Package 'snpMatrix' documentation

of

August 21, 2008

Title The snp.matrix and X.snp.matrix classes

Version 1.5.4

Date 2008-08-21

Author David Clayton <david.clayton@cimr.cam.ac.uk> and Hin-Tak Leung
<htl10@users.sourceforge.net>

Description Implements classes and some basic methods for large-scale SNP association studies

Maintainer David Clayton <david.clayton@cimr.cam.ac.uk>

Depends R(>= 2.3.0), survival, methods

Suggests hexbin

Enhances genetics

License GNU General Public Licence (GPLv3)

URL http://www-gene.cimr.cam.ac.uk/clayton/software/

Collate ld.with.R ss.R contingency.table.R glm_test.R ibs.stats.R imputation.R indata.R ld.snp.R
ld.with.R.eml.R misc.R outdata.R plink.R qc.R qq_chisq.R read.chiamo.R read.HapMap.R
read.snps.pedfile.R single.R tdt_single.R structure.R wtccc.sample.list.R wtccc.signals.R
xstuff.R zzz.R

LazyLoad yes

biocViews Microarray, SNPsAndGeneticVariability

R **topics documented:**

---

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

Description

Generic functions to extract chi-squared values and p-values from SNP association test objects

Usage

```
chi.squared(x, df)
p.value(x, df)
```

Arguments

x   Currently an object of class `"snp.tests.single"` or `"snp.tests.single.score"`

df   Either the numeric value 1 or 2

Details

Currently these function are only available for objects output by `single.snp.tests`. Eventually versions will also be provided for output objects produced by `snp.lhs.tests` and `snp.rhs.tests`.

Value

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

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

`snp.lhs.tests`, `snp.rhs.tests`, `snp.tests.single-class`, `snp.tests.single.score-class`

| epsout.ld.snp | *Function to write an eps file directly to visualize LD* |

Description

`epsout.ld.snp` takes an object of `snp.matrix` class and a given snp range and depth, draw a eps file to visualize the LD in the same color scheme as haploview's default view. It was the first prototype of this bunch of software. Also, it does not keep any pair-wise data in memory at all, and maybe more suitable where the actual pair-wise LD data is not needed.

Usage

```
epsout.ld.snp(snpdata, filename, start, end, depth, do.notes=FALSE)
```

Arguments

snpdata	An object of `snp.matrix` class with M samples of N snps
filename	The file name of the output, preferably ending with ".eps", but this rule not enforced
start	The index of the start of the range of interest. Should be between 1 and (N-1)
end	The index of the end of the range of interest. Should be between 2 and N.
depth	The depth or lag of pair-wise calculation. Should be between 1 and N-1
do.notes	Boolean for whether to generate pdf annotation-related code

Details

The functinality of this routine has since been split into a two-stage processes involving `ld.snp` which generates a `snp.dprime` object which contains the result of the pairwise LD calculation, and `plot.snp.dprime` (or the `plot` method of a `snp.dprime` object) which does the drawing.

Value

return nothing

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

References

Clayton, D.G. and Leung, Hin-Tak (2007) An R package for analysis of whole-genome association studies. *Human Heredity* **64**:45-51.
GSL (GNU Scientific Library) http://www.gnu.org/software/gsl/
The postscript language reference manual: http://www.adobe.com/products/postscript/pdfs/PLRM.pdf
The pdf specification: http://partners.adobe.com/public/developer/en/pdf/PDFReference16.pdf

See Also

[snp.dprime-class](), [ld.snp](), [plot.snp.dprime]()

Examples

```
#
data(testdata)
epsout.ld.snp(Autosomes, start=1, end=500, depth=50, filename="test.eps")
```

---

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 `"snp.reg.imputation"` 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 ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

[snp.reg.imputation-class](), [snp.imputation]()

Examples

```
# No example yet
```

---

for.exercise   *Data for exercise in use of the snpMatrix 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 "snp.matrix" 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/todd)

References

http://www-gene.cimr.cam.ac.uk/clayton

Examples

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

---

glm.test.control   *Set up control object for GLM tests*

Description

To carry out a score test for a GLM, we first fit a "base" model using the standard iteratively reweighted least squares (IRLS) algorithm and then carry out a score test for addition of further terms. This function sets various control parameters for this.

Usage

```
glm.test.control(maxit, epsilon, R2Max)
```

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 the "base" generalized linear model (GLM) before carrying out a score test for effect of adding new term(s). The maxit parameter sets the maximum number of iterations to be carried out, while the epsilon parameter sets the criterion for determining convergence. After fitting the base model, the new terms are added, but terms judged to be "aliased" are omitted. The method for determining aliasing is as follows (denoting the "design" matrix for the additional terms by Z):

Step 1 Regress each column of Z on the base model matrix, using the final GLM weights from the base model fit, and replace Z with the residuals from these regressions.

Step 2 Consider each column of the new Z matrix in turn, regressing it on the *previous* columns (again using the weights from the base model fit). If the proportion of the weighted sum of squares "explained" by this regression exceeds R2Max, the term is dropped and not included in the test,

The aim of this procedure to avoid wasting degrees of freedom on columns so strongly aliased that there is little power to detect their effect.

Value

Returns the parameters as a list in the expected order

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

snp.lhs.tests, snp.rhs.tests

---

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)
```

Arguments

snps   An input object of class `"snp.matrix"` or `"X.snp.matrix"`

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 upper triangle contains the total number of comparisons and the lower 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 `"snp.matrix"` or `"X.snp.matrix"`), 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 ⟨david.clayton@cimr.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 ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

`ibsCount`, `dist`

Examples

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

---

ibs.stats   *function to calculate the identity-by-state stats of a group of samples*

Description

Given a snp.matrix-class or a X.snp.matrix-class object with N samples, calculates some statistics about the relatedness of every pair of samples within.

Usage

```
ibs.stats(x)
```

Arguments

x   a [snp.matrix-class](#) or a [X.snp.matrix-class](#) object containing N samples

Details

No-calls are excluded from consideration here.

Value

A data.frame containing $N(N-1)/2$ rows, where the row names are the sample name pairs separated by a comma, and the columns are:

Count   count of identical calls, exclusing no-calls

Fraction   fraction of identical calls comparied to actual calls being made in both samples

Warning

In some applications, it may be preferable to subset a (random) selection of SNPs first - the calculation time increases as $N(N-1)M/2$. Typically for N = 800 samples and M = 3000 SNPs, the calculation time is about 1 minute. A full GWA scan could take hours, and quite unnecessary for simple applications such as checking for duplicate or related samples.

Note

This is mostly written to find mislabelled and/or duplicate samples.

Illumina indexes their SNPs in alphabetical order so the mitochondria SNPs comes first - for most purpose it is undesirable to use these SNPs for IBS purposes.

TODO: Worst-case S4 subsetting seems to make 2 copies of a large object, so one might want to subset before rbind(), etc; a future version of this routine may contain a built-in subsetting facility to work around that limitation.

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

Examples

```
data(testdata)
result <- ibs.stats(Autosomes[11:20,])
summary(result)
```

`impute.snps`   *Impute snps*

Description

Given SNPs stored in an object of class `"snpMatrix"` or `"X.snp.matrix"` and a set of imputation equations in as object of class `"snp.reg.imputation"`, this function calculates imputed values.

Usage

```
impute.snps(rules, snps, subset=NULL)
```

Arguments

`rules`	The imputation equations; an object of class `"snp.reg.imputation"`
`snps`	The object of class `"snpMatrix"` or `"X.snp.matrix"` containing the observed SNPs
`subset`	A vector describing the subset of subjects to be used. If `NULL` (default), then use all subjects

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.

Note

Because the imputation is based on a linear model, the imputed value may lie outside the range 0 to 2.

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

References

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

See Also

snp.imputation

Examples

```
# Remove 5 SNPs from a datset and derive imputation rules for them
library(snpMatrix)
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.snp   *Function to calculate pairwise D', r-squared*

Description

`ld.snp` takes an object of `snp.matrix` class and suitable range and depth and calculation the pairwise D', r^2, LOD and return the result as a `snp.dprime` object.

