Package ‘haplo.stats’

July 13, 2004

Version 1.1.1

Date 2004-04

Title Statistical Analysis of Haplotypes with Traits and Covariates when Linkage Phase is Ambiguous.

Author Jason P. Sinnwell and Daniel J. Schaid

Maintainer Jason P. Sinnwell <sinnwell@mayo.edu>

Description Haplo Stats is a suite of S-PLUS/R routines for the analysis of indirectly measured haplotypes. The statistical methods assume that all subjects are unrelated and that haplotypes are ambiguous (due to unknown linkage phase of the genetic markers). The genetic markers are assumed to be codominant (i.e., one-to-one correspondence between their genotypes and their phenotypes), and so we refer to the measurements of genetic markers as genotypes. The main functions in Haplo Stats are: haplo.em, haplo.glm and haplo.score.

License Copyright 2003 Mayo Foundation for Medical Education and Research. This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. For other licensing arrangements, please contact Daniel J. Schaid, Ph.D., Division of Biostatistics, Harwick Building Room 775, Mayo Clinic, 200 First St., SW, Rochester, MN 55905. Phone: 507-284-0639, fax: 507-284-9542 email: schaid@mayo.edu

Depends R (>= 1.7.1)

Suggests Design, Hmisc

URL http://www.mayo.edu/hsr/people/schaid.html

R topics documented:

Ginv *Compute Generalized Inverse of Input Matrix*

Description

Singular value decomposition (svd) is used to compute a generalized inverse of input matrix.

Usage

```
Ginv(x)
```

Arguments

`x`	A matrix.

Details

The function svd is used to compute the singular values of the input matrix, and the rank of the matrix is determined by the number of singular values that are at least as large as max(svd)*eps, where eps is a small value (currently eps = .000001).

Value

List with components:

`Ginv`	Generalized inverse of x.
`rank`	Rank of matrix x.

Side Effects

References

Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C. The art of scientific computing. 2nd ed. Cambridge University Press, Cambridge.1992. page 61.

See Also

svd

Examples

```
# for matrix x, extract the generalized inverse and
# rank of x as follows
#   > save <- Ginv(x)
#   > ginv.x <- save$Ginv
#   > rank.x <- save$rank
```

`allele.recode` *Recode allele values to integer ranks*

Description

Genotypes for subjects represented by a pair of vectors, with the vectors containing allele values (either numeric, factor, or character), are recoded to the rank order of allele values.

Usage

```
allele.recode(a1, a2, miss.val=NA)
```

Arguments

`a1`	Vector of "first" alleles.
`a2`	Vector of "second" alleles.
`miss.val`	Vector of missing value codes for alleles.

Details

If alleles are numeric, they are recoded to the rank order of the alleles. If the alleles are factor or character, they are recoded to interger values that correspond to the indices of the sorted values of the unique alleles, but sorted as character values.

Value

List with components:

`a1`	Vector of recoded "first" alleles.
`a2`	Recode of recoded "second" alleles.
`allele.label`	Vector of labels for unique alleles.

Side Effects

References

See Also

geno.recode

Examples

`dglm.fit` *Density function for GLM fit*

Description

For internal use within the haplo.stats library

Usage

```
dglm.fit(fit)
```

Arguments

`fit`

Details

For internal use within the haplo.stats library

Value

Side Effects

References

See Also

Examples

`geno.count.pairs` *Counts of Total Haplotype Pairs Produced by Genotypes*

Description

Provide a count of all possible haplotype pairs for each subject, according to the phenotypes in the rows of the geno matrix. The count for each row includes the count for complete phenotypes, as well as possible haplotype pairs for phenotypes where there are missing alleles at any of the loci.

Usage

```
geno.count.pairs(geno)
```

Arguments

geno — Matrix of alleles, such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a chromosome. If there are K loci, then geno has 2*K columns. Rows represent all observed alleles for each subject, their phenotype.

Details

When a subject has no missing alleles, and has h heterozygous sites, there are 2**(h-1) haplotype pairs that are possible ('**'=power). For loci with missing alleles, we consider all possible pairs of alleles at those loci. Suppose that there are M loci with missing alleles, and let the vector V have values 1 or 0 acccording to whether these loci are imputed to be heterozygous or homozygous, respectively. The length of V is M. The total number of possible states of V is 2**M. Suppose that the vector W, also of length M, provides a count of the number of possible heterozygous/homozygous states at the loci with missing data. For example, if one allele is missing, and there are K possible alleles at that locus, then there can be one homozygous and (K-1) heterozygous genotypes. If two alleles are missing, there can be K homozygous and K(K-1)/2 heterozygous genotypes. Suppose the function H(h+V) counts the total number of heterozygous sites among the loci without missing data (of which h are heterozygous) and the imputed loci (represented by the vector V). Then, the total number of possible pairs of haplotypes can be respresented as SUM(W*H(h+V)), where the sum is over all possible values for the vector V.

Value

Vector where each element gives a count of the number haplotype pairs that are consistent with a subject's phenotype, where a phenotype may include 0, 1, or 2 missing alleles at any locus.

