for each locus, giving the two alleles observed.

Usage

```
read.pedfile(file, n, snps, which, split = "\t| +", sep = ".", na.strings = "0", lex.order = FALSE)
```

Arguments

`file`	The input pedfile. This may be (but need not be) gzipped
`n`	(Optional) The number of lines of data to be read. If not supplied the pedfile is read once and rewound to determine how many lines it contains
`snps`	(Optional) Either a character vector giving the names of the loci, or a single character variable giving the name of a locus information file from which these can be read. This file is assumed to be white-space delimited with one line per locus and no header line. If this argument is not supplied, locus names are generated as a numerical sequence, prefixed by `locus` and a separator character
`which`	(Optional) If locus names are to be read from a file, this argument should specify which column contains the names. If not supplied, the first column giving unique locus names is used
`split`	A "regexp" specifying how the input pedfile will be split into fields. The default value specifies either a TAB character or one or more spaces
`sep`	The separator character used in constructing row and column names of the output `SnpMatrix` object
`na.strings`	One or more strings to be set to `NA`. Any field taking one of these values will be set to `NA`
`lex.order`	If TRUE, then alleles will be allocated to internal `1` and `2` values in lexographic order. Otherwise they are converted in the order in which they are encountered when reading the file (the default setting)

Details

Row names for the output `SnpMatrix` object and for the accompanying subject description dataframe are taken as the pedigree identifiers, when these provide the required unique identifiers. When these are duplicated, an attempt is made to use the pedigree-member identifiers instead but, when these too are duplicated, row names are obtained by concatenating, with a separator character, the pedigree and pedigree-member identifiers.

Value

A list, comprising

genotypes	The output genotype data as an object of class "SnpMatrix". If either the pedigree or pedigree-member identifiers in the ped file are not duplicated, these are used for the row names of the output object. Otherwise these two fields are concatenated, separated by sep
fam	A dataframe containing the first six fields in the pedfile. The row names will correspond with those of the SnpMatrix
map	A dataframe giving the alleles at each locus. If locus names were obtained from a dataframe read from an existing file, then the allele information is simply appended to this frame. Otherwise a new dataframe is created. The row names will correspond with the column names of the SnpMatrix

Note

This function is written entirely in R and may not be particularly fast. However, it imposes no restrictions on the allele codes recognized.

Homozygous genotypes may be represented in the input file either (a) by coding both alleles to the same value, or (b) setting the second allele to "missing" (as specified by the missing.allele argument). No special provision is made to read XSnpMatrix objects; such data should first be read as a SnpMatrix and then coerced to an XSnpMatrix using new or as.

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

SnpMatrix-class, XSnpMatrix-class

Examples

```
##
## No example supplied yet
##
```

---

read.plink *Read a PLINK binary data file as a SnpMatrix*

Description

The package PLINK saves genome-wide association data in groups of three files, with the extensions .bed, .bim, and .fam. This function reads these files and creates an object of class "SnpMatrix"

Usage

```
read.plink(bed, bim, fam, na.strings = c("0", "-9"), sep = "." , select.subjects = NULL, select.snps = NU
```

Arguments

bed	The name of the file containing the packed binary SNP genotype data. It should have the extension .bed; if it doesn't, then this extension will be appended
bim	The file containing the SNP descriptions
fam	The file containing subject (and, possibly, family) identifiers. This is basically a tab-delimited "pedfile"
na.strings	Strings in .bam and .fam files to be recoded as NA
sep	A separator character for constructing unique subject identifiers
select.subjects	A numeric vector indicating a subset of subjects to be selected from the input file (see details)
select.snps	Either a numeric or a character vector indicating a subset of SNPs to be selected from the input file (see details)

Details

If the bed argument does not contain a filename with the file extension .bed, then this extension is appended to the argument. The remaining two arguments are optional; their default values are obtained by replacing the .bed filename extension by .bim and .fam respectively. See the PLINK documentation for the detailed specification of these files.

The select.subjects or select.snps argument can be used to read a subset of the data. Use of select.snps requires that the .bed file is in SNP-major order (the default in PLINK). Likewise, use of select.snps requires that the .bed file is in individual-major order. Subjects are selected by their numeric order in the PLINK files, while SNPs are selected either by order or by name. Note that the order of selected SNPs/subjects in the output objects will be the same as their order in the PLINK files.

Row names for the output SnpMatrix object and for the accompanying subject description dataframe are taken as the pedigree identifiers, when these provide the required unique identifiers. When these are duplicated, an attempt is made to use the pedigree-member identifiers instead but, when these too are duplicated, row names are obtained by concatenating, with a separator character, the pedigree and pedigree-member identifiers.

Value

A list with three elements:

genotypes	The output genotype data as an object of class
"SnpMatrix".	
fam	A dataframe corresponding to the .fam file, containing the first six fields in a standard pedfile. The row names will correspond with those of the SnpMatrix
map	A dataframe correponding to the .bim file. the row names correpond with the column names of the SnpMatrix

Note

No special provision is made to read XSnpMatrix objects; such data should first be read as a SnpMatrix and then coerced to an XSnpMatrix using new or as.

Author(s)

David Clayton <dc208@cam.ac.uk>

References

PLINK: Whole genome association analysis toolset. http://pngu.mgh.harvard.edu/~purcell/plink/

See Also

write.plink, SnpMatrix-class, XSnpMatrix-class

---

read.snps.long *Read SNP data in long format (deprecated)*

Description

Reads SNP data when organized in free format as one call per line. Other than the one call per line requirement, there is considerable flexibility. Multiple input files can be read, the input fields can be in any order on the line, and irrelevant fields can be skipped. The samples and SNPs to be read must be pre-specified, and define rows and columns of an output object of class "SnpMatrix". This function has been replaced in versions 1.3 and later by the more flexible function read.long.

Usage

```
read.snps.long(files, sample.id = NULL, snp.id = NULL, diploid = NULL,
              fields = c(sample = 1, snp = 2, genotype = 3, confidence = 4),
              codes = c("0", "1", "2"), threshold = 0.9, lower = TRUE,
              sep = " ", comment = "#", skip = 0, simplify = c(FALSE,FALSE),
              verbose = FALSE, in.order=TRUE, every = 1000)
```

Arguments

files	A character vector giving the names of the input files
sample.id	A character vector giving the identifiers of the samples to be read
snp.id	A character vector giving the names of the SNPs to be read
diploid	A logical array of the same length as sample.id, required if reading data into an XSnpMatrix rather than a SnpMatrix. This vector gives the expected ploidy for each row. If the same value suffices for all rows, then a scalar may be supplied

fields	A integer vector with named elements specifying the positions of the required fields in the input record. The fields are identified by the names sample and snp for the sample and SNP identifier fields, confidence for a call confidence score (if present) and either genotype if genotype calls occur as a single field, or allele1 and allele2 if the two alleles are coded in different fields
codes	Either the single string "nucleotide" denoting that coding in terms of nucleotides (A, C, G or T, case insensitive), or a character vector giving genotype or allele codes (see below)
threshold	A numerical value for the calling threshold on the confidence score
lower	If TRUE, then threshold represents a lower bound. Otherwise it is an upper bound
sep	The delimiting character separating fields in the input record
comment	A character denoting that any remaining input on a line is to be ignored
skip	An integer value specifying how many lines are to be skipped at the beginning of each data file
simplify	If TRUE, sample and SNP identifying strings will be shortened by removal of any common leading or trailing sequences when they are used as row and column names of the output SnpMatrix
verbose	If TRUE, a progress report is generated as every every lines of data are read
in.order	If TRUE, input lines are assumed to be in the correct order (see details)
every	See verbose

Details

If nucleotide coding is not used, the codes argument should be a character array giving the valid codes. For genotype coding of autosomal SNPs, this should be an array of length 3 giving the codes for the three genotypes, in the order homozygous(AA), heterozygous(AB), homozygous(BB). All other codes will be treated as "no call". The default codes are "0", "1", "2". For X SNPs, males are assumed to be coded as homozygous, unless an additional two codes are supplied (representing the AY and BY genotypes). For allele coding, the codes array should be of length 2 and should specify the codes for the two alleles. Again, any other code is treated as "missing" and, for X SNPs, males should be coded either as homozygous or by omission of the second allele.

For nucleotide coding, nucleotides are assigned to the nominal alleles in alphabetic order. Thus, for a SNP with either "T" and "A" nucleotides in the variant position, the nominal genotypes AA, AB and BB will refer to A/A, A/T and T/T.

Although the function allows for reading into an object of class XSnpMatrix directly, it is usually preferable to read such data as a "SnpMatrix" (i.e. as autosomal) and to coerce it to an object of type "XSnpMatrix" later using as(..., "X.SnpMatrix") or new("XSnpMatrix", ..., diploid=...). If diploid is coded NA for any subject the latter course *must* be followed, since NAs are not accepted in the diploid argument.

If the in.order argument is set TRUE, then the vectors sample.id and snp.id must be in the same order as they vary on the input file(s) and this ordering must be consistent. However, there is no requirement that either SNP or sample should vary fastest as this is detected from the input. If in.order is FALSE, then no assumptions about the ordering of the input file are assumed and SNP and sample identifiers are looked up in hash tables as they are read. This option must be expected,

therefore, to be somewhat slower. Each file may represent a separate sample or SNP, in which case the appropriate `.id` argument can be omitted; row or column names are then taken from the file names.

Value

An object of class `"SnpMatrix"` or `"XSnpMatrix"`.

Note

The function will read gzipped files.

If `in.order` is `TRUE`, every combination of sample and snp listed in the `sample.id` and `snp.id` arguments *must* be present in the input file(s). Otherwise the function will search for any missing observation until reaching the end of the data, ignoring everything else on the way.