Usage

```
ld.snp(snpdata, depth = 100, start = 1, end = dim(snpdata)[2], signed.r=FALSE)
```

Arguments

snpdata	An object of `snp.matrix` class with M samples of N snps
depth	The depth or lag of pair-wise calculation. Should be between 1 and N-1; default 100. Using 0 (an invalid value) is the same as picking the maximum
start	The index of the start of the range of interest. Should be between 1 and (N-1); default 1
end	The index of the end of the range of interest. Should be between 2 and N. default N.
signed.r	Boolean for whether to returned signed r values instead of r^2

Details

The cubic equation and quadratic equation solver code is borrowed from GSL (GNU Scientific Library).

Value

return a `snp.dprime` object, which is a list of 3 named matrices `dprime`, `rsq2` (or `r` depending on the input), `lod`, and an attribute `snp.names` for the list of snps involved. (Note that if x snps are involved, the row numbers of the 3 matrices are $(x - 1)$). Only one of `rsq2` or `r` is present.

`dprime`	D'
`rsq2`	r^2
`r`	signed r
`lod`	Log of Odd's

All the matrices are defined such that the (n, m)th entry is the pair-wise value between the (n)th snp and the $(n + m)$th snp. Hence the lower right triangles are always filled with zeros. (See example section for the actual layout)

Invalid values are represented by an out-of-range value - currently we use -1 for D', r^2 (both of which are between 0 and 1), and -2 for r (valid values are between -1 and +1). lod is set to zero in most of these invalid cases. (lod can be any value so it is not indicative).

Note

The output `snp.dprime` object is suitable for input to `plot.snp.dprime` for drawing.

The speed of "ld.snp" LD calculation, on a single-processor opteron 2.2GHz box:

unsigned r^2, 13191 snps, depth 100 = 36.4 s (1.3 mil pairs)

signed r , 13191 snps, depth 100 = 40.94s (1.3 mil pairs)

signed r , 13191 snps, depth 1500 = 582s (18.5 mil pairs)

For depth=1500, it uses 500MB just for the three matrices. So I actually cannot do the full depth at 13,000; full depth should be under 50 minutes for 87 mil pairs, even in the signed-r version.

The LD code can be ran outside of R - mainly for debugging:

```
gcc -DWITHOUT_R -o /tmp/hello pairwise_linkage.c solve_cubic.c \
    solve_quadratic.c -lm
```

When used in this form, it takes 9 numbers:

```
$/tmp/hello 4 0 0 0 30 0 0 0 23
case 3                  <- internal code for which cases it falls in
root count 1            <- how many roots
trying 1.000000
p = 1.000000
4       0       0       6.333333        0.000000        0.000000
0       30      0       0.000000        25.333333       0.000000
0       0       23      0.000000        0.000000        25.333333
57 8 38.000000 38 38
8 0 0 46 30, 38 38 76 76
0.333333 0.000000 0.000000 0.666667
d' = 1.000000 , r2 = 1.000000, lod= 22.482643
```

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

References

Clayton, D.G. and Leung, Hin-Tak (2007) An R package for analysis of whole-genome association studies. *Human Heredity* **64**:45-51.
GSL (GNU Scientific Library) `http://www.gnu.org/software/gsl/`

See Also

`snp.dprime-class`, `plot.snp.dprime`, `ld.with`

Examples

```
# LD stats between 500 SNPs at a depth of 50
data(testdata)
ldinfo <- ld.snp(Autosomes, start=1, end=500, depth=50)
```

---

ld.with	*function to calculate the LD measures of specific SNPs against other SNPs*

Description

This function calculates the LD measures (r^2, D', LOD) of specific SNPs against other SNPs.

Usage

```
ld.with(data, snps, include.itself = as.logical(length(snps) - 1), signed.r = NULL)
```

Arguments

data	either a `snp.dprime-class` object or a `snp.matrix-class` object
snps	A list of snps, some of which are found in `data`
include.itself	
	Whether to include LD measures of SNPs against itself - it is FALSE for one SNP, since in that case, the result is known and trivial; but otherwise TRUE
signed.r	Logical, whether to output signed r or r^2

Details

Not all combinations of the `include.itself` and `signed.r` make sense, nor fully operational.

Value

The returned value is somewhat similiar to a `snp.dprime` object, but not the same. It is a list of 3 named matrices `dprime`, `rsq2` (or `r` depending on the input), `lod`.

Warning

Because this is really two functions rolled into one, depending on the class of `data`, not all combinations of the `include.itself` and `signed.r` make sense, nor fully operational.

Also, the two versions have slightly different idea about invalid values, e.g. the LOD value for a SNPs against itself, or r^2 for two monomorphic snps (such as one against itself).

Note

The `ld.with` function started its life as an extractor function to take the output of `ld.snp`, a `snp.dprime-class` object, to rearrange it in a more convenient form to focus on the LD's against specific SNPs, but then evolved to take a `snp.matrix-class` object alternatively and perform the same task directly and more efficiently.

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

See Also

`ld.snp`, `snp.dprime-class`

Examples

```
data(testdata)
snps10 <- Autosomes[1:10,1:10]
obj.snp.dprime <- ld.snp(snps10)

# result1 and result2 should be almost identical
# except where noted in the warning section above:
result1 <- ld.with(obj.snp.dprime, colnames(snps10))
result2 <- ld.with(snps10, colnames(snps10))
```

---

pair.result.ld.snp *Function to calculate the pairwise D', r-squared, LOD of a pair of specified SNPs*

Description

`pair.result.ld.snp.Rd` calculates the pairwise D', r^2, LOD of a pair of specified SNPs in a `snp.matrix` object. This is used mainly for debugging.

Usage

```
pair.result.ld.snp(snpdata, loc.snpA, loc.snpB)
```

Arguments

`snpdata`	An object of `snp.matrix` class with M samples of N snps
`loc.snpA`	index of the first snp; should be between 1 and N
`loc.snpB`	index of the second snp; should be between 1 and N

Value

Returns nothing. Results are displayed in stdout/console.

Note

Not really recommended for daily usage; the result isn't saved anywhere and this routine is primarily for debugging the details and correctness of the calculation.

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

References

Clayton, D.G. and Leung, Hin-Tak (2007) An R package for analysis of whole-genome association studies. *Human Heredity* **64**:45-51.
GSL (GNU Scientific Library) http://www.gnu.org/software/gsl/

See Also

snp.matrix-class

Examples

```
data(testdata)
pair.result.ld.snp(Autosomes, 1, 2)
```

`plot.snp.dprime`	*Function to draw the pairwise D' in a eps file*

Description

`plot.snp.dprime` takes a `snp.dprime` object and draw an eps file to visualize the pairwise D', r^2 and LOD.

Usage

```
## S3 method for class 'snp.dprime':
plot(x, filename, scheme = "standard", do.notes = FALSE,
metric=NULL, ...)
```

Arguments

x	An object of class `snp.dprime`
filename	The output file name, preferably ending with ".eps" (not enforced)
scheme	The colour scheme used. Valid values are "standard" for the Haploview default, and "rsq" for grayscale r^2. More may come later
do.notes	Boolean for whether to generate pdf annotation-related code
metric	An integer vector, detailing the chromosome position of the SNP, to drawa scaled metric of the location of the SNP. If NULL, no metric would be drawn
...	place holder

Details

Annotation is a little used pdf features where certain part of a pdf file are hot spots where one can get pop-up balloons containing extra information, which doesn't appear in print. This is written to imitate the extra information one can get from right-clicking in Haploview's GUI.

Value

return nothing. Write a file as a result. And if `do.notes` is specified, Will also suggest user to execute `ps2pdf -dEPSCrop <filename>` to get a suitable pdf.

Note

Unfortunately, there are two problems with annotations: only Acrobat Reader (out of all the pdf viewers, e.g. xpdf, kpdf, evince, various ghostscript based viewers) implements the feature, and a few thousand annotations can really make Acrobat Reader crawl.

Also, Acrobat Reader has an implementation limit of 200 inches of the widest dimension of a document. This translates to 1200 snps in the current implementation of the drawing code, hence a warning is emitted that pdf written this way is not viewable by Acrobat Reader.(but viewable by xpdf, etc). A work around is possible based on LaTeX/pdfpage, or eps can be included with scaling in another document, to stay inside 200 inches.

In the future, one might want to put some additional scaling code to fit the whole drawing within an A4, for example.

There is a Google Summer of code http://code.google.com/soc/ 2006 project to improve kpdf's annotation support. http://wiki.kde.org/tiki-index.php?page=KDE%20Google%20SoC%202006%20ideas#id60851 I am involved.