Side Effects

See Also

haplo.em, summaryGeno

Examples

```
setupData(hla.demo)
geno <- hla.demo[,c(17,18,21:24)]
geno <- geno.recode(geno)$grec
count.geno <- geno.count.pairs(geno)
print(count.geno)
```

---

geno.recode *Recode Genotypes*

Description

For all loci as pairs of columns in a matrix, recode alleles

Usage

```
geno.recode(geno, miss.val=0)
```

Arguments

`geno`	Matrix of alleles, such that each locus has a pair of adjacent columns of alleles. If there are K loci, then ncol(geno) = 2*K. Rows represent alleles for each subject.
`miss.val`	Vector of codes for missing values of alleles.

Details

Value

List with components:

`grec`	Matrix of recoded alleles - see allele.recode
`alist`	List of allele labels. For K loci, there are K components in the list, and the kth component is a vector of sorted unique allele labels for the kth locus.

Side Effects

References

See Also

allele.recode

Examples

`glm.fit.nowarn` *Modified from glm.fit function to not warn users for binomial non-integer weights.*

Description

An internal function for the haplo.stats library

Usage

```
glm.fit.nowarn(x, y, weights = rep(1, nobs), start = NULL,
              etastart = NULL, mustart = NULL, offset = rep(0,nobs),
              family=gaussian(), control=glm.control(), intercept=TRUE)
```

Arguments

`x`	`x`
`y`	`y`
`weights`	`weights`
`start`	`start`
`etastart`	`etastart`
`mustart`	`mustart`
`offset`	`offset`
`family`	`family`
`control`	`control`
`intercept`	`intercept`

Details

Value

Note

Author(s)

Sinnwell JP

References

See Also

`haplo.glm`

Examples

haplo.em — *EM Computation of Haplotype Probabilities, with Progressive Insertion of Loci*

Description

For genetic marker phenotypes measured on unrelated subjects, with linkage phase unknown, compute maximum likelihood estimates of haplotype probabilities. Because linkage phase is unknown, there may be more than one pair of haplotypes that are consistent with the oberved marker phenotypes, so posterior probabilities of pairs of haplotypes for each subject are also computed. Unlike the usual EM which attempts to enumerate all possible pairs of haplotypes before iterating over the EM steps, this "progressive insertion" algorithm progressively inserts batches of loci into haplotypes of growing lengths, runs the EM steps, trims off pairs of haplotypes per subject when the posterior probability of the pair is below a specified threshold, and then continues these insertion, EM, and trimming steps until all loci are inserted into the haplotype. The user can choose the batch size. If the batch size is chosen to be all loci, and the threshold for trimming is set to 0, then this algorithm reduces to the usual EM algorithm.

Usage

```
haplo.em(geno, locus.label=NA, miss.val=c(0, NA), weight, control=
           haplo.em.control())
```

Arguments

`geno`	matrix of alleles, such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a chromosome. If there are K loci, then ncol(geno) = 2*K. Rows represent the alleles for each subject.
`locus.label`	vector of labels for loci.
`miss.val`	vector of values that represent missing alleles in geno.
`weight`	weights for observations (rows of geno matrix).
`control`	list of control parameters. The default is constructed by the function haplo.em.control. The default behavior of this function results in the following parameter settings: loci.insert.order=1:n.loci, insert.batch.size=min(4,n.loci), min.posterior= 0.0001, tol=0.00001, max.iter=500, random.start=0 (no random start), iseed=NULL (no saved seed to start random start), verbose=0 (no printout during EM iterations). See haplo.em.control for more details.

Details

Value

list with components:

`converge`	indicator of convergence of the EM algorithm (1 = converge, 0 = failed).

`lnlike` value of lnlike at last EM iteration (maximum lnlike if converged).

`lr` likelihood ratio statistic to test the final lnlike against the lnlike that assumes complete linkage equilibrium among all loci (i.e., haplotype frequencies are products of allele frequencies).

`df.lr` degrees of freedom for likelihood ratio statistic. The df for the unconstrained final model is the number of non-zero haplotype frequencies minus 1, and the df for the null model of complete linkage equilibrium is the sum, over all loci, of (number of alleles - 1). The df for the lr statistic is df[unconstrained] - df[null]. This can result in negative df, if many haplotypes are estimated to have zero frequency, or if a large amount of trimming occurs, when using large values of min.posterior in the list of control parameters.

`hap.prob` vector of mle's of haplotype probabilities. The ith element of hap.prob corresponds to the ith row of haplotype.

`locus.label` vector of labels for loci, of length K (see definition of input values).

`subj.id` vector of id's for subjects used in the analysis, based on row number of input geno matrix. If subjects are removed, then their id will be missing from subj.id.

`rows.rem` now defunct, but set equal to a vector of length 0, to be compatible with other functions that check for rows.rem.

`indx.subj` vector for row index of subjects after expanding to all possible pairs of haplotypes for each person. If indx.subj=i, then i is the ith row of geno. If the ith subject has n possible pairs of haplotypes that correspond to their marker genotype, then i is repeated n times.

`nreps` vector for the count of haplotype pairs that map to each subject's marker genotypes.