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

`read.plink`, `SnpMatrix-class`, `XSnpMatrix-class`

`row.summary` *Summarize rows or columns of a snp matrix*

Description

These function calculates summary statistics of each row or column of call rates and heterozygosity for each row of a an object of class `"SnpMatrix"` or `"XSnpMatrix"`

Usage

```
row.summary(object)
col.summary(object, rules = NULL, uncertain = TRUE)
```

Arguments

`object`	genotype data as a `SnpMatrix-class` or `XSnpMatrix-class` object
`rules`	An object of class `"ImputationRules"`. If supplied, the rules coded in this object are used, together with the snp genotype data in `object`, to generate imputed SNPs. The column summary of these imputed data are then returned
`uncertain`	If `TRUE` uncertain genotypes are used in calculation of allele and genotype frequencies (by scoring as posterior expectations). Otherwise, and for Hardy-Weinberg tests, they are ignored

Value

`row.summary` returns a data frame with rows corresponding to rows of the input object and with columns/elements:

- Call.rate: Proportion of SNPs called
- Certain.calls: Proportion of called SNPs with certain calls
- Heterozygosity: Proportion of called SNPs which are heterozygous

Uncertain calls are ignored for calculating the heterozygosity.

`col.summary` returns a data frame with rows corresponding to columns of the input object and with columns/elements:

- Calls: The number of valid calls
- Call.rate: The proportion of genotypes called
- Certain.calls: Proportion of called SNPs with certain calls
- RAF: The "risk" allele (allele `B`) frequency
- MAF: The minor allele frequency
- P.AA: The frequency of homozygous genotype 1 (A/A)
- P.AB: The frequency of heterozygous genotype 2 (A/B)
- P.BB: The frequency of homozygous genotype 3 (B/B)
- z.HWE: A z-test for Hardy-Weinberg equilibrium

For objects of class `"XSnpMatrix"`, the following additional columns are returned:

- P.AY: The frequency of allele A in males
- P.BY: The frequency of allele B in males
- Calls.female: The number of valid calls in females (only these calls are used in the z-test for HWE)

Note

The current version of `row.summary` does not deal with the X chromosome differently, so that males are counted as homozygous.

Author(s)

David Clayton <dc208@cam.ac.uk>

Examples

```
data(testdata)
rs <- row.summary(Autosomes)
summary(rs)
cs <- col.summary(Autosomes)
summary(cs)
cs <- col.summary(Xchromosome)
summary(cs)
```

`sample.ped.gz` *Sample datasets to illustrate data input*

Description

The first five files concern data on 20 diallelic loci on 120 subjects. These data are distributed with the Haploview package (Barrett et al., 2003). The sixth files contains a additional dataset of 18 SNPs in 100 subjects, coded in "long" format, and the seventh file duplicates this dataset in an alternative long format. These seven files are used in the data input vignette. The final file is a sample imputed genotype dataset distributed with the MACH imputation package, and used in the imputation vignette.

These files are stored in the `extdata` relative to the package base. Full file names can be obtained using the `system.file` function.

Format

The following files are described here:

- `sample.ped.gz`: A gzipped pedfile
- `sample.info`: An accompanying locus information file
- `sample.bed`: The corresponding PLINK `.bed` file
- `sample.bim`: The PLINK `.bim` file
- `sample.fam`: The PLINK `.fam` file
- `sample-long.gz`: A sample of long-formatted data
- `sample-long-alleles.gz`: The same as above, but allele-coded
- `mach1.out.mlprob.gz`: An `mlprob` output file from the MACH genotype imputation program. This file contains, for each imputed genotype call, posterior probabilities for the three possible genotypes

Source

http://www.broadinstitute.org/scientific-community/science/programs/medical-and-population-genetics haploview/downloads http://www.sph.umich.edu/csg/abecasis/MACH/download

References

Barrett JC, Fry B, Maller J, Daly MJ.(2005) Haploview: analysis and visualization of LD and haplotype maps. *Bioinformatics*, 2005 Jan 15, [PubMed ID: 15297300]

`single.snp.tests` *1-df and 2-df tests for genetic associations with SNPs (or imputed SNPs)*

Description

This function carries out tests for association between phenotype and a series of single nucleotide polymorphisms (SNPs), within strata defined by a possibly confounding factor. SNPs are considered one at a time and both 1-df and 2-df tests are calculated. For a binary phenotype, the 1-df test is the Cochran-Armitage test (or, when stratified, the Mantel-extension test). The function will also calculate the same tests for SNPs imputed by regression analysis.

Usage

```
single.snp.tests(phenotype, stratum, data = sys.parent(), snp.data,
   rules=NULL, subset, snp.subset, uncertain = FALSE, score=FALSE)
```

Arguments

`phenotype`	A vector containing the values of the phenotype
`stratum`	Optionally, a factor defining strata for the analysis
`data`	A dataframe containing the `phenotype` and `stratum` data. The row names of this are linked with the row names of the `snps` argument to establish correspondence of phenotype and genotype data. If this argument is not supplied, `phenotype` and `stratum` are evaluated in the calling environment and should be in the same order as rows of `snps`
`snp.data`	An object of class `"SnpMatrix"` containing the SNP genotypes to be tested
`rules`	An object of class `"ImputationRules"`. If supplied, the rules coded in this object are used, together with `snp.data`, to calculate tests for imputed SNPs
`subset`	A vector or expression describing the subset of subjects to be used in the analysis. This is evaluated in the same environment as the `phenotype` and `stratum` arguments
`snp.subset`	A vector describing the subset of SNPs to be considered. Default action is to test all SNPs in `snp.data` or, in imputation mode, as specified by `rules`
`uncertain`	If TRUE, uncertain genotypes are handled by replacing score contributions by their posterior expectations. Otherwise they are treated as missing. Setting this option authomatically invokes use of `robust` variance estimates
`score`	If `TRUE`, the output object will contain, for each SNP, the score vector and its variance-covariance matrix

Details

Formally, the test statistics are score tests for generalized linear models with canonical link. That is, they are inner products between genotype indicators and the deviations of phenotypes from their

stratum means. Variances (and covariances) are those of the permutation distribution obtained by randomly permuting phenotype within stratum.

When the function is used to calculate tests for imputed SNPs, the test is still a score test. The score statistics are calculated from the expected value, given observed SNPs, of the score statistic if the SNP to be tested were itself observed.

The `subset` argument can either be a logical vector of length equal to the length of the vector of phenotypes, an integer vector specifying positions in the `data` frame, or a character vector containing names of the selected rows in the `data` frame. Similarly, the `snp.subset` argument can be a logical, integer, or character vector.

Value

An object of class `"SingleSnpTests"`. If `score` is set to `TRUE`, the output object will be of the extended class `"SingleSnpTestsScore"` containing additional slots holding the score statistics and their variances (and covariances). This allows meta-analysis using the `pool` function.

Note

The 1 df imputation tests are described by Chapman et al. (2008) and the 2 df imputation tests are a simple extension of these. The behaviour of this function for objects of class `XSnpMatrix` is as described by Clayton (2008). Males are treated as homozygous females and corrected variance estimates are used.

Author(s)

David Clayton <dc208@cam.ac.uk>

References

Chapman J.M., Cooper J.D., Todd J.A. and Clayton D.G. (2003) *Human Heredity*, **56**:18-31.
Clayton (2008) Testing for association on the X chromosome *Biostatistics*, **9**:593-600.)

See Also

`snp.lhs.tests`, `snp.rhs.tests`, `impute.snps`, `ImputationRules-class`, `pool`, `SingleSnpTests-class`, `SingleSnpTestsScore-class`

Examples

```
data(testdata)
results <- single.snp.tests(cc, stratum=region, data=subject.data,
   snp.data=Autosomes, snp.subset=1:10)
print(summary(results))

# writing to an (anonymous and temporary) csv file
csvfile <- tempfile()
write.csv(file=csvfile, as(results, 'data.frame'))
unlink(csvfile)
# QQ plot
qq.chisq(chi.squared(results, 1), 1)
```

```
qq.chisq(chi.squared(results, 2), 2)
```

`SingleSnpTests-class` *Classes "SingleSnpTests" and "SingleSnpTestsScore"*

Description

These are classes to hold the objects created by `single.snp.tests` and provide methods for extracting key elements. The class `"SingleSnpTestsScore"` extends class `"SingleSnpTests"` to include the score and score variance statistics in order to provide methods for pooling results from several studies or parts of a study

Objects from the Class

Objects can be created by calls of the form `new("SingleSnpTests", ...)` and `new("SingleSnpTestsScore", ...)` but, more usually, will be created by calls to `single.snp.tests`

Slots

`snp.names`**:** The names of the SNPs tested, as they appear as column names in the original `SnpMatrix`

`chisq`**:** A two-column matrix holding the 1 and 2 df association tests

`N`**:** The numbers of observations included in each test

`N.r2`**:** For tests on imputed SNPs, the product of `N` and the imputation r^2. Otherwise a zero-length object

`U`**:** (class `"SingleSnpTestsScore"`) Score statistics

`V`**:** (class `"SingleSnpTestsScore"`) Score variances

Methods

[ ] `signature(x = "SingleSnpTests", i = "ANY")`: Subsetting operator

[ ] `signature(x = "SingleSnpTestsScore", i = "ANY")`: Subsetting operator

chi.squared `signature(x = "SingleSnpTests", df = "numeric")`: Extract 1- and 2-df chi-squared test values

effect.sign `signature(x = "SingleSnpTestsScore", simplify = "missing")`: Extract signs of associations tested by the 1df tests

names `signature(x="SingleSnpTests")`: Extract names of test values (`snp.names` slot)

p.value `signature(x = "SingleSnpTests", df = "numeric")`: Evaluate 1- and 2-df test p-values

show `signature(object = "SingleSnpTests")`: List all tests and p-values

coerce `signature(from = "SingleSnpTests", to = "data.frame")`: Conversion to data frame class

sample.size `signature(object = "SingleSnpTests")`: Extract sample sizes for tests

effective.sample.size `signature(object = "SingleSnpTests")`: Extract effective sample sizes for tests. For imputed tests, these are the real sample sizes multiplied by the corresponding R-squared values for imputation

summary `signature(object = "SingleSnpTests")`: Summarize all tests and p-values

pool2 `signature(x = "SingleSnpTestsScore",y = "SingleSnpTestsScore", score = "logical")`: Combine two sets of test results. Used recursively by `pool`

switch.alleles `signature(x = "SingleSnpTestsScore", snps = "ANY")`: Emulate, in the score vector and its (co)variances, the effect of switching of the alleles for the specified tests

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

`single.snp.tests`, `pool`

Examples

```
showClass("SingleSnpTests")
showClass("SingleSnpTestsScore")
```

`sm.compare` *Compare two SnpMatrix objects*

Description

For quality control purposes, it is sometimes necessary to compare genotype data derived from different sources. This function facilitates this.