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

References

Clayton, D.G. and Leung, Hin-Tak (2007) An R package for analysis of whole-genome association studies. *Human Heredity* **64**:45-51.
GSL (GNU Scientific Library) http://www.gnu.org/software/gsl/

The postscript language reference manual: [http://www.adobe.com/products/postscript/](http://www.adobe.com/products/postscript/pdfs/PLRM.pdf)
[pdfs/PLRM.pdf](http://www.adobe.com/products/postscript/pdfs/PLRM.pdf)
The pdf specification: [http://partners.adobe.com/public/developer/en/pdf/](http://partners.adobe.com/public/developer/en/pdf/PDFReference16.pdf)
[PDFReference16.pdf](http://partners.adobe.com/public/developer/en/pdf/PDFReference16.pdf)

See Also

[snp.dprime-class](#)

Examples

```
data(testdata)
# As for ld.snp example ...
data(testdata)
ldinfo <- ld.snp(Autosomes, start=1, end=500, depth=50)
# Now plot to an eps file
plot.snp.dprime(ldinfo, filename="test.eps")
```

---

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. Currently these must be of class [snp.tests.single.score](#)
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 ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

[snp.tests.single.score-class](#), [single.snp.tests](#)

---

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. Currently it combines tests produced by single.snp.tests but it will be extended to deal with further tests.

Usage

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

Arguments

...	Objects holding the (extended) test results. Currently these must be of class snp.tests.single.score
score	Is extended score information to be returned in the output object?

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 ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

snp.tests.single.score-class

Examples

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

| `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,
    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 here
`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?
`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, he 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 ⟨david.clayton@cimr.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
```

---

read.HapMap.data   *function to import HapMap genotype data as snp.matrix*

Description

Given a URL for HapMap genotype data, `read.HapMap.data`, download and convert the geno-
type data into a `snp.matrix` class object, and saving snp support infomation into an associated
data.frame.

Usage

```
read.HapMap.data(url, verbose=FALSE, save=NULL, ...)
```

Arguments

url   URL for HapMap data. Web data is to be specified with prefix "http://", ftp data
                    with prefix "ftp://", and local file as "file://"

verbose   Where the dnSNPalleles annotation is ambiguous, output more details informa-
                    tion about how/why assignment is made. See Notes below.

save   filename to save the download - if unspecified, a temporary file will be created
                    but removed afterwards.

...   Place-holder for further switches - currently ignored.

Details

During the conversion, if the dbSNPAlleles entry is exactly of the form "X/Y", where X, Y = A or
C or G or T, then it is used directly for assigning allele 1 and allele 2.

However, about 1 in 1000 entries are more complicated e.g. may involving deletion, e.g. "-/A/G"
or "-/A/AGT/G/T". Some heuristics are used in such cases, in which the observed genotypes in the
specific snp of the current batch are examined in two passes. The first time to see which bases are
present, excluding "N".

If more than 2 bases are observed in the batch specified in the url, the routine aborts, but so far
this possibility has not arisen in tests. If there is exactly two, then allele 1 and 2 are assigned in
alphabatical order (dbSNPAlleles entries seems to be always in dictionary order, so the assignment
made should agree with a shorten version of the dbSNPAlleles entry). Likewise, if only "A" or "T"
is observed, then we know automatically it is the first (assigned as "A/.") or the last allele (assigned
as "./T") of a hypothetical pair, without looking at the dbSNPAlleles entry. For other observed cases
of 1 base, the routine goes further and look at the dnSNPAlleles entry and see if it begins with
"-/X/" or ends with "/X", as a single base, and compare it with the single base observed to see if it
should be allele 1 (same as the beginning, or different from the end) and allele 2 (same as the end,
or different from the beginning). If no decision can be made for a particular snp entry, the routine
aborts with an appropriate message. (for zero observed bases, assignment is "./.", and of course, all
observed genotypes of that snp are therefore converted to the equivalent of NA)

(This heuristics does not cover all grounds, but practically it seems to work. See Notes below.)

Value

Returns a list containing these two items when successful, otherwise returns NULL:

snp.data   A `snp.matrix-class` object containing the snp data

snp.support A data.frame, containing the `dbSNPalleles`, `Chromosome`, `Position`, `Strand` entries from the hapmap genotype file, together with the actual `Assignment` used for allele 1 and allele 2 during the conversion (See Details above and Note below).

Note

Using both "file://" for `url` and `save` duplicates the file. (i.e. by default, the routine make a copy of the url in any case, but tidy up afterwards if run without `save`).

Sometimes the assignment may not be unique e.g. dnSNPAlleles entry "A/C/T" and only "C" is observed - this can be assigned "A/C" or "C/T". (currently it does the former). One needs to be especially careful when joining two sets of snp data and it is imperative to compare the assigment supplementary data to see they are compatible. (e.g. for an "A/C/T" entry, one data set may have "C" only and thus have assignment "A/C" and have all of it assigned Allele 2 homozygotes, whereas another data set contains both "C" and "T" and thus the first set needs to be modified before joining).

A typical run, chromosome 1 for CEU, contains about 400,000 snps and 100 samples, and the snp.matrix object is about 60MB (40 million bytes for snps plus overhead) and similiar for the support data (i.e. 2x), takes about 30 seconds, and at peak memory usage requires 4x . The actual download is 20MB, which is compressed from 200MB.

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

References

<http://www.hapmap.org/genotypes>

See Also

[snp.matrix-class](#)

Examples

```
## Not run:

## ** Please be aware that the HapMap project generates new builds from
## ** to time and the build number in the URL changes.

> library(snpMatrix)
> testurl <- "http://www.hapmap.org/genotypes/latest/fwd_strand/non-redundant/genotypes_chr1
> result1 <- read.HapMap.data(testurl)
> sum1 <- summary(result1$snp.data)

> head(sum1[is.finite(sum1$z.HWE),], n=10)
```

```
          Calls Call.rate        MAF       P.AA       P.AB       P.BB      z.HWE
rs1933024    87 0.9666667 0.005747126 0.0000000 0.01149425 0.9885057 0.05391549
rs11497407   89 0.9888889 0.005617978 0.0000000 0.01123596 0.9887640 0.05329933
rs12565286   88 0.9777778 0.056818182 0.0000000 0.11363636 0.8863636 0.56511033
rs11804171   83 0.9222222 0.030120482 0.0000000 0.06024096 0.9397590 0.28293272
rs2977656    90 1.0000000 0.005555556 0.9888889 0.01111111 0.0000000 0.05299907
rs12138618   89 0.9888889 0.050561798 0.0000000 0.10112360 0.8988764 0.50240136
rs3094315    88 0.9777778 0.136363636 0.7272727 0.27272727 0.0000000 1.48118392
rs17160906   89 0.9888889 0.106741573 0.0000000 0.21348315 0.7865169 1.12733108
rs2519016    85 0.9444444 0.047058824 0.0000000 0.09411765 0.9058824 0.45528615
rs12562034   90 1.0000000 0.088888889 0.0000000 0.17777778 0.8222222 0.92554468

## ** Please be aware that the HapMap project generates new builds from
## ** to time and the build number in the URL changes.