Usage

```
sm.compare(obj1, obj2, row.wise = TRUE, col.wise = TRUE)
```

Arguments

`obj1`	The first of the two `SnpMatrix` objects to be compared
`obj2`	The second `SnpMatrix` object
`row.wise`	Calculate comparison statistics aggregated in a row-wise manner
`col.wise`	Calculate column-wise comparison statistics

Details

Initially row and column names of the two objects are compared to identify subsets of subjects and SNPs which they have in common. Then, every instance of a SNP genotype in the two objects are compared and agreements and disagreements counted by row and/or by column.

Value

If only one of the row-wise and column-wise summaries are to be calculated, the return value is a matrix with rows defined by subjects or SNPs and columns giving counts of:

`Agree`	Agreements (all)
`Disagree`	Disgreements (all)
`NA.agree`	Genotype coded `NA` in both objects
`NA.disagree`	Genotype coded `NA` in only one object
`Hom.agree`	Objects agree and genotype is homozygous
`Hom.switch`	Genotype coded as homozygous in both objects, but alleles switched
`Het.agree`	Genotype coded as heterozygous in both objects
`Het.Hom`	Genotype coded as heterozygous in one object and homozygous in the other

If both row-wise and column-wise summaries are computed (the default behaviour) , the function returns a list containing two matrices of the form described above. These are named `row.wise` and `col.wise`

Note

No special provision is yet made for objects of class `XSnpMatrix`, in which haploid calls are coded as homozygous.

Author(s)

David Clayton `<dc208@cam.ac.uk>`

See Also

`SnpMatrix-class`, `XSnpMatrix-class`

Examples

```
##
## No example yet available
##
```

---

snp.cor *Correlations with columns of a SnpMatrix*

Description

This function calculates Pearson correlation coefficients between columns of a `SnpMatrix` and columns of an ordinary matrix. The two matrices must have the same number of rows. All valid pairs are used in the computation of each correlation coefficient.

Usage

```
snp.cor(x, y, uncertain = FALSE)
```

Arguments

x	An N by M SnpMatrix
y	An N by P general matrix
uncertain	If TRUE, uncertain genotypes are replaced by posterior expectations. Otherwise these are treated as missing values

Details

This can be used together with xxt and eigen to calculate standardized loadings in the principal components

Value

An M by P matrix of correlation coefficients

Note

This version cannot handle X chromosomes

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

xxt

Examples

```
# make a SnpMatrix with a small number of rows
data(testdata)
small <- Autosomes[1:100,]
# Calculate the X.X-transpose matrix
xx <- xxt(small, correct.for.missing=TRUE)
# Calculate the principal components
pc <- eigen(xx, symmetric=TRUE)$vectors
# Calculate the loadings in first 10 components */
loadings <- snp.cor(small, pc[,1:10])
```

`snp.imputation` *Calculate imputation rules*

Description

Given two set of SNPs typed in the same subjects, this function calculates rules which can be used to impute one set from the other in a subsequent sample. The function can also calculate rules for imputing each SNP in a single dataset from other SNPs in the same dataset

Usage

```
snp.imputation(X, Y, pos.X, pos.Y, phase=FALSE, try=50, stopping=c(0.95, 4, 0.05),
               use.hap=c(1.0, 0.0), em.cntrl=c(50,0.01,10,0.01), minA=5)
```

Arguments

`X`	An object of class `"SnpMatrix"` or `"XSnpMatrix"` containing observations of the SNPs to be used for imputation ("predictor SNPs")
`Y`	An object of same class as `X` containing observations of the SNPs to be imputed in a future sample ("target SNPs"). If this argument is missing, then target SNPs are also drawn from `X`
`pos.X`	The positions of the predictor SNPs. Can be missing if there is no `Y` argument and the columns of `X` are in genome position order
`pos.Y`	The positions of the target SNPs. Only required when a `Y` argument is present
`phase`	See "Details" below
`try`	The number of potential predictor SNPs to be considered in the stepwise regression procedure around each target SNP . The nearest `try` predictor SNPs to each target SNP will be considered
`stopping`	Parameters of the stopping rule for the stepwise regression (see below)
`use.hap`	Parameters to control use of the haplotype imputation method (see below)
`em.cntrl`	Parameters to control test for convergence of EM algorithm for fitting phased haplotypes (see below)
`minA`	A minimum data quantity measure for estimating pairwise linkage disequilibrium (see below)

Details

The routine first carries out a series of step-wise least-square regression analyses in which each Y SNP is regressed on the nearest `try` predictor (X) SNPs. If `phase` is `TRUE`, the regressions will be calculated at the chromosome (haplotype) level, variances being simply $p(1-p)$ and covariances estimated from the estimated two-locus haplotypes (this option is not yet implemented). Otherwise, the analysis is carried out at the genotype level based on conventional variance and covariance estimates using the `"pairwise.complete.obs"` missing value treatment (see `cov`). New SNPs are added to the regression until either (a) the value of R^2 exceeds the first parameter of `stopping`,

(b) the number of "tag" SNPs has reached the maximum set in the second parameter of `stopping`, or (c) the change in R^2 does not achieve the target set by the third parameter of `stopping`. If the third parameter of `stopping` is `NA`, this last test is replaced by a test for improvement in the Akaike information criterion (AIC).

After choosing the set of "tag" SNPs in this way, a prediction rule is generated either by calculating phased haplotype frequencies, either (a) under a log-linear model for linkage disequilibrium with only first order association terms fitted, or (b) under the "saturated" model. These methods do not differ if there is only one tag SNP but, otherwise, choice between methods is controlled by the `use.hap` parameters. If the prediction, as measure by R^2 achieved with the log-linear smoothing model exceeds a threshold (the first parameter of `use.hap`) then this method is used. Otherwise, if the gain in R^2 achieved by using the second method exceeds the second parameter of `use.hap`, then the second method is used. Current experience is that, the log-linear method is rarely preferred with reasonable choices for `use.hap`, and imputation is much faster when the second method only is considered. The current default ensures that this second method is used, but the other possibility might be considered if imputing from very small samples; however this code is not extensively tested and should be regarded as experimental.

The argument `em.cntrl` controls convergence testing for the EM algorithm for fitting haplotype frequencies and the IPF algorithm for fitting the log-linear model. The first parameter is the maximum number of EM iterations, and the second parameter is the threshold for the change in log likelihood below which the iteration is judged to have converged. The third and fourth parameters give the maximum number of IPF iterations and the convergence tolerance. There should be no need to change the default values.

All SNPs selected for imputation must have sufficient data for estimating pairwise linkage disequilibrium with each other and with the target SNP. The statistic chosen is based on the four-fold tables of two-locus haplotype frequencies. If the frequencies in such a table are labelled a, b, c and d then, if $ad > bc$ then $t = min(a, d)$ and, otherwise, $t = min(b, c)$. The cell frequencies t must exceed `minA` for all pairwise comparisons.

Value

An object of class `"ImputationRules"`.

Note

The `phase=TRUE` option is not yet implemented

Author(s)

David Clayton <dc208@cam.ac.uk>

References

Chapman J.M., Cooper J.D., Todd J.A. and Clayton D.G. (2003) *Human Heredity*, **56**:18-31.

Wallace, C. et al. (2010) *Nature Genetics*, **42**:68-71

See Also

`ImputationRules-class`, `imputation.maf`, `imputation.r2`

Examples

```
# Remove 5 SNPs from a datset and derive imputation rules for them
data(for.exercise)
sel <- c(20, 1000, 2000, 3000, 5000)
to.impute <- snps.10[,sel]
impute.from <- snps.10[,-sel]
pos.to <- snp.support$position[sel]
pos.fr <- snp.support$position[-sel]
imp <- snp.imputation(impute.from, to.impute, pos.fr, pos.to)
```

---

snp.lhs.estimates *Logistic regression with SNP genotypes as dependent variable*

Description

Under the assumption of Hardy-Weinberg equilibrium, a SNP genotype is a binomial variate with two trials for an autosomal SNP or with one or two trials (depending on sex) for a SNP on the X chromosome. With each SNP in an input `"SnpMatrix"` as dependent variable, this function fits a logistic regression model. The Hardy-Weinberg assumption can be relaxed by use of a "robust" option.

Usage

```
snp.lhs.estimates(snp.data, base.formula, add.formula, subset, snp.subset,
                  data = sys.parent(), robust = FALSE, uncertain = FALSE,
                  control=glm.test.control())
```

Arguments

`snp.data`	The SNP data, as an object of class `"SnpMatrix"` or `"XSnpMatrix"`
`base.formula`	A `formula` object describing a base model containing those terms which are to be fitted but for which parameter estimates are not required (the dependent variable is omitted from the model formula)
`add.formula`	A `formula` object describing the additional terms in the model for which parameter estimates are required (again, the dependent variable is omitted)
`subset`	An array describing the subset of observations to be considered
`snp.subset`	An array describing the subset of SNPs to be considered. Default action is to test all SNPs.
`data`	The data frame in which `base.formula`, `add.formula` and `subset` are to be evaluated
`robust`	If `TRUE`, Hardy-Weinberg equilibrium will is not assumed in calculating the variance-covariance matrix of parameter estimates
`uncertain`	If `TRUE`, uncertain genotypes are used and scored by their posterior expectations. Otherwise they are treated as missing. If set, this option forces `robust` variance estimates
`control`	An object giving parameters for the IRLS algorithm fitting of the base model and for the acceptable aliasing amongst new terms to be tested. See `glm.test.control`

Details

The model fitted is the union of the `base.formula` and `add.formula` models, although parameter estimates (and their variance-covariance matrix) are only generated for the parameters of the latter. The "robust" option causes a Huber-White "sandwich" estimate of the variance-covariance matrix to be used in place of the usual inverse second derivative matrix of the log-likelihood (which assumes Hardy-Weinberg equilibrium). If a `data` argument is supplied, the `snp.data` and `data` objects are aligned by rowname. Otherwise all variables in the model formulae are assumed to be stored in the same order as the columns of the `snp.data` object.