## This URL is broken up into two to fit the width of
## the paper. There is no need in actual usage:
> testurl2 <- paste("http://www.hapmap.org/genotypes/latest/",
              "fwd_strand/non-redundant/genotypes_chr1_JPT_r21_nr_fwd.txt.gz", sep="")
> result2 <- read.HapMap.data(testurl2)

> head(result2$snp.support)
         dbSNPalleles Assignment Chromosome Position Strand
rs10399749        C/T        C/T       chr1    45162      +
rs2949420         A/T        A/T       chr1    45257      +
rs4030303         A/G        A/G       chr1    72434      +
rs4030300         A/C        A/C       chr1    72515      +
rs3855952         A/G        A/G       chr1    77689      +
rs940550          C/T        C/T       chr1    78032      +
## End(Not run)
```

---

read.pedfile.info   *function to read the accompanying info file of a LINKAGE ped file*

Description

This function read the accompanying info file of a LINKAGE ped file, for the SNP names, position and chromosome.

Usage

```
read.pedfile.info(file)
```

Arguments

file   An info file

Details

One such info file is the one accompanying the sample ped file of Haploview.

Value

A data frame with columns "snp.names", "position", "chromosome".

Note

This is used internally by `read.snps.pedfile` to read an accompanying info file.

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

References

See the documentation and description of ped files in Haploview (http://www.broad.mit.edu/mpg/haploview/)

See Also

`read.snps.pedfile`

---

read.pedfile.map   *function to read the accompanying map file of a LINKAGE ped file*

Description

This function read the accompanying map file of a LINKAGE ped file, for the SNP names, position and chromosome.

Usage

```
read.pedfile.map(file)
```

Arguments

file   A Plink map file

Details

One such map file is the one accompanying the sample ped file of Haploview.

Value

A data frame with columns "snp.names", "position", "chromosome".

Note

This is used internally by `read.snps.pedfile` to read an accompanying map file.

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

References

See the documentation and description of ped files in Haploview ([http://www.broad.mit.](http://www.broad.mit.edu/mpg/haploview/)
[edu/mpg/haploview/](http://www.broad.mit.edu/mpg/haploview/))

See Also

read.snps.pedfile

---

read.plink   *Read a PLINK binary data file as a snp.matrix*

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
`"snp.matrix"`

Usage

```
read.plink(bed, bim, fam)
```

Arguments

bed	The name of the file containing the packed binary SNP genotype data
bim	The file containing the SNP descriptions
fam	The file containing subject (and, possibly, family) identifiers. This is basically a tab-delimited "pedfile"

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

Value

An object of class `"snp.matrix"`. The column names are the SNP names as listed in the `bim` file. If the subject identifiers in the `fam` file are not duplicated, these are used for the row names of the output object. But, if there are duplicated subject identifiers, the row names are generated by concatenating the family and subject identifiers (separated by the character `:`)

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

References

put references to the literature/web site here

See Also

read.HapMap.dataread.snps.pedfile, read.snps.chiamo, read.snps.long, snp.matrix-class, X.snp.matrix-class

---

read.snps.chiamo   *Read genotype data from the output of Chiamo*

Description

This function reads data from the raw output of Chiamo

Usage

```
read.snps.chiamo(filename, sample.list, threshold)
```

Arguments

filename	List of file names of output from Chiamo ; the outcome is the concatenation from runs of Chiamo, e.g. on blocks of SNPs, which is often done for practical reasons
sample.list	A character vector giving the sample list
threshold	Cut-off for the posterior probability for a no-call

Details

The raw output of Chiamo consists of the first 5 columns of read.wtccc.signals, followed by triplets of posterior probabilities of calling A-A, A-B, or B-B.

The sample list can typically be obtained using wtccc.sample.list, from one of the (smaller) signal files, which are the inputs to Chiamo.

Value

The result is a list of two items:

snp.data	The genotype data as a snp.matrix-class object.
snp.support	The information from the first 5 columns of read.wtccc.signals.

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

References

To obtain a copy of the Chiamo software please email Jonathan L. Marchini <marchini@stats.ox.ac.uk>.

See Also

[wtccc.sample.list](), [read.wtccc.signals]()

Examples

```
#
```

---

read.snps.long.old *Read SNP input data in "long" format (old version)*

Description

This function reads SNP genotype data and creates an object of class `"snp.matrix"` or `"X.snp.matrix"`. Input data are assumed to be arranged as one line per SNP-call (without any headers). This function can read gzipped files.

Usage

```
read.snps.long.old(file, chip.id, snp.id, codes, female,
                       conf = 1, threshold = 0.9, drop=FALSE,
                       sorted=FALSE, progress=interactive())
```

Arguments

file	Name of file containing the input data. Input files which have been compressed by the `gzip` utility are recognized
chip.id	Array of type `"character"` containing (unique) identifiers for the chips, samples, or subjects for which calls are to be read. Other samples in the input data will be ignored
snp.id	Array of type `"character"` containing (unique) identifiers of the SNPs for which data will be read. Again, further SNPs in the input data will be ignored
codes	For autosomal SNPs, an array of length 3 giving the codes for the three genotypes, in the order homozygous(AA), heterozygous(AB), homozygous(BB). For X SNPs, an additional two codes for the male genotypes (AY and BY) must be supplied. All other codes will be treated as "no call". The default codes are `"0"`, `"1"`, `"2"` [`"0"`, `"2"`]
female	If the data to be read refer to SNPs on the X chromosome, this argument must be supplied and should indicate whether each row of data refers to a female (`TRUE`) or to a male (`FALSE`). The output object will then be of class `"X.snp.matrix"`.
conf	Confidence score. See details

drop	If TRUE, any rows or columns without genotype calls will be dropped from the output matrix. Otherwise the full matrix, with rows and columns defined by the chip.id and snp.id arguments, will be returned
threshold	Acceptance threshold for confidence score
sorted	Is input file already sorted into the correct order (see details)?
progress	If TRUE, progress will be reported to the standard output stream

Details

Data are assumed to be input with one line per call, in free format:

<chip-id> <snp-id> <code for genotype call> [*<confidence>*] *...*

Currently, any fields following the first three (or four) are ignored. If the argument sorted is TRUE, the file is assumed to be sorted with *snp-id* as primary key and *chip-id* as secondary key using the current locale. The rows and columns of the returned matrix will also be ordered in this manner. If sorted is set to FALSE, then an algorithm which avoids this assumption is used. The rows and columns of the returned matrix will then be in the same order as the input chip_id and snp_id vectors. Calls in which both id fields match elements in the chip.id and snp.id arguments are read in, after (optionally) checking that the level of confidence achieves a given threshold. Confidence level checking is controlled by the conf argument. conf=0 indicates that no confidence score is present and no checking is done. conf>0 indicates that calls with scores *above* threshold are accepted, while conf<0 indicates that only calls with scores *below* threshold should be accepted.

The routine is case-sensitive and it is important that the *<chip-id>* and *<snp-id>* match the cases of chip.id and snp.id exactly.

Value

An object of class snp.matrix.

Note

If more than one instance of any combination of chip_id element and snp_id element passes the confidence threshold, the called to be used is decided by the following rules:

1 Any call trumps "no-call"

2 In the event of call conflict, "no-call" is returned

Use of sorted=TRUE is usually discouraged since the alternative algorithm is safer and, usually, not appreciably slower. However, if the input file is to be read multiple times and there is a reasonably close correspondence between cells of the matrix to be returned and lines of the input file, the sorted option can be faster.

This function has been replaced by the more flexible function read.snps.long.

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩ and Hin-Tak Leung

References

<http://www-gene.cimr.cam.ac.uk/clayton>

See Also

`snp.matrix-class`, `X.snp.matrix-class`

`read.snps.long`	*Read SNP data in long format*

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 `"snp.matrix"`.

Usage

```
read.snps.long(files, sample.id = NULL, snp.id = NULL, female = 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, 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
`female`	If the SNPs are on the X chromosome and the data are to be read as such, this logical vector (of the same length as `sample.id` should specify whether each sample was from a female subject
`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 snp.matrix
verbose	If TRUE, a progress report is generated as every every lines of data are read
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.

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

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; this is detected from the input. Each file may represent a separate sample or SNP, in which case the appropriate .id argument can be omitted and row or column names taken from the file names.

Value

An object of class "snp.matrix" or "X.snp.matrix".

Note

The function will read gzipped files.

This function has replaced and earlier version which was much less flexible. Because all features have not been fully tested, the older version has been retained as read.snps.long.old.

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 ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

read.HapMap.dataread.snps.pedfile, read.snps.chiamo, read.plink, snp.matrix-class, X.snp.matrix-class

---

read.snps.pedfile   *Read genotype data from a LINKAGE "pedfile"*

Description

This function reads data arranged as a LINKAGE "pedfile" with some restrictions and returns a list of three objects: a data frame containing the initial 6 fields giving pedigree structure, sex and disease status, a vector or a data frame containing snp assignment and possibly other snp infomation, and an object of class "snp.matrix" or "X.snp.matrix" containing the genotype data

Usage

```
read.snps.pedfile(file, snp.names=NULL, assign=NULL, missing=NULL, X=FALSE, sep=".'
```

Arguments

file	The file name for the input pedfile
snp.names	A character vector giving the SNP names. If an accompanying map file or an info file is present, it will be read and the information used for the SNP names, and also the information merged with the result. If absent, the SNPs will be named numerically ("1", "2", ...)
assign	A list of named mappings for which letter maps to which Allele; planned for the future, not currently used
missing	Meant to be a single character giving the code recorded for alleles of missing genotypes ; not used in the current code
X	If TRUE the pedfile is assumed to describe loci on the X chromosome
sep	The character separating the family and member identifiers in the constructed row names; not used
low.mem	Switch over to input with a routine which requires less memory to run, but takes a little longer. This option also has the disadvantage that assignment of A/B genotype is somewhat non-deterministic and depends the listed order of samples.