Value

An object of class `GlmEstimates`

Note

A factor (or several factors) may be included as arguments to the function `strata(...)` in the `base.formula`. This fits all interactions of the factors so included, but leads to faster computation than fitting these in the normal way. Additionally, a `cluster(...)` call may be included in the base model formula. This identifies clusters of potentially correlated observations (e.g. for members of the same family); in this case, an appropriate robust estimate of the variance-covariance matrix of parameter estimates is calculated. No more than one `strata()` call may be used, and neither `strata(...)` or `cluster(...)` calls may appear in the `add.formula`.

If uncertain genotypes (e.g. as a result of imputation) are used, the interpretation of the regression coefficients is questionable.

A known bug is that the function fails when no `data` argument is supplied and the base model formula contains no variables (`~1`). A work-round is to create a data frame to hold the variables in the models and pass this as `data=`.

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

`GlmEstimates-class`, `snp.lhs.tests`

Examples

```
data(testdata)
test1 <-
snp.lhs.estimates(Autosomes[,1:10], ~cc, ~region, data=subject.data)
test2 <-
snp.lhs.estimates(Autosomes[,1:10], ~strata(region), ~cc,
   data=subject.data)
test3 <-
snp.lhs.estimates(Autosomes[,1:10], ~cc, ~region, data=subject.data, robust=TRUE)
test4 <-
snp.lhs.estimates(Autosomes[,1:10], ~strata(region), ~cc,
   data=subject.data, robust=TRUE)
```

```
test5 <- snp.lhs.estimates(Autosomes[,1:10], ~region+sex, ~cc, data=subject.data, robust=TRUE)
print(test1)
print(test2)
print(test3)
print(test4)
print(test5)
```

`snp.lhs.tests` *Score tests with SNP genotypes as dependent variable*

Description

Under the assumption of Hardy-Weinberg equilibrium, a SNP genotype is a binomial variate with two trials for an autosomal SNP or with one or two trials (depending on sex) for a SNP on the X chromosome. With each SNP in an input `"SnpMatrix"` as dependent variable, this function first fits a "base" logistic regression model and then carries out a score test for the addition of further term(s). The Hardy-Weinberg assumption can be relaxed by use of a "robust" option.

Usage

```
snp.lhs.tests(snp.data, base.formula, add.formula, subset, snp.subset,
                data = sys.parent(), robust = FALSE, uncertain = FALSE,
                control=glm.test.control(), score=FALSE)
```

Arguments

`snp.data`	The SNP data, as an object of class `"SnpMatrix"` or `"XSnpMatrix"`
`base.formula`	A `formula` object describing the base model, with dependent variable omitted
`add.formula`	A `formula` object describing the additional terms to be tested, also with dependent variable omitted
`subset`	An array describing the subset of observations to be considered
`snp.subset`	An array describing the subset of SNPs to be considered. Default action is to test all SNPs.
`data`	The data frame in which `base.formula`, `add.formula` and `subset` are to be evaluated
`robust`	If `TRUE`, a test which does not assume Hardy-Weinberg equilibrium will be used
`uncertain`	If `TRUE`, uncertain genotypes are used and scored by their posterior expectations. Otherwise they are treated as missing. If set, this option forces `robust` variance estimates
`control`	An object giving parameters for the IRLS algorithm fitting of the base model and for the acceptable aliasing amongst new terms to be tested. See `glm.test.control`
`score`	Is extended score information to be returned?

Details

The tests used are asymptotic chi-squared tests based on the vector of first and second derivatives of the log-likelihood with respect to the parameters of the additional model. The "robust" form is a generalized score test in the sense discussed by Boos(1992). If a `data` argument is supplied, the `snp.data` and `data` objects are aligned by rowname. Otherwise all variables in the model formulae are assumed to be stored in the same order as the columns of the `snp.data` object.

Value

An object of class `snp.tests.glm` or `GlmTests.score` depending on whether `score` is set to `FALSE` or `TRUE` in the call.

Note

A factor (or several factors) may be included as arguments to the function `strata(...)` in the `base.formula`. This fits all interactions of the factors so included, but leads to faster computation than fitting these in the normal way. Additionally, a `cluster(...)` call may be included in the base model formula. This identifies clusters of potentially correlated observations (e.g. for members of the same family); in this case, an appropriate robust estimate of the variance of the score test is used. No more than one `strata()` call may be used, and neither `strata(...)` or `cluster(...)` calls may appear in the `add.formula`. A known bug is that the function fails when no `data` argument is supplied and the base model formula contains no variables (`~1`). A work-round is to create a data frame to hold the variables in the models and pass this as `data=`.

Author(s)

David Clayton <dc208@cam.ac.uk>

References

Boos, Dennis D. (1992) On generalized score tests. *The American Statistician*, **46**:327-333.

See Also

`GlmTests-class`, `GlmTestsScore-class`, `glm.test.control`,`snp.rhs.tests` `single.snp.tests`, `SnpMatrix-class`, `XSnpMatrix-class`

Examples

```
data(testdata)
snp.lhs.tests(Autosomes[,1:10], ~cc, ~region, data=subject.data)
snp.lhs.tests(Autosomes[,1:10], ~strata(region), ~cc,
   data=subject.data)
```

`snp.pre.multiply` *Pre- or post-multiply a SnpMatrix object by a general matrix*

Description

These functions first standardize the input `SnpMatrix` in the same way as does the function `xxt`. The standardized matrix is then either pre-multiplied (`snp.pre.multiply`) or post-multiplied (`snp.post.multiply`) by a general matrix. Allele frequencies for standardizing the input SnpMatrix may be supplied but, otherwise, are calculated from the input SnpMatrix

Usage

```
snp.pre.multiply(snps, mat, frequency=NULL, uncertain = FALSE)
snp.post.multiply(snps, mat, frequency=NULL, uncertain = FALSE)
```

Arguments

`snps`	An object of class `"SnpMatrix"` or `"XSnpMatrix"`
`mat`	A general (numeric) matrix
`frequency`	A numeric vector giving the allele (relative) frequencies to be used for standardizing the columns of `snps`. If `NULL`, allele frequencies will be calculated internally. Frequencies should refer to the second (`B`) allele
`uncertain`	If `TRUE`, uncertain genotypes are replaced by posterior expectations. Otherwise these are treated as missing values

Details

The two matrices must be conformant, as with standard matrix multiplication. The main use envisaged for these functions is the calculation of factor loadings in principal component analyses of large scale SNP data, and the application of these loadings to other datasets. The use of externally supplied allele frequencies for standardizing the input SnpMatrix is required when applying loadings calculated from one dataset to a different dataset

Value

The resulting matrix product

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

`xxt`

Examples

```
##--
##-- Calculate first two principal components and their loading, and verify
##--
# Make a SnpMatrix with a small number of rows
data(testdata)
small <- Autosomes[1:20,]
# Calculate the X.X-transpose matrix
xx <- xxt(small, correct.for.missing=FALSE)
# Calculate the first two principal components and corresponding eigenvalues
eigvv <- eigen(xx, symmetric=TRUE)
pc <- eigvv$vectors[,1:2]
ev <- eigvv$values[1:2]
# Calculate loadings for first two principal components
Dinv <- diag(1/sqrt(ev))
loadings <- snp.pre.multiply(small,  Dinv %*% t(pc))
# Now apply loadings back to recalculate the principal components
pc.again <- snp.post.multiply(small, t(loadings) %*% Dinv)
print(cbind(pc, pc.again))
```

---

snp.rhs.estimates *Fit GLMs with SNP genotypes as independent variable(s)*

Description

This function fits a generalized linear model with phenotype as dependent variable and with a series of SNPs (or small sets of SNPs) as predictor variables. Optionally, one or more potential confounders of a phenotype-genotype association may be included in the model. In order to protect against misspecification of the variance function, "robust" estimates of the variance-covariance matrix of estimates may be calculated in place of the usual model-based estimates.

Usage

```
snp.rhs.estimates(formula, family = "binomial", link, weights, subset,
data = parent.frame(), snp.data,
   rules = NULL, sets = NULL, robust = FALSE, uncertain = FALSE, control
= glm.test.control())
```

Arguments

formula	The model formula, with phenotype as dependent variable and any potential confounders as independent variables. Note that parameter estimates are not returned for these model terms
family	A string defining the generalized linear model family. This currently should (partially) match one of "binomial", "Poisson", "Gaussian" or "gamma" (case-insensitive)

link	A string defining the link function for the GLM. This currently should (partially) match one of "logit", "log", "identity" or "inverse". The default action is to use the "canonical" link for the family selected
data	The dataframe in which the model formula is to be interpreted
snp.data	An object of class "SnpMatrix" or "XSnpMatrix" containing the SNP data
rules	Optionally, an object of class "ImputationRules"
sets	Either a vector of SNP names (or numbers) for the SNPs to be added to the model formula, or a logical vector of length equal to the number of columns in snp.data or a list of short vectors defining sets of SNPs to be included (see Details)
weights	"Prior" weights in the generalized linear model
subset	Array defining the subset of rows of data to use
robust	If TRUE, robust tests will be carried out
uncertain	If TRUE, uncertain genotypes are used and scored by their posterior expectations. Otherwise they are treated as missing
control	An object giving parameters for the IRLS algorithm fitting of the base model and for the acceptable aliasing amongst new terms to be tested. See glm.test.control

Details

Homozygous SNP genotypes are coded 0 or 2 and heterozygous genotypes are coded 1. For SNPs on the X chromosome, males are coded as homozygous females. For X SNPs, it will often be appropriate to include sex of subject in the base model (this is not done automatically). The "robust" option causes Huber-White estimates of the variance-covariance matrix of the parameter estimates to be returned. These protect against mis-specification of the variance function in the GLM, for example if binary or count data are overdispersed,

If a data argument is supplied, the snp.data and data objects are aligned by rowname. Otherwise all variables in the model formulae are assumed to be stored in the same order as the columns of the snp.data object.