Details

Input variables are assumed to take the usual codes, with the restriction that the family (or pedigree) identifiers will be held as strings, but identifiers for members within families must be coded as integers. Genotype should be coded as pairs of single character allele codes (which can be alphameric or numeric), from either 'A', 'C', 'G', 'T' or '1', '2', '3', '4', with 'N', '-' and '0' denoting a missing; everything else is considered invalid and would invalidate the whole snp; also more than 2 alleles also cause the snp to be marked invalid.

Row names of the output objects are constructed by concatenation of the pedigree and member identifiers, "Family", "Individual" joined by ".", e.g. "Family.Adams.Individual.0".

Value

snps   The output "snp.matrix" or "X.snp.matrix"
subject.support
                A data frame containing the first six fields of the pedfile

Author(s)

Hin-Tak Leung

See Also

[snp.matrix-class](), [X.snp.matrix-class](), [read.snps.long](), [read.HapMap.data](),
[read.pedfile.info](), [read.pedfile.map]()

---

read.wtccc.signals   *read normalized signals in the WTCCC signal file format*

Description

read.wtccc.signals takes a file and a list of snp ids (either Affymetrix ProbeSet IDs or rs numbers), and extract the entries into a form suitable for plotting and further analysis

Usage

```
read.wtccc.signals(file, snp.list)
```

Arguments

file   file contains the signals. There is no need to gunzip.
snp.list   A list of snp id's. Some Affymetrix SNPs don't have rsnumbers both rsnumbers
                and Affymetrix ProbeSet IDs are accepted

Details

Do not specify both rs number and Affymetrix Probe Set ID in the input; one of them is enough.

The signal file is formatted as follows, with the first 5 columns being the Affymetrix Probe Set ID, rs number, chromosome position, AlleleA and AlleleB. The rest of the header containing the sample id appended with "_A" and "_B".

```
  AFFYID         RSID       pos    AlleleA AlleleB 12999A2_A 12999A2_B ...
  SNP_A-4295769  rs915677   14433758 C       T       0.318183  0.002809
  SNP_A-1781681  rs9617528  14441016 A       G       1.540461  0.468571
  SNP_A-1928576  rs11705026 14490036 G       T       0.179653  2.261650
```

The routine matches the input list against the first and the 2nd column.

(some early signal files, have the first "AFFYID" missing - this routine can cope with that also)

Value

The routine returns a list of named matrices, one for each input SNP (NULL if the SNP is not found); the row names are sample IDs and columns are "A", "B" signals.

Note

TODO: There is a built-in limit to the input line buffer (65535) which should be sufficient for 2000 samples and 30 characters each. May want to seek backwards, re-read and dynamically expand if the buffer is too small.

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

References

http://www.wtccc.org.uk

Examples

```
## Not run:
answer <-
  read.wtccc.signals("NBS_22_signals.txt.gz", c("SNP_A-4284341","rs4239845"))
> summary(answer)
             Length Class  Mode
SNP_A-4284341 2970   -none- numeric
rs4239845      2970   -none- numeric

> head(a$"SNP_A-4284341")
              A        B
12999A2 1.446261 0.831480
12999A3 1.500956 0.551987
12999A4 1.283652 0.722847
12999A5 1.549140 0.604957
```

```
12999A6 1.213645 0.966151
12999A8 1.439892 0.509547
>
## End(Not run)
```

---

row.summary   *Summarize rows of a snp matrix*

Description

This function calculates call rates and heterozygosity for each row of a an object of class `"snp.matrix"`

Usage

```
row.summary(object)
```

Arguments

object   genotype data as a `snp.matrix-class` or `X.snp.matrix-class` object

Value

A data frame with rows corresponding to rows of the input object and with columns/elements:

Call.rate   Proportion of SNPs called

Heterozygosity
                Proportion of called SNPs which are heterozygous

Note

The current version does not deal with the X chromosome differently, so that males are counted as homozygous

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

Examples

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

---

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, 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 cor-
              respondence 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 `"snp.matrix"` containing the SNP genotypes to be tested

rules   An object of class `"snp.reg.imputation"`. 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 analy-
              sis. 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`

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 "snp.tests.single". If score=TRUE, the output object will be of the extended class "snp.tests.single.score" 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 X.snp.matrix is as described by Clayton (2008). Males are treated as homozygous females and corrected variance estimates are used.

Author(s)

David Clayton ⟨david.clayton@cimr.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*, In press)

See Also

snp.lhs.tests, snp.rhs.tests, impute.snps, pool, snp.reg.imputation-class,
snp.tests.single-class, snp.tests.single.score-class

Examples

```
data(testdata)
results <- single.snp.tests(cc, stratum=region, data=subject.data,
   snp.data=Autosomes, snp.subset=1:10)
summary(results)
# QQ plot - see this-is-escaped-codenormal-bracket72bracket-normal
qq.chisq(chi.squared(results, 1), 1)
qq.chisq(chi.squared(results, 2), 2)
```

---

snp.cbind   *Bind together two or more snp.matrix objects*

Description

These functions bind together two or more objects of class `"snp.matrix"` or `"X.snp.matrix"`.

Usage

```
snp.cbind(...)
snp.rbind(...)
```

Arguments

`...`	Objects of class `"snp.matrix"` or `"X.snp.matrix"`.

Details

These functions reproduce the action of the standard functions `cbind` and `rbind`. These are constrained to work by recursive calls to the generic functions `cbind2` and `rbind2` which take just two arguments. This is somewhat inefficient in both time and memory use when binding more than two objects, so the functions `snp.cbind` and `snp.rbind`, which take multiple arguments, are also supplied.

When matrices are bound together by column, row names must be identical, column names must not be duplicated and, for objects of class `X.snp.matrix` the contents of the `Female` slot much match. When matrices are bound by row, column names must be identical. and duplications of row names generate warnings.

Value

A new matrix, of the same type as the input matrices.

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

`cbind`, `rbind`

Examples

```
data(testdata)
# subsetting ( Autosomes[c(1:9,11:19,21:29),] ) is quicker. this is just for illustrating
# rbind and cbind
first <- Autosomes[1:9,]
second <- Autosomes[11:19,]
third <- Autosomes[21:29,]
```

```
result1 <- rbind(first, second, third)
result2 <- snp.rbind(first, second, third)
all.equal(result1, result2)

result3 <- Autosomes[c(1:9,11:19,21:29),]
all.equal(result1, result3)

first <- Autosomes[,1:9]
second <- Autosomes[,11:19]
third <- Autosomes[,21:29]
result1 <- cbind(first, second, third)
result2 <- snp.cbind(first, second, third)
all.equal(result1, result2)

result3 <- Autosomes[,c(1:9,11:19,21:29)]
all.equal(result1, result3)

first <- Xchromosome[1:9,]
second <- Xchromosome[11:19,]
third <- Xchromosome[21:29,]
result1 <- rbind(first, second, third)
result2 <- snp.rbind(first, second, third)
all.equal(result1, result2)

result3 <- Xchromosome[c(1:9,11:19,21:29),]
all.equal(result1, result3)

first <- Xchromosome[,1:9]
second <- Xchromosome[,11:19]
third <- Xchromosome[,21:29]
result1 <- cbind(first, second, third)
result2 <- snp.cbind(first, second, third)
all.equal(result1, result2)

result3 <- Xchromosome[,c(1:9,11:19,21:29)]
all.equal(result1, result3)
```

---

snp-class   *Class "snp"*

Description

Compact representation of data concerning single nucleotide polymorphisms (SNPs)

Objects from the Class

Objects can be created by calls of the form `new("snp", ...)` or by subset selection from an object of class `"snp.matrix"`. Holds one row or column of an object of class `"snp.matrix"`

Slots

.**Data:** The genotype data coded as 0, 1, 2, or 3

Methods

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

coerce signature(from = "snp", to = "numeric"): map to codes 0, 1, 2, or NA

coerce signature(from = "snp", to = "genotype"): maps a single SNP to an object of class "genotype". See the "genetics" package.