Usually SNPs to be fitted in models will be referenced by name. However, they can also be referenced by number, indicating the appropriate column in the input snp.data. They can also be referenced by a logical selection vector of length equal to the number of columns in snp.data.

If the rules argument is supplied, SNPs may be imputed using these rules and included in the model.

Value

An object of class GlmEstimates

Note

A factor (or several factors) may be included as arguments to the function strata(...) in the formula. This fits all interactions of the factors so included, but leads to faster computation than fitting these in the normal way. Additionally, a cluster(...) call may be included in the base model formula. This identifies clusters of potentially correlated observations (e.g. for members

of the same family); in this case, an appropriate robust estimate of the variance of the parameter estimates is used.

If uncertain genotypes (e.g. as a result of imputation) are used, the interpretation of the regression coefficients is questionable; the regression coefficient for an imperfectly measurement of a variable is not a biased (attenuated) estimate of the coefficient of the variable measured.

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

GlmEstimates-class, snp.lhs.estimates, snp.rhs.tests, SnpMatrix-class, XSnpMatrix-class

Examples

```
data(testdata)
test <- snp.rhs.estimates(cc~strata(region), family="binomial",
   data=subject.data, snp.data= Autosomes, sets=1:10)
print(test)
test2 <- snp.rhs.estimates(cc~region+sex, family="binomial",
   data=subject.data, snp.data= Autosomes, sets=1:10)
print(test2)
test.robust <- snp.rhs.estimates(cc~strata(region), family="binomial",
   data=subject.data, snp.data= Autosomes, sets=1:10, robust=TRUE)
print(test.robust)
```

---

snp.rhs.tests *Score tests with SNP genotypes as independent variable*

Description

This function fits a generalized linear model with phenotype as dependent variable and, optionally, one or more potential confounders of a phenotype-genotype association as independent variable. A series of SNPs (or small groups of SNPs) are then tested for additional association with phenotype. In order to protect against misspecification of the variance function, "robust" tests may be selected.

Usage

```
snp.rhs.tests(formula, family = "binomial", link, weights, subset, data = parent.frame(),
   snp.data, rules=NULL, tests=NULL, robust = FALSE, uncertain=FALSE,
   control=glm.test.control(), allow.missing=0.01, score=FALSE)
```

Arguments

formula	The base model formula, with phenotype as dependent variable
family	A string defining the generalized linear model family. This currently should (partially) match one of "binomial", "Poisson", "Gaussian" or "gamma" (case-insensitive)
link	A string defining the link function for the GLM. This currently should (partially) match one of "logit", "log", "identity" or "inverse". The default action is to use the "canonical" link for the family selected
data	The dataframe in which the base model is to be fitted
snp.data	An object of class "SnpMatrix" or "XSnpMatrix" containing the SNP data
rules	An object of class "ImputationRules". If supplied, the rules coded in this object are used, together with snp.data, to calculate tests for imputed SNPs
tests	Either a vector of SNP names (or numbers) for the SNPs to be tested, or a logical vector of length equal to the number of columns in snp.data, or a list of short numeric or character vectors defining groups of SNPs to be tested (see Details)
weights	"Prior" weights in the generalized linear model
subset	Array defining the subset of rows of data to use
robust	If TRUE, robust tests will be carried out
uncertain	If TRUE, uncertain genotypes are used and scored by their posterior expectations. Otherwise they are treated as missing
control	An object giving parameters for the IRLS algorithm fitting of the base model and for the acceptable aliasing amongst new terms to be tested. See glm.test.control
allow.missing	The maximum proportion of SNP genotype that can be missing before it becomes necessary to refit the base model
score	Is extended score information to be returned?

Details

The tests used are asymptotic chi-squared tests based on the vector of first and second derivatives of the log-likelihood with respect to the parameters of the additional model. The "robust" form is a generalized score test in the sense discussed by Boos(1992). The "base" model is first fitted, and a score test is performed for addition of one or more SNP genotypes to the model. Homozygous SNP genotypes are coded 0 or 2 and heterozygous genotypes are coded 1. For SNPs on the X chromosome, males are coded as homozygous females. For X SNPs, it will often be appropriate to include sex of subject in the base model (this is not done automatically).

If a data argument is supplied, the snp.data and data objects are aligned by rowname. Otherwise all variables in the model formulae are assumed to be stored in the same order as the columns of the snp.data object.

Usually SNPs to be used in tests will be referenced by name. However, they can also be referenced by number, a positive number indicating the appropriate column in the input snp.data, and a negative number indicating (minus) a position in the rules list. They can also be referenced by a logical selection vector of length equal to the number of columns in snp.data. Sets of tests involving more than one SNP are referenced by a list and can use a mixture of observed and imputed

SNPs. If the tests argument is missing, single SNP tests are carried out; if a rules is given, all *imputed* SNP tests are calculated, otherwise all SNPs in the input snp.data matrix are tested. But note that, for single SNP tests, the function single.snp.tests will often achieve the same result much faster.

Value

An object of class GlmTests or GlmTestsScore depending on whether score is set to FALSE or TRUE in the call.

Note

A factor (or several factors) may be included as arguments to the function strata(...) in the formula. This fits all interactions of the factors so included, but leads to faster computation than fitting these in the normal way. Additionally, a cluster(...) call may be included in the base model formula. This identifies clusters of potentially correlated observations (e.g. for members of the same family); in this case, an appropriate robust estimate of the variance of the score test is used.

Author(s)

David Clayton <dc208@cam.ac.uk>

References

Boos, Dennis D. (1992) On generalized score tests. *The American Statistician*, **46**:327-333.

See Also

GlmTests-class, GlmTestsScore-class, single.snp.tests, snp.lhs.tests, impute.snps, ImputationRules-class, SnpMatrix-class, XSnpMatrix-class

Examples

```
data(testdata)
slt3 <- snp.rhs.tests(cc~strata(region), family="binomial",
   data=subject.data, snp.data= Autosomes, tests=1:10)
print(slt3)
```

SnpMatrix-class *Class "SnpMatrix"*

Description

This class defines objects holding large arrays of single nucleotide polymorphism (SNP) genotypes generated using array technologies.

Objects from the Class

Objects can be created by calls of the form `new("SnpMatrix", x)` where `x` is a matrix with storage mode `"raw"`. Chips (usually corresponding to samples or subjects) define rows of the matrix while polymorphisms (loci) define columns. Rows and columns will usually have names which can be used to link the data to further data concerning samples and SNPs

Slots

`.Data`**:** Object of class `"matrix"` and storage mode `raw` Internally, missing data are coded 0 and SNP genotypes are coded 1, 2 or 3. Imputed values may not be known exactly. Such uncertain calls are grouped by probability and represented by codes 4 to 253

Extends

Class `"matrix"`, from data part. Class `"structure"`, by class `"matrix"`. Class `"array"`, by class `"matrix"`. Class `"vector"`, by class "matrix", with explicit coerce. Class `"vector"`, by class "matrix", with explicit coerce.

Methods

[ ] `signature(x = "SnpMatrix", i = "ANY", j = "ANY",drop = "missing")`: subset operations

cbind2 `signature(x = "SnpMatrix", y = "SnpMatrix")`: S4 generic function to provide cbind() for two or more matrices together by column. Row names must match and column names must not coincide. If the matrices are of the derived class `XSnpMatrix-class`, the `diploid` slot values must also agree

coerce `signature(from = "SnpMatrix", to = "numeric")`: map to numeric values 0, 1, 2 or, for uncertain assignments, to the posterior expectation of the 0, 1, 2 code

coerce `signature(from = "SnpMatrix", to = "character")`: map to codes "A/A", "A/B", "B/B", ""

coerce `signature(from = "matrix", to = "SnpMatrix")`: maps numeric matrix (coded 0, 1, 2 or NA) to a SnpMatrix

coerce `signature(from = "SnpMatrix", to = "XSnpMatrix")`: maps a SnpMatrix to an XSnpMatrix. Ploidy is inferred from the genotype data since haploid genotypes should always be coded as homozygous. After inferring ploidy, heterozygous calls for haploid genotypes are set to `NA`

is.na `signature(x = "SnpMatrix")`: returns a logical matrix indicating whether each element is `NA`

rbind2 `signature(x = "SnpMatrix", y = "snp.matrix")`: S4 generic function to provide rbind() for two or more matrices by row. Column names must match and duplicated row names prompt warnings

show `signature(object = "SnpMatrix")`: shows the size of the matrix (since most objects will be too large to show in full)

summary `signature(object = "SnpMatrix")`: returns summaries of the data frames returned by `row.summary` and `col.summary`

is.na `signature(x = "SnpMatrix")`: returns a logical matrix of missing call indicators

switch.alleles `signature(x = "SnpMatrix", snps ="ANY")`: Recode specified columns of of the matrix to reflect allele switches

Note

This class requires at least version 2.3 of R

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

XSnpMatrix-class

Examples

```
data(testdata)
summary(Autosomes)

# Just making it up - 3-10 will be made into NA during conversion
snps.class<-new("SnpMatrix", matrix(1:10))
snps.class
if(!isS4(snps.class)) stop("constructor is not working")

pretend.X <- as(Autosomes, 'XSnpMatrix')
if(!isS4(pretend.X)) stop("coersion to derived class is not S4")
if(class(pretend.X) != 'XSnpMatrix') stop("coersion to derived class is not working")

pretend.A <- as(Xchromosome, 'SnpMatrix')
if(!isS4(pretend.A)) stop("coersion to base class is not S4")
if(class(pretend.A) != 'SnpMatrix') stop("coersion to base class is not working")

# display the first 10 snps of the first 10 samples
print(as(Autosomes[1:10,1:10], 'character'))

# convert the empty strings (no-calls) explicitly to "NC" before
# writing to an (anonymous and temporary) csv file
csvfile <- tempfile()
write.csv(file=csvfile, gsub ('^$', 'NC',
                               as(Autosomes[1:10,1:10], 'character')
                              ), quote=FALSE)
unlink(csvfile)
```

---

switch.alleles *Switch alleles in columns of a SnpMatrix or in test results*

Description

This is a generic function which can be applied to objects of class "SnpMatrix" or "XSnpMatrix" (which hold SNP genotype data), or to objects of class "SingleSnpTestsScore" or "GlmTests" (which hold association test results). In the former case, specified SNPs can be recoded as if the alleles were switched (so that *AA* genotypes become *BB* and vice-versa while *AB* remain unchanged). In the latter case, test results are modified *as if* alleles had been switched.

Usage

```
switch.alleles(x, snps)
```

Arguments

x	The input object, of class "SnpMatrix", "XSnpMatrix", "SingleSnpTestsScore", or "GlmTests"
snps	A vector of type integer, character or logical specifying the SNP to have its alleles switched

Value

An object of the same class as the input object

Note

Switching alleles for SNPs has no effect on test results. These functions are required when carrying out meta-analysis, bringing together several sets of results. It is then important that alleles line up in the datasets to be combined. It is often more convenient (and faster) to apply this process to the test result objects rather than to the genotype data themselves.

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

SnpMatrix-class, XSnpMatrix-class, SingleSnpTests-class, GlmTests-class

Examples

```
data(testdata)
which <-  c("173774", "173811")
Asw <- switch.alleles(Autosomes, which)
col.summary(Autosomes[,which])
col.summary(Asw[,which])
```

---

tdt.snp *1-df and 2-df tests for genetic associations with SNPs (or imputed SNPs) in family data*

Description

Given large-scale SNP data for families comprising both parents and one or more affected offspring, this function computes 1 df tests (the TDT test) and a 2 df test based on observed and expected transmissions of genotypes. Tests based on imputation rules can also be carried out.

Usage

```
tdt.snp(ped, id, father, mother, affected, data = sys.parent(), snp.data,
    rules = NULL, snp.subset, check.inheritance = TRUE, robust = FALSE,
    uncertain = FALSE, score = FALSE)
```

Arguments

ped	Pedigree identifiers
id	Subject identifiers
father	Identifiers for subjects' fathers
mother	Identifiers for subjects' mothers
affected	Disease status (TRUE if affected, FALSE otherwise)
data	A data frame in which to evaluate the previous five arguments
snp.data	An object of class "SnpMatrix" containing the SNP genotypes to be tested
rules	An object of class "ImputationRules". If supplied, the rules coded in this object are used, together with snp.data, to calculate tests for imputed SNPs
snp.subset	A vector describing the subset of SNPs to be considered. Default action is to test all SNPs in snp.data or, in imputation mode, as specified by rules
check.inheritance	If TRUE, each affected offspring/parent trio is tested for Mendelian inheritance and excluded if the test fails. If FALSE, misinheriting trios are used but the "robust" variance option is forced
robust	If TRUE, forces the robust (Huber-White) variance option (with ped determining independent "clusters")
uncertain	If TRUE, uncertain genotypes are handed by replacing score contributions by their posterior expectations. Otherwise these are treated as missing. Setting this option authomatically invokes use of robust variance estimates
score	If TRUE, the output object will contain, for each SNP, the score vector and its variance-covariance matrix

Details

Formally, the test statistics are score tests for the "conditioning on parental genotype" (CPG) likelihood. Parametrization of associations is the same as for the population-based tests calculated by single.snp.tests so that results from family-based and population-based studies can be combined using pool.

When the function is used to calculate tests for imputed SNPs, the test is still an approximate score test. The current version does not use the family relationships in the imputation. With this option, the robust variance estimate is forced.

The first five arguments are usually derived from a "pedfile". If a data frame is supplied for the data argument, the first five arguments will be evaluated in this frame. Otherwise they will be evaluated in the calling environment. If the arguments are missing, they will be assumed to be in their usual positions in the pedfile data frame i.e. in columns one to four for the identifiers and column six for disease status (with affected coded 2). If the pedfile data are obtained from a dataframe, the row

names of the data and snp.data files will be used to align the pedfile and SNP data. Otherwise, these vectors will be assumed to be in the same order as the rows of snp.data.

The snp.subset argument can be a logical, integer, or character vector.

If imputed rather than observed SNPs are tested, or if check.inheritance is set to FALSE, the "robust" variance estimate is used regardless of the value supplied for the robust argument.

Value

An object of class "SingleSnpTests". If score=TRUE, the output object will be of the extended class "SingleSnpTestsScore" containing additional slots holding the score statistics and their variances (and covariances). This allows meta-analysis using the pool function.

Note

When the snps are on the X chromosome (i.e. when the snp.data argument is of class "XSnpMatrix"), the tests are constructed in the same way as was described by Clayton (2008) for population-based association tests i.e. assuming that genotype relative risks for males mirror thos of homozygous females

Author(s)

David Clayton <dc208@cam.ac.uk>

References

Clayton (2008) Testing for association on the X chromosome *Biostatistics*, **9**:593-600.)

See Also

single.snp.tests, impute.snps, pool, ImputationRules-class, SingleSnpTests-class, SingleSnpTestsScore-cla

Examples

```
data(families)
tdt.snp(data=pedData, snp.data=genotypes)
```

---

test.allele.switch *Test for switch of alleles between two collections*

Description

When testing genotype data derived from different platforms or scoring algorithms a common problem is switching of alleles. This function provides a diagnostic for this. Input can either be two objects of class "SnpMatrix" to be examined, column by column, for allele switching, or a single "SnpMatrix" object together with an indicator vector giving group membership for its rows.

Usage

```
test.allele.switch(snps, snps2 = NULL, split = NULL, prior.df = 1)
```

Arguments

snps	An object of class "SnpMatrix" or "XSnpMatrix"
snps2	A second object of the same class as snps
split	If only one SnpMatrix object supplied, a vector with the same number of elements as rows of snps. It must be capable of coercion to a factor with two levels.
prior.df	A degree of freedom parameter for the prior distribution of the allele frequency prior.df (see Details)

Details

This function calculates a Bayes factor for the comparison of the hypothesis that the alleles have been switched with the hypothesis that they have not been switched. This requires integration over the posterior distribution of the allele frequency. The prior is taken as a beta distribution with both parameters equal to prior.dfso that the prior is symmetric about 0.5. The default, prior.df=1 represents a uniform prior on (0,1).

Value

A vector containing the log (base 10) of the Bayes Factors for an allele switch.

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

SnpMatrix-class, XSnpMatrix-class

Examples

```
data(testdata)
#
# Call with two SnpMatrix arguments
#
cc <- as.numeric(subject.data$cc)
lbf1 <- test.allele.switch(Autosomes[cc==1,], Autosomes[cc==2,])
#
# Single matrix call (giving the same result)
#
lbf2 <- test.allele.switch(Autosomes, split=cc)
```

`testdata` *Test data for the snpStats package*

Description

This dataset comprises several data frames from a fictional (and unrealistically small) study. The dataset started off as real data from a screen of non-synonymous SNPs for association with type 1 diabetes, but the original identifiers have been removed and a random case/control status has been generated.

Usage

```
data(testdata)
```

Format

There are five data objects in the dataset:

- `Autosomes`: An object of class `"SnpMatrix"` containing genotype calls for 400 subjects at 9445 autosomal SNPs
- `Xchromosome`: An object of class `"XSnpMatrix"` containing genotype calls for 400 subjects at 155 SNPs on the X chromosome
- `Asnps`: A dataframe containing information about the autosomal SNPs. Here it contains only one variable, `chromosome`, indicating the chromosomes on which the SNPs are located
- `Xsnps`: A dataframe containing information about the X chromosome SNPs. Here it is empty and is only included for completeness
- `subject.data`: A dataframe containing information about the subjects from whom each row of SNP data was obtained. Here it contains:
  - `cc`: Case-control status
  - `sex`: Sex
  - `region`: Geographical region of residence

Source

The data were obtained from the diabetes and inflammation laboratory (see http://www-gene.cimr.cam.ac.uk)

Examples

```
data(testdata)
Autosomes
Xchromosome
summary(Asnps)
summary(Xsnps)
summary(subject.data)
summary(summary(Autosomes))
summary(summary(Xchromosome))
```

write.plink *Write files for analysis in the PLINK toolset*

Description

Given a SnpMatrix object, together with associated subject and SNP support dataframes, this function writes .bed, .bim, and .fam files for processing in the PLINK toolset

Usage

```
write.plink(file.base, snp.major = TRUE, snps,
   subject.data, pedigree, id, father, mother, sex, phenotype,
   snp.data, chromosome, genetic.distance, position, allele.1, allele.2,
   na.code = 0, human.genome=TRUE)
```

Arguments

file.base	A character string giving the base filename. The extensions .bed, .bim, and .fam are appended to this string to give the filenames of the three output files
snp.major	Logical variable controlling whether the .bed file is in SNP-major or subject-major order
snps	The SnpMatrix or XSnpMatrix object to be written out
subject.data	(Optional) A subject support dataframe. If supplied, the next six arguments (which define the fields of the PLINK .fam file) will be evaluated in this environment, after matching row names with the row names of snps. Otherwise they will be evaluated in the calling environment; they then must be of the right length and in the correct order.
pedigree	A pedigree (family) identifier. Default is the row names of snps.
id	An identifier of an individual within family. Default is a vector of na.code.
father	The within-family identifier of the subject's father. Default is a vector of na.code.
mother	The within-family identifier of the subject's mother. Default is a vector of na.code.
sex	Sex of the individual. Default is a vector of na.code. This will be coerced to type numeric.
phenotype	The primary phenotype value. Default is a vector of na.code. This will be coerced to type numeric.
snp.data	(Optional) A SNP support dataframe. If supplied, the next five arguments (which define the columns of the PLINK .bim file) will be evaluated in this environment, after matching row names with the column names of snps. Otherwise they will be evaluated in the calling environment; they then must be of the right length and in the correct order.
chromosome	The chromosome on which the SNP is located. This should either be numeric, as specified by SPLINK, or character, with "X", "Y", "XY", and "MT" for the non-numeric values. Default is a vector of na.code, or a vector of 23's if snps is a XSnpMatrix.

genetic.distance
: The location of the SNP, expressed as a genetic distance. Default is a vector of na.code. This will be coerced to type numeric.

position
: The physical location of the SNP, expressed in base pairs. Default is a vector of na.code. This will be coerced to type numeric.

allele.1
: A character vector giving the first allele. Default is a vector of "A"s.

allele.2
: A character vector giving the first allele. Default is a vector of "B"s.

na.code
: The code to be written for NA in the .fam and .bin output files. It should be numeric (or capable of coercion to numeric).

human.genome
: If TRUE, check the chromosome argument for valid values.