show signature(object = "snp"): shows character representation of the object

is.na signature(x = "snp"): returns a logical vector of missing call indicators

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

References

http://www-gene.cimr.cam.ac.uk/clayton

See Also

snp.matrix-class, X.snp.matrix-class, X.snp-class

Examples

```
## data(testdata)
## s <- autosomes[,1]
## class(s)
## s
```

---

snp.cor   *Correlations with columns of a snp.matrix*

Description

This function calculates Pearson correlation coefficients between columns of a snp.matrix 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)
```

Arguments

x	An *N* by *M* snp.matrix
y	An *N* by *P* general matrix

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 ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

xxt

Examples

```
# make a snp.matrix 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,
# for eaxample to plot against chromosome position
loadings <- snp.cor(small, pc[,1:10])
```

---

snp.dprime-class   *Class "snp.dprime" for Results of LD calculation*

Description

The snp.dprime class encapsulates results returned by ld.snp (— routine to calculate D', r^2 and LOD of a snp.matrix-class object, given a range and a depth) and is based on a list of three named matrices.

The lower right triangle of the snp.dprime object returned by ld.snp always consists zeros. This is delibrate. The associated plotting routine would not normally access those elements either.

Value

The `snp.dprime` class is a list of 3 named matrices `dprime`, `rsq2` or `r`, `lod`, and an attribute `snp.names` for the list of snps involved. (Note that if x snps are involved, the row numbers of the 3 matrices are $(x - 1)$). Only one of `r` or `rsq2` is present.

`dprime`	D'
`rsq2`	r^2
`r`	signed r^2
`lod`	Log of Odd's
`attr(*, class)`	"snp.dprime"
`attr(*, snp.names)`	character vectors of the snp names involved

All the matrices are defined such that the (n, m)th entry is the pair-wise value between the (n)th snp and the $(n + m)$th snp. Hence the lower right triangles are always filled with zeros.

Invalid values are represented by an out-of-range value - currently we use -1 for D', r^2 (both of which are between 0 and 1), and -2 for r (valid values are between -1 and +1). lod is set to zero in most of these invalid cases. (lod can be any value so it is not indicative).

Methods

See `plot.snp.dprime`.

Note

TODO: Need a subsetting operator.

TODO: an assemble operator

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

Source

reference to a publication or URL from which the data were obtained

References

possibly secondary sources and usages

Examples

```
data(testdata)
snps20.20 <- Autosomes[11:20,11:20]
obj.snp.dprime <- ld.snp(snps20.20)
class(obj.snp.dprime)
summary(obj.snp.dprime)
## Not run:
# The following isn't executable-as-is example, so these illustrations
# are commented out to stop R CMD check from complaining:

> d<- ld.snp(all, 3, 10, 15)
rows = 48, cols = 132
... Done
> d
$dprime
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    1    1    1
[4,]    1    1    0
[5,]    1    0    0

$rsq2
          [,1]       [,2]       [,3]
[1,] 1.0000000 0.9323467 1.0000000
[2,] 0.9285714 1.0000000 0.1540670
[3,] 0.9357278 0.1854481 0.9357278
[4,] 0.1694915 1.0000000 0.0000000
[5,] 0.1694915 0.0000000 0.0000000

$lod
          [,1]       [,2]       [,3]
[1,] 16.793677 11.909686 16.407120
[2,] 10.625650 15.117962  2.042668
[3,] 12.589586  2.144780 12.589586
[4,]  2.706318 16.781859  0.000000
[5,]  2.706318  0.000000  0.000000

attr(,"class")
[1] "snp.dprime"
attr(,"snp.names")
[1] "dil118"  "dil119"  "dil5904" "dil121"  "dil5905" "dil5906"

## End(Not run)
```

---

snp.imputation   *Calculate regression-based imputation rules*

Description

Given two set of SNPs typed in the same subjects, this function calculates regression equations which can be used to impute one set from the other in a subsequent sample.

Usage

```
snp.imputation(X, Y, pos.X, pos.Y, phase=FALSE, try = 50, r2.stop = 0.9, max.X = 4)
```

Arguments

X	An object of class `"snpMatrix"` or `"X.snp.matrix"` containing observations of the SNPs to be used for imputation ("regressor SNPs")
Y	An object of same class as X containing observations of the SNPs to be imputed in a future sample ("target SNPs")
pos.X	The positions of the regressor SNPs
pos.Y	The positions of the target SNPs
phase	See "Details" below
try	The number of potential regressor SNPs to be considered in the stepwise regression procedure around each target SNP . The nearest `try` regressor SNPs to each target SNP will be considered
r2.stop	The value of R^2 at which to stop entering new variables in the step-wise regression. If set to 1.0 or greater, the AIC will be used
max.X	The maximum number of regressor SNPs to be used for each imputation

Details

The routine carries out a series of step-wise regression analyses in which each Y SNP is regressed on the nearest `try` regressor SNPs. If `phase` is TRUE, the regressions are calculated at the chromosome (haplotype) level, variances being simply $p(1 - p)$ and covariances estimated using the same algorithm used in `ld.snp`. Otherwise, the analysis is carried out at the diplotype level based on conventional variance and covariance estimates using the `"all.obs"` missing value treatment (see `cov`).

Value

An object of class `"snp.reg.imputation"`.

Note

The `phase=TRUE` option is not yet implemented

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

References

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

See Also

snp.reg.imputation-class, ld.snp

Examples

```
# Remove 5 SNPs from a datset and derive imputation rules for them
library(snpMatrix)
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.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 "snp.matrix" 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,
              control=glm.test.control(maxit=20, epsilon=1.e-4, R2Max=0.98))
```

Arguments

snp.data	The SNP data, as an object of class "snp.matrix" or "X.snp.matrix"
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
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 codeglm.test.control

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

A data frame containing, for each SNP,

`Chi.squared`	The value of the chi-squared test statistic
`Df`	The corresponding degrees of freedom
`Df.residual`	The residual degrees of freedom for the base model; *i.e.* the number of observations minus the number of parameters fitted

For the logistic model, the base model can, in some circumstances, lead to perfect prediction of some observations (*i.e.* fitted probabilities of 0 or 1). These observations are ignored in subsequent calculations; in particular they are not counted in the residual degrees of freedom.

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.

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

References

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

See Also

[glm.test.control](), [snp.rhs.tests](), [single.snp.tests](), [snp.matrix-class](), [X.snp.matrix-class]()

Examples

```
data(testdata)
slt1 <- snp.lhs.tests(Autosomes[,1:10], ~cc, ~region, data=subject.data)
print(slt1)
slt2 <- snp.lhs.tests(Autosomes[,1:10], ~strata(region), ~cc,
   data=subject.data)
print(slt2)
```

snp.matrix-class   *Class "snp.matrix"*

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("snp.matrix", 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 00 and SNP genotypes are coded 01, 02 or 03.

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 = "snp.matrix")`: subset operations. Currently rather slow owing to excessive copying.

cbind2 `signature(x = "snp.matrix", y = "snp.matrix")`: 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 X.snp.matrix-class, the `Female` slot values must also agree

coerce `signature(from = "snp.matrix", to = "numeric")`: map to codes 0, 1, 2, or NA

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

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

coerce `signature(from = "snp.matrix", to = "X.snp.matrix")`: maps a snp.matrix to an X.snp.matrix. Sex is inferred from the genotype data since males should not be heterozygous at any locus. After inferring sex, heterozygous calls for males are set to `NA`

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

rbind2 `signature(x = "snp.matrix", 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 = "snp.matrix")`: shows the size of the matrix (since most
objects will be too large to show in full)

summary `signature(object = "snp.matrix")`: calculate call rates, allele frequencies,
genotype frequencies, and z-tests for Hardy-Weinberg equilibrium. Results are returned as a
dataframe with column names `Calls`, `Call.rate`, `MAF`, `P.AA`, `P.AB`, `P.BB`, and `z.HWE`

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

show `signature(object = "snp.matrix")`: ...

summary `signature(object = "snp.matrix")`: ...

Note

This class requires at least version 2.3 of R

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

References

http://www-gene.cimr.cam.ac.uk/clayton

See Also

snp-class, X.snp-class, X.snp.matrix-class

Examples

```
data(testdata)
summary(summary(Autosomes))

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

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

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

---

snpMatrix-internal *snpMatrix-internal*

Description

All the dirty details that doesn't belong elsewhere. At the moment just for hiding references to the genotype-class and haplotype-class class which is in the genetics package. The two links should be [genotype-class](#) and [haplotype-class](#) .

---

snpMatrix-package   *The snp.matrix and X.snp.matrix classes*

Description

Implements classes and some basic methods for large-scale SNP association studies

Details

Package: snpMatrix
Version: 1.5.4
Date: 2008-08-21
Depends: R(>= 2.3.0), survival, methods
Suggests: hexbin
Enhances: genetics
License: GNU General Public Licence (GPLv3)
URL: http://www-gene.cimr.cam.ac.uk/clayton/software/
Collate: ld.with.R ss.R contingency.table.R glm_test.R ibs.stats.R imputation.R indata.R ld.snp.R ld.with.R.eml.R misc.R
LazyLoad: yes
biocViews: Microarray, SNPsAndGeneticVariability
Packaged: Thu Aug 21 12:41:45 BST 2008
Built: R 2.7.1; x86_64-pc-linux-gnu; 2008-08-21 13:17:24; unix

Index:

```
X.snp-class           Class "X.snp"
X.snp.matrix-class    Class "X.snp.matrix"
chi.squared           Extract test statistics and p-values
epsout.ld.snp         Function to write an eps file directly to
                      visualize LD
filter.rules          Filter a set of imputation rules
for.exercise          Data for exercise in use of the snpMatrix
                      package
genotype-class        snpMatrix-internal
glm.test.control      Set up control object for GLM tests
```

```
ibs.stats             function to calculate the identity-by-state
                      stats of a group of samples
ibsCount              Count alleles identical by state
ibsDist               Distance matrix based on identity by state
                      (IBS)
impute.snps           Impute snps
ld.snp                Function to calculate pairwise D', r-squared
ld.with               function to calculate the LD measures of
                      specific SNPs against other SNPs
pair.result.ld.snp    Function to calculate the pairwise D',
                      r-squared, LOD of a pair of specified SNPs
plot.snp.dprime       Function to draw the pairwise D' in a eps file
pool                  Pool test results from several studies or
                      sub-studies
pool2                 Pool results of tests from two independent
                      datasets
qq.chisq              Quantile-quantile plot for chi-squared tests
read.HapMap.data      function to import HapMap genotype data as
                      snp.matrix
read.pedfile.info     function to read the accompanying info file of
                      a LINKAGE ped file
read.pedfile.map      function to read the accompanying map file of a
                      LINKAGE ped file
read.plink            Read a PLINK binary data file as a snp.matrix
read.snps.chiamo      Read genotype data from the output of Chiamo
read.snps.long        Read SNP data in long format
read.snps.long.old    Read SNP input data in "long" format (old
                      version)
read.snps.pedfile     Read genotype data from a LINKAGE "pedfile"
read.wtccc.signals    read normalized signals in the WTCCC signal
                      file format
row.summary           Summarize rows of a snp matrix
single.snp.tests      1-df and 2-df tests for genetic associations
                      with SNPs (or imputed SNPs)
snp-class             Class "snp"
snp.cbind             Bind together two or more snp.matrix objects
snp.cor               Correlations with columns of a snp.matrix
snp.dprime-class      Class "snp.dprime" for Results of LD
                      calculation
snp.imputation        Calculate regression-based imputation rules
snp.lhs.tests         Score tests with SNP genotypes as dependent
                      variable
snp.matrix-class      Class "snp.matrix"
snp.pre               Pre- or post-multiply a snp.matrix object by a
                      general matrix
snp.reg.imputation-class
                      Class "snp.reg.imputation"
snp.rhs.tests         Score tests with SNP genotypes as independent
```

```
                            variable
snp.tests.single-class

                            Classes "snp.tests.single" and
                            "snp.tests.single.score"
snpMatrix-package           The snp.matrix and X.snp.matrix classes
switch.alleles              Switch alleles in columns of a snp.matrix
switch.alleles.single       Switch alleles in test results
tdt.snp                     1-df and 2-df tests for genetic associations
                            with SNPs (or imputed SNPs) in family data
test.allele.switch          Test for switch of alleles between two
                            collections
testdata                    Test data for the snpMatrix package
write.snp.matrix            Write a snp.matrix object as a text file
wtccc.sample.list           read the sample list from the header of the
                            WTCCC signal file format
xxt                         X.X-transpose for a standardized snp.matrix
```

Further information is available in the following vignettes:

imputation-vignette	imputation (source)
snpMatrix-vignette	snpMatrix (source)

Author(s)

David Clayton <david.clayton@cimr.cam.ac.uk> and Hin-Tak Leung <htl10@users.sourceforge.net>

Maintainer: David Clayton <david.clayton@cimr.cam.ac.uk>

---

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

Description

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

Usage

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

Arguments

snps	An object of class `"snp.matrix"` or `"X.snp.matrix"`
mat	A general (numeric) matrix

frequency   A numeric vector giving the allele (relative) frequencies to be used for stan-
             dardizing the columns of `snps`. If `NULL`, allele frequencies will be calculated
             internally. Frequencies should refer to the second (`B`) allele

Details

The two matrices must be conformant, as with standard matrix multiplication. The main use en-
visaged 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 exter-
nally supplied allele frequencies for standardizing the input snp.matrix is required when applying
loadings calculated from one dataset to a different dataset

Value

The resulting matrix product

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

[xxt](xxt)

Examples

```
##--
##-- Calculate first two principal components and their loading, and verify
##--
# Make a snp.matrix 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(small,  Dinv %*% t(pc))
# Now apply loadings back to recalculate the principal components
pc.again <- snp.post(small, t(loadings) %*% Dinv)
print(cbind(pc, pc.again))
```

```
snp.reg.imputation-class
```
Class "snp.reg.imputation"

Description

A class defining a list of regression equations, or "rules", for imputation of 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:

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.

coefficients A numeric array containing the regression equation intercept followed by the regression coefficients for the SNPs listed in `snps`.

If any target SNP is monomorphic, the corresponding rule is returned as `NULL`.

Methods

show `signature(object = "snp.reg.imputation")`: prints an abreviated listing of the rules

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

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

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

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

Examples

```
showClass("snp.reg.imputation")
```

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.fr
    snp.data, tests=NULL, robust = FALSE,
    control=glm.test.control(maxit=20, epsilon=1.e-4, R2Max=0.98), allow.missing=0.0
```

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 `"snp.matrix"` or `"X.snp.matrix"` containing the SNP data
tests	Either a vector of column names or numbers for the SNPs to be tested, or a list of short vectors defining groups of SNPs to be tested (again by name or number). The default action is to carry out *all* single SNP tests, but `single.snp.tests` will often achieve the same result much faster
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
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 code[glm.test.control]
allow.missing	The maximum proportion of SNP genotype that can be missing before it becomes necessary to refit the base model

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.

Value

A data frame containing, for each SNP,

`Chi.squared`	The value of the chi-squared test statistic
`Df`	The corresponding degrees of freedom
`Df.residual`	The residual degrees of freedom for the base model; *i.e.* the number of observations minus the number of parameters fitted

For the binomial family model, the base model can, in some circumstances, lead to perfect prediction of some observations (*i.e.* fitted probabilities of 0 or 1). These observations are ignored in subsequent calculations; in particular they are not counted in the residual degrees of freedom. Similarly for Poisson means fitted exactly to zero.

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 ⟨david.clayton@cimr.cam.ac.uk⟩

References

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

See Also

`single.snp.tests`, `snp.lhs.tests`, `snp.matrix-class`, `X.snp.matrix-class`

Examples

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

```
snp.tests.single-class
```
                    Classes "snp.tests.single" and "snp.tests.single.score"

Description

These are classes to hold the objects created by `single.snp.tests` and provide methods for extracting key elements. The class `"snp.tests.single.score"` extends class `"snp.tests.single"` 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("snp.tests.single", ...)` and `new("snp.tests.single.s` `...)` 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 `snp.matrix`

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 `"snp.tests.single.score"`) Score statistics

V: (class `"snp.tests.single.score"`) Score variances

Methods

[ ] signature(x = "snp.tests.single", i = "ANY"): Subsetting operator

[ ] signature(x = "snp.tests.single.score", i = "ANY"): Subsetting operator

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

chi.squared signature(x = "snp.tests.single.score", df = "numeric"): Extract 1- and 2-df chi-squared test values

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

p.value signature(x = "snp.tests.single.score", df = "numeric"): Evaluate 1- and 2-df test p-values

show signature(object = "snp.tests.single"): List all tests and p-values

summary signature(object = "snp.tests.single"): Summarize all tests and p-values

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

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

single.snp.tests, pool

Examples