Details

For more details of required codings in .fam and .bin files, see the PLINK documentation.

Value

Returns NULL.

Author(s)

David Clayton <dc208@cam.ac.uk>

References

PLINK: Whole genome association analysis toolset. http://pngu.mgh.harvard.edu/~purcell/plink/

See Also

read.plink, SnpMatrix-class, XSnpMatrix-class

Examples

```
data(testdata)
## the use of as.numeric() below is not necessary since this is done
## automatically. It just illustrates use of expressions for these arguments
## Note that cc and sex are variables within the subject.data frame
## This command writes files test.bed. test.fam and test.bim
write.plink(file.base="test", snps=Autosomes,
    subject.data=subject.data, phenotype = as.numeric(cc), sex=as.numeric(sex),
    snp.major=FALSE)
```

`write.SnpMatrix` *Write a SnpMatrix object as a text file*

Description

This function is closely modelled on `write.table`. It writes an object of class `SnpMatrix` as a text file with one line for each row of the matrix. Genotpyes are written in numerical form, *i.e.* as 0, 1 or 2 (where 1 denotes heterozygous) or, optionally, as a pair of alleles (each coded 1 or 2).

Usage

```
write.SnpMatrix(x, file, as.alleles= FALSE, append = FALSE, quote = TRUE, sep = " ", eol = "\n", na = "NA"
```

Arguments

`x`	The object to be written
`file`	The name of the output file
`as.alleles`	If `TRUE`, write each genotype as two alleles
`append`	If `TRUE`, the output is appended to the designated file. Otherwise a new file is opened
`quote`	If `TRUE`, row and column names will be enclosed in quotes
`sep`	The string separating entries within a line
`eol`	The string terminating each line
`na`	The string written for missing genotypes
`row.names`	If `TRUE`, each row will commence with the row name
`col.names`	If `TRUE`, the first line will contain all the column names

Value

A numeric vector giving the dimensions of the matrix written

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

`write.table`, `SnpMatrix-class`, `XSnpMatrix-class`

XSnpMatrix-class *Class "XSnpMatrix"*

Description

This class extends the SnpMatrix-class to deal with SNPs on the X and Y chromosomes and mitocondrial SNPs.

Objects from the Class

Objects can be created by calls of the form `new("XSnpMatrix", x, diploid)`. Such objects have an additional `slot` to objects of class `"SnpMatrix"` consisting of a logical array of the same length as the number of rows. This array indicates whether genotypes in that row are diploid (`TRUE`) or haploid (`FALSE` as, for example, SNPs on the X chromosome for males).

Slots

`.Data`**:** Object of class `"matrix"` and storage mode `"raw"`

`diploid`**:** Object of class `"logical"` indicating sex of samples

Extends

Class `"SnpMatrix"`, directly, with explicit coerce. Class `"matrix"`, by class `"SnpMatrix"`. Class `"structure"`, by class `"SnpMatrix"`. Class `"array"`, by class `"SnpMatrix"`. Class `"vector"`, by class "SnpMatrix", with explicit coerce. Class `"vector"`, by class "SnpMatrix", with explicit coerce.

Methods

[ ] `signature(x = "XSnpMatrix", i = "ANY", j = "ANY",drop = "missing")`: subset extraction

[<- `signature(x = "XSnpMatrix", i = "ANY", j = "ANY", "XSnpMatrix")`: subset assignment operation to replace part of an object

coerce `signature(from = "XSnpMatrix", to = "character")`: map to codes 0, 1, 2, or NA

coerce `signature(from = "SnpMatrix", to = "XSnpMatrix")`: maps a SnpMatrix to an XSnpMatrix. Ploidy is inferred from the genotype data since haploid genotypes should always be coded as homozygous. After inferring ploidy, heterozygous calls for haploid genotpes are set to `NA`

show `signature(object = "XSnpMatrix")`: map to codes "A/A", "A/B", "B/B", "A", "B" or ""

summary `signature(object = "XSnpMatrix")`: returns the distribution of ploidy, together with summaries of the data frames returned by `row.summary` and `col.summary`

Author(s)

David Clayton <dc208@cam.ac.uk>

See Also

SnpMatrix-class

Examples

```
data(testdata)
summary(Xchromosome)

# display the first 10 snps of the first 10 samples
print(as(Xchromosome[1:10,1:10],'character'))

# convert the empty strings (no-calls) explicitly to "NC" before
# writing to an (anonymous and temporary) csv file
csvfile <- tempfile()
write.csv(file=csvfile, gsub ('^$', 'NC',
                               as(Xchromosome[1:10,1:10], 'character')
                              ), quote=FALSE)
unlink(csvfile)
```

---

xxt *X.X-transpose for a standardized SnpMatrix*

Description

The input SnpMatrix is first standardized by subtracting the mean (or stratum mean) from each call and dividing by the expected standard deviation under Hardy-Weinberg equilibrium. It is then post-multiplied by its transpose. This is a preliminary step in the computation of principal components.

Usage

```
xxt(snps, strata = NULL, correct.for.missing = FALSE, lower.only = FALSE,
  uncertain = FALSE)
```

Arguments

snps	The input matrix, of type "SnpMatrix"
strata	A factor (or an object which can be coerced into a factor) with length equal to the number of rows of snps defining stratum membership
correct.for.missing	If TRUE, an attempt is made to correct for the effect of missing data by use of inverse probability weights. Otherwise, missing observations are scored zero in the standardized matrix
lower.only	If TRUE, only the lower triangle of the result is returned and the upper triangle is filled with zeros. Otherwise, the complete symmetric matrix is returned
uncertain	If TRUE, uncertain genotypes are replaced by posterior expectations. Otherwise these are treated as missing values

Details

This computation forms the first step of the calculation of principal components for genome-wide SNP data. As pointed out by Price et al. (2006), when the data matrix has more rows than columns it is most efficient to calculate the eigenvectors of $X.X$-transpose, where X is a `SnpMatrix` whose columns have been standardized to zero mean and unit variance. For autosomes, the genotypes are given codes 0, 1 or 2 after subtraction of the mean, $2p$, are divided by the standard deviation sqrt(2p(1-p)) (p is the estimated allele frequency). For SNPs on the X chromosome in male subjects, genotypes are coded 0 or 2. Then the mean is still $2p$, but the standard deviation is 2sqrt(p(1-p)). If the `strata` is supplied, a stratum-specific estimate value for p is used for standardization.

Missing observations present some difficulty. Price et al. (2006) recommended replacing missing observations by their means, this being equivalent to replacement by zeros in the standardized matrix. However this results in a biased estimate of the complete data result. Optionally this bias can be corrected by inverse probability weighting. We assume that the probability that any one call is missing is small, and can be predicted by a multiplicative model with row (subject) and column (locus) effects. The estimated probability of a missing value in a given row and column is then given by $m = RC/T$, where R is the row total number of no-calls, C is the column total of no-calls, and T is the overall total number of no-calls. Non-missing contributions to $X.X$-transpose are then weighted by $w = 1/(1 - m)$ for contributions to the diagonal elements, and products of the relevant pairs of weights for contributions to off–diagonal elements.

Value

A square matrix containing either the complete X.X-transpose matrix, or just its lower triangle

Warning

The correction for missing observations can result in an output matrix which is not positive semi-definite. This should not matter in the application for which it is intended

Note

In genome-wide studies, the SNP data will usually be held as a series of objects (of class `"SnpMatrix"` or`"XSnpMatrix"`), one per chromosome. Note that the $X.X$-transpose matrices produced by applying the `xxt` function to each object in turn can be added to yield the genome-wide result.

If the matrix is converted to a correlation matrix by pre- and post-multiplying by the sqrt of the inverse of its diagonal, then this is an unbiased estimate of twice the kinship matrix.