```
showClass("snp.tests.single")
showClass("snp.tests.single.score")
```

---

switch.alleles   *Switch alleles in columns of a snp.matrix*

Description

Switch the allele coding for specified columns of an object of class "snp.matrix" or "X.snp.matrix". Thus *AA* genotypes become *BB* and vice-versa while *AB* remain unchanged.

Usage

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

Arguments

snps   The input snp.matrix or X.snp.matrix

cols   A vector of type integer, character or logical specifying the column selection

Value

A snp.matrix in which the specified columns have had their alleles switched.

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

`test.allele.switch`, `snp.matrix-class`, `X.snp.matrix-class`

Examples

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

`switch.alleles.single`

Switch alleles in test results

Description

In meta-analysis of two or more studies, tests are calculated separately for each study, saved, and then pooled across studies. However certain SNPs may have alleles reversed in one study, resulting in incorrect pooling of results. This function corrects the saved test statistics for such switches.

Usage

```
switch.alleles.single(tests, switch)
```

Arguments

tests	An object of class `snp.tests.single.score` holding the test results
switch	A logical, integer, or character array identifying which SNPs should have alleles switched

Details

It is sometimes more convenient to use this function rather than to use the function `switch.alleles` to correct the original data matrix.

Value

An object of class `snp.tests.single.score` holding a copy of the input matrix in which the required allele switches have been made.

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

`snp.tests.single.score-class`, `switch.alleles`

`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,
    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 `"snp.matrix"` containing the SNP genotypes to be tested
`rules`	An object of class `"snp.reg.imputation"`. 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")
`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).

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 `"snp.tests.single"`. If `score=TRUE`, the output object will be of the extended class `"snp.tests.single.score"` 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 `"X.snp.matrix"`), 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 ⟨david.clayton@cimr.cam.ac.uk⟩

References

Clayton (2008) Testing for association on the X chromosome *Biostatistics*, Advanced on-line access)

See Also

`single.snp.tests`, `impute.snps`, `pool`, `snp.reg.imputation-class`, `snp.tests.single-class`, `snp.tests.single.score-class`

Examples

```
# No example yet
```

`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 `"snp.matrix"` to be examined, column by column, for allele switching, or a single `"snp.matrix"` 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 `"snp.matrix"` or `"X.snp.matrix"`

snps2   A second object of the same class as `snps`

split   If only one `snp.matrix` 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.df` so 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 ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

[snp.matrix-class](), [X.snp.matrix-class]()

Examples

```
data(testdata)
#
# Call with two snp.matrix 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 snpMatrix 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 `"snp.matrix"` containing genotype calls for 400 subjects at 9445 autosomal SNPs

Xchromosome An object of class `"X.snp.matrix"` 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/todd)

References

http://www-gene.cimr.cam.ac.uk/clayton

Examples

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

---

write.snp.matrix   *Write a snp.matrix object as a text file*

Description

This function is closely modelled on write.table. It writes an object of class snp.matrix 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).

Usage

```
write.snp.matrix(x, file, append = FALSE, quote = TRUE, sep = " ", eol = "\n", na =
```

Arguments

x	The object to be written
file	The name of the output file
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 ⟨david.clayton@cimr.cam.ac.uk⟩

See Also

`write.table`, `snp.matrix-class`, `X.snp.matrix-class`

`wtccc.sample.list`   *read the sample list from the header of the WTCCC signal file format*

Description

This is a convenience function for constructing the sample list from the header of a WTCCC signal file.

Usage

```
wtccc.sample.list(infile)
```

Arguments

infile   One of the signal files in a set of 23 (it is advisable to use the smaller ones such as number 22, although it shouldn't matter).

Details

The header of a WTCCC signal file is like this:

```
AFFYID RSID pos  AlleleA AlleleB 12999A2_A 12999A2_B ...
```

The first 5 fields are discarded. There after, every other token is retained, with the "_A" or "_B" part removed to give the sample list.

See also `read.wtccc.signals` for more details.

Value

The value returned is a character vector contain the sample names or the plate-well names as appropriate.

Author(s)

Hin-Tak Leung ⟨htl10@users.sourceforge.net⟩

References

<http://www.wtccc.org.uk>

See Also

`read.wtccc.signals`

`X.snp-class`   *Class "X.snp"*

Description

Compact representation of data concerning single nucleotide polymorphisms (SNPs) on the X chromosome

Objects from the Class

Objects can be created by calls of the form `new("snp", ..., Female=...)` or by subset selection from an object of class `"X.snp.matrix"`. Holds one row or column of an object of class `"X.snp.matrix"`

Slots

`.Data:` The genotype data coded as 0, 1, 2, or 3. For males are coded as homozygious females

`Female:` A logical array giving the sex of the sample(s)

Extends

Class `"snp"`, directly. Class `"raw"`, by class `"snp"`. Class `"vector"`, by class `"snp"`.

Methods

coerce `signature(from = "X.snp", to = "character")`: map to codes "A/A", "A/B", "B/B", "A/Y", "B/Y", or ""

coerce `signature(from = "X.snp", to = "numeric")`: map to codes 0, 1, 2, or NA

coerce `signature(from = "X.snp", to = "genotype")`: Yet to be implemented

show `signature(object = "X.snp")`: shows character representation of the object

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

References

<http://www-gene.cimr.cam.ac.uk/clayton>

See Also

X.snp.matrix-class, snp.matrix-class, snp-class

Examples

```
data(testdata)
s <- Xchromosome[,1]
class(s)
s
```

X.snp.matrix-class *Class "X.snp.matrix"*

Description

This class extends the snp.matrix-class to deal with SNPs on the X chromosome.

Objects from the Class

Objects can be created by calls of the form new("X.snp.matrix", x, Female). Such objects have an additional slot to objects of class "snp.matrix" consisting of a logical array of the same length as the number of rows. This array indicates whether the sample corresponding to that row came from a female (TRUE) or a male (FALSE).

Slots

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

Female: Object of class "logical" indicating sex of samples

Extends

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

Methods

[ ] signature(x = "X.snp.matrix"): subset operations. Currently rather slow owing to excessive copying

[<- signature(x = "X.snp.matrix"): subset assignment operation to replace part of an object

coerce signature(from = "X.snp.matrix", to = "character"): map to codes 0, 1, 2, or NA

coerce signature(from = "snp.matrix", to = "X.snp.matrix"): maps a snp.matrix to an X.snp.matrix. Sex is inferred from the genotype data since males should not be heterozygous at any locus. After inferring sex, heterozygous calls for males are set to NA

show signature(object = "X.snp.matrix"): map to codes "A/A", "A/B", "B/B", "A/Y", "B/Y" or ""

summary signature(object = "X.snp.matrix"): calculate call rates, allele frequencies, genotype frequencies, and chi-square tests for Hardy-Weinberg equilibrium. Genotype frequencies are calculated for males and females separately and Hardy-Weinberg equilibrium tests use only the female data. Allele frequencies are calculated using data from both males and females. Results are returned as a dataframe with column names Calls, Call.rate, MAF, P.AA, P.AB, P.BB, P.AY, P.BY, and z.HWE

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

References

http://www-gene.cimr.cam.ac.uk/clayton

See Also

X.snp-class, snp.matrix-class, snp-class

Examples

```
data(testdata)
summary(summary(Xchromosome))
```

---

xxt   *X.X-transpose for a standardized snp.matrix*

Description

The input snp.matrix 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)
```

Arguments

snps   The input matrix, of type "snp.matrix"

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

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 snp.matrix
whose columns have been standardized to zero mean and unit variance. For autosomes, the geno-
types are given codes 0, 1 or 2 after subtraction of the mean, 2*p*, are divided by the standard
deviation sqrt(2*p*(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 2*p*, 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 "snp.matrix"
or"X.snp.matrix"), 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.

Author(s)

David Clayton ⟨david.clayton@cimr.cam.ac.uk⟩

References

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

Examples

```
# make a snp.matrix 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