Author(s)

David Clayton <dc208@cam.ac.uk>

References

Price et al. (2006) Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics*, **38**:904-9

Examples

```
# make a SnpMatrix with a small number of rows
data(testdata)
small <- Autosomes[1:100,]
# Calculate the X.X-transpose matrix
xx <- xxt(small, correct.for.missing=TRUE)
# Calculate the principal components
pc <- eigen(xx, symmetric=TRUE)$vectors
```

Index

∗ **IO**
read.beagle, 29
read.impute, 30
read.long, 31
read.mach, 33
read.pedfile, 34
read.plink, 35
read.snps.long, 37
write.plink, 67
write.SnpMatrix, 69
∗ **array**
snp.cor, 46
snp.pre.multiply, 54
xxt, 71
∗ **classes**
convert.snpMatrix, 5
GlmEstimates-class, 11
GlmTests-class, 12
ImputationRules-class, 16
SingleSnpTests-class, 44
SnpMatrix-class, 59
XSnpMatrix-class, 70
∗ **cluster**
ibsCount, 13
ibsDist, 14
∗ **datasets**
example-new, 5
families, 6
for.exercise, 8
ld.example, 19
sample.ped.gz, 41
testdata, 66
∗ **file**
read.beagle, 29
read.impute, 30
read.long, 31
read.mach, 33
read.pedfile, 34
read.plink, 35
read.snps.long, 37
write.plink, 67
write.SnpMatrix, 69
∗ **hplot**
plotUncertainty, 23
qq.chisq, 26
∗ **htest**
mvtests, 22
pool, 24
pool2, 25
single.snp.tests, 42
snp.lhs.estimates, 50
snp.lhs.tests, 52
snp.rhs.estimates, 55
snp.rhs.tests, 57
tdt.snp, 62
∗ **manip**
imputation.maf, 15
misinherits, 21
read.long, 31
read.pedfile, 34
read.plink, 35
read.snps.long, 37
write.plink, 67
write.SnpMatrix, 69
∗ **misc**
ld, 18
∗ **models**
filter.rules, 7
impute.snps, 17
snp.imputation, 48
∗ **multivariate**
snp.cor, 46
snp.pre.multiply, 54
xxt, 71
∗ **package**
snpStats-package, 3
∗ **programming**
mean2g, 20

∗ **regression**
filter.rules, 7
impute.snps, 17
snp.imputation, 48
∗ **univar**
Fst, 9
∗ **utilities**
chi.squared, 3
glm.test.control, 10
random.snps, 28
read.long, 31
read.pedfile, 34
read.plink, 35
read.snps.long, 37
row.summary, 39
sm.compare, 45
switch.alleles, 61
test.allele.switch, 64
write.plink, 67
write.SnpMatrix, 69
[,GlmEstimates,ANY,missing,missing-method *(GlmEstimates-class)*, 11
[,GlmTests,ANY,missing,missing-method *(GlmTests-class)*, 12
[,GlmTestsScore,ANY,missing,missing-method *(GlmTests-class)*, 12
[,ImputationRules,ANY,missing,missing-method *(ImputationRules-class)*, 16
[,SingleSnpTests,ANY,missing,missing-method *(SingleSnpTests-class)*, 44
[,SingleSnpTestsScore,ANY,missing,missing-method *(SingleSnpTests-class)*, 44
[,SnpMatrix,ANY,ANY,ANY-method *(SnpMatrix-class)*, 59
[,XSnpMatrix,ANY,ANY,ANY-method *(XSnpMatrix-class)*, 70
[<-,XSnpMatrix,ANY,ANY,XSnpMatrix-method *(XSnpMatrix-class)*, 70

Asnps *(testdata)*, 66
Autosomes *(testdata)*, 66

can.impute *(imputation.maf)*, 15
cbind,SnpMatrix-method *(SnpMatrix-class)*, 59
cbind2,SnpMatrix,SnpMatrix-method *(SnpMatrix-class)*, 59
ceph.1mb *(ld.example)*, 19
chi.squared, 3
chi.squared,GlmTests,missing-method *(GlmTests-class)*, 12
chi.squared,SingleSnpTests,numeric-method *(SingleSnpTests-class)*, 44
coerce,GlmEstimates,GlmTests-method *(GlmEstimates-class)*, 11
coerce,GlmTests,data.frame-method *(GlmTests-class)*, 12
coerce,matrix,SnpMatrix-method *(SnpMatrix-class)*, 59
coerce,SingleSnpTests,data.frame-method *(SingleSnpTests-class)*, 44
coerce,SnpMatrix,character-method *(SnpMatrix-class)*, 59
coerce,SnpMatrix,numeric-method *(SnpMatrix-class)*, 59
coerce,SnpMatrix,XSnpMatrix-method *(XSnpMatrix-class)*, 70
coerce,XSnpMatrix,character-method *(XSnpMatrix-class)*, 70
col.summary, *60*, *70*
col.summary *(row.summary)*, 39
convert.snpMatrix, 5
cov, *48*

deg.freedom *(chi.squared)*, 3
deg.freedom,GlmTests-method *(GlmTests-class)*, 12
dist, *14*, *15*

effect.sign *(chi.squared)*, 3
effect.sign,GlmTests,logical-method *(GlmTests-class)*, 12
effect.sign,SingleSnpTestsScore,missing-method *(SingleSnpTests-class)*, 44
effective.sample.size *(chi.squared)*, 3
effective.sample.size,SingleSnpTests-method *(SingleSnpTests-class)*, 44
eigen, *47*
example-new, 5

families, 6
filter.rules, 7
for.exercise, 8
Fst, 9

g2post *(mean2g)*, 20
genotypes *(families)*, 6
glm.test.control, 10, *50*, *52*, *53*, *56*, *58*

GlmEstimates, *51*, *56*
GlmEstimates-class, 11
GlmTests, *59*
GlmTests-class, 12
GlmTests.score, *23*, *53*
GlmTestsScore, *59*
GlmTestsScore-class *(*GlmTests-class*)*, 12

ibsCount, 13, *14, 15*
ibsDist, *14*, 14
imputation.maf, 15, *49*
imputation.nsnp *(*imputation.maf*)*, 15
imputation.r2, *49*
imputation.r2 *(*imputation.maf*)*, 15
ImputationRules-class, 16
impute.snps, *16*, 17, *43*, *59*, *64*
initialize,SnpMatrix-method
*(*SnpMatrix-class*)*, 59
initialize,XSnpMatrix-method
*(*XSnpMatrix-class*)*, 70
is.na,SnpMatrix-method
*(*SnpMatrix-class*)*, 59

ld, 18
ld.example, 19
list, *11*

mach1.out.mlprob.gz *(*sample.ped.gz*)*, 41
Matrix, *18*
mean2g, 20
misinherits, 21
mvtests, 22

names,GlmTests-method *(*GlmTests-class*)*,
12
names,SingleSnpTests-method
*(*SingleSnpTests-class*)*, 44

p.value *(*chi.squared*)*, 3
p.value,GlmTests,missing-method
*(*GlmTests-class*)*, 12
p.value,SingleSnpTests,numeric-method
*(*SingleSnpTests-class*)*, 44
pedData *(*families*)*, 6
plot,ImputationRules,missing-method
*(*ImputationRules-class*)*, 16
plotUncertainty, 23
pool, 24, *25*, *43*, *45*, *63, 64*
pool2, *24*, 25
pool2,GlmTestsScore,GlmTestsScore,logical-method
*(*GlmTests-class*)*, 12
pool2,SingleSnpTestsScore,SingleSnpTestsScore,logical-meth
*(*SingleSnpTests-class*)*, 44
post2g *(*mean2g*)*, 20
pp, 26

qq.chisq, 26

random.snps, 28
rbind,SnpMatrix-method
*(*SnpMatrix-class*)*, 59
rbind2,SnpMatrix,SnpMatrix-method
*(*SnpMatrix-class*)*, 59
read.beagle, 29
read.impute, 30
read.long, 31
read.mach, 33
read.pedfile, 34
read.plink, 35, *39*, *68*
read.snps.long, 37
row.summary, 39, *60*, *70*

sample-long-alleles.gz *(*sample.ped.gz*)*,
41
sample-long.gz *(*sample.ped.gz*)*, 41
sample.bed *(*sample.ped.gz*)*, 41
sample.bim *(*sample.ped.gz*)*, 41
sample.fam *(*sample.ped.gz*)*, 41
sample.info *(*sample.ped.gz*)*, 41
sample.ped.gz, 41
sample.size *(*chi.squared*)*, 3
sample.size,GlmTests-method
*(*GlmTests-class*)*, 12
sample.size,SingleSnpTests-method
*(*SingleSnpTests-class*)*, 44
sapply, *4*
show,GlmEstimates-method
*(*GlmEstimates-class*)*, 11
show,GlmTests-method *(*GlmTests-class*)*,
12
show,ImputationRules-method
*(*ImputationRules-class*)*, 16
show,SingleSnpTests-method
*(*SingleSnpTests-class*)*, 44
show,SnpMatrix-method
*(*SnpMatrix-class*)*, 59
show,XSnpMatrix-method
*(*XSnpMatrix-class*)*, 70

single.snp.tests, *4*, *16*, *24, 25*, *28*, 42, *44, 45*, *53*, *59*, *63, 64*
SingleSnpTests, *13*
SingleSnpTests-class, 44
SingleSnpTests.score, *24, 25*
SingleSnpTestsScore, *13*
SingleSnpTestsScore-class *(*SingleSnpTests-class*)*, 44
sm.compare, 45
snp.cor, 46
snp.imputation, *7*, *15–17*, 48
snp.lhs.estimates, *11*, 50, *57*
snp.lhs.tests, *4*, *10–13*, *24, 25*, *28*, *43*, *51*, 52, *59*
snp.matrix, *6*
snp.post.multiply *(*snp.pre.multiply*)*, 54
snp.pre.multiply, 54
snp.rhs.estimates, *11*, 55
snp.rhs.tests, *4*, *10–13*, *24, 25*, *28*, *43*, *53*, *57*, 57
snp.support *(*for.exercise*)*, 8
snp.tests.glm, *23–25*, *53*
SnpMatrix-class, 59
snps.10 *(*for.exercise*)*, 8
snpStats *(*snpStats-package*)*, 3
snpStats-package, 3
subject.data *(*testdata*)*, 66
subject.support *(*for.exercise*)*, 8
summary,GlmTests-method *(*GlmTests-class*)*, 12
summary,ImputationRules-method *(*ImputationRules-class*)*, 16
summary,SingleSnpTests-method *(*SingleSnpTests-class*)*, 44
summary,SnpMatrix-method *(*SnpMatrix-class*)*, 59
summary,XSnpMatrix-method *(*XSnpMatrix-class*)*, 70
support.ld *(*ld.example*)*, 19
switch.alleles, 61
switch.alleles,GlmTestsScore,character-method *(*GlmTests-class*)*, 12
switch.alleles,SingleSnpTestsScore,ANY-method *(*SingleSnpTests-class*)*, 44
switch.alleles,SnpMatrix,ANY-method *(*SnpMatrix-class*)*, 59

tdt.snp, *22*, 62
test.allele.switch, 64
testdata, 66

vector, *11*

write.plink, *37*, 67
write.SnpMatrix, 69
write.table, *69*

Xchromosome *(*testdata*)*, 66
XSnpMatrix-class, 70
Xsnps *(*testdata*)*, 66
xxt, *47*, *54*, 71

yri.1mb *(*ld.example*)*, 19