`max.pairs` vector of maximum number of pairs of haplotypes per subject that are consistent with their marker data in the matrix geno. The length of max.pairs = nrow(geno). This vector is computed by geno.count.pairs.

`hap1code` vector of codes for each subject's first haplotype. The values in hap1code are the row numbers of the unique haplotypes in the returned matrix haplotype.

`hap2code` similar to hap1code, but for each subject's second haplotype.

`post` vector of posterior probabilities of pairs of haplotypes for a person, given their marker phenotypes.

`haplotype` matrix of unique haplotypes. Each row represents a unique haplotype, and the number of columns is the number of loci.

`control` list of control parameters for algorithm. See haplo.em.control

Side Effects

References

The basis of this progressive insertion algorithm is from the sofware snphap by David Clayton. Although some of the features and control parameters of this S-PLUS version are modeled after snphap, there are substantial differences, such as extension to allow for more than two alleles per locus, and some other nuances on how the alogrithm is implemented.

See Also

haplo.em.control

Examples

```
setupData(hla.demo)
attach(hla.demo)
geno <- hla.demo[,c(17,18,21:24)]
label <-c("DQB","DRB","B")
keep <- !apply(is.na(geno) | geno==0, 1, any)

save.em.keep <- haplo.em(geno=geno[keep,], locus.label=label)

# warning: output will not exactly match

print.haplo.em(save.em.keep)
```

`haplo.em.control` *Create the Control Parameters for the EM Computation of Haplotype Probabilities, with Progressive Insertion of Loci*

Description

This function creates a list of parameters that control the EM algorithm based on progressive insertion of loci. Non-default parameters for the EM algorithm can be set as parameters passed to haplo.em.control.

Usage

```
haplo.em.control(loci.insert.order=NULL, insert.batch.size = 6,
                                min.posterior = 1e-07, tol = 1e-05,
                                max.iter=500, random.start=0, n.try = 10,
                                iseed=NULL, max.haps.limit = 2e6, verbose=0)
```

Arguments

`loci.insert.order`
: Numeric vector with specific order to insert the loci. If this value is NULL, the insert oder will be in sequential order (1, 2, ..., No. Loci).

`insert.batch.size`
: Number of loci to be inserted in a single batch.

`min.posterior`
: Minimum posterior probability of haplotype pair, conditional on observed marker genotypes. Posteriors below this minimum value will have their pair of haplotypes "trimmed" off the list of possible pairs.

`tol`
: Default 1e-5

`max.iter`
: Maximum number of iterations allowed for the EM algorithm before it stops and prints an error. Default is 500.

random.start If random.start = 0, then the inititial starting values of the posteriors for the first EM attempt will be based on assuming equal posterior probabilities (conditional on genotypes). If random.start = 1, then the initial starting values of the first EM attempt will be based on assuming a uniform distribution for the initial posterior probabilities.

n.try Number of times to try to maximize the lnlike by the EM algorithm. The first try will use, as initial starting values for the posteriors, either equal values or uniform random variables, as determined by random.start. All subsequent tries will use uniform random values as initial starting values for the posterior probabilities.

iseed An integer or a saved copy of .Random.seed. This allows simulations to be reproduced by using the same initial seed.

max.haps.limit The maximum number of haplotypes for which memory is allocated.

verbose Logical, if [T]rue, print lots of debug messages to the screen. If [F]alse, default, do not print any messages. It is best to use verbose=F.

Details

The default is to use n.try = 10. If this takes too much time, it may be worthwhile to decrease n.try. Other tips for computing haplotype frequencies for a large number of loci, particularly if some have many alleles, is to decrease the batch size (insert.batch.size), increase the memory (max.haps.limit).

Value

A list of the parameters passed to the function.

Side Effects

References

See Also

haplo.em, haplo.score

Examples

```
# This is how it is used within haplo.score
#    > score.gauss <- haplo.score(resp, geno, trait.type="gaussian",
#    >           em.control=haplo.em.control(insert.batch.size = 2, n.try=1))
```

haplo.em.fitter *Compute engine for haplotype EM algorithm*

Description

For internal use within the haplo.stats library

Usage

```
haplo.em.fitter(n.loci, n.subject, weight, geno.vec, n.alleles,
                max.haps, max.iter, loci.insert.order, min.posterior,
                tol, insert.batch.size, random.start, iseed1, iseed2,
                iseed3, verbose)
```

Arguments

`n.loci`

`n.subject`

`weight`

`geno.vec`

`n.alleles`

`max.haps`

`max.iter`

`loci.insert.order`

`min.posterior`

`tol`

`insert.batch.size`

`random.start`

`iseed1`

`iseed2`

`iseed3`

`verbose`

Details

For internal use within the haplo.stats library

Value

Side Effects

References

See Also

Examples

`haplo.enum` *Enumerate all possible pairs of haplotypes that are consistent with a set of un-phased multilocus markers*

Description

Given subject un-phased genotype hmat, enumerate all possible pairs of haplotypes, and return enumerated pairs in matrices h1 and h2.

Usage

```
haplo.enum(hmat)
```

Arguments

`hmat`	A genotype vector of length 2*K (K = number of loci). When used in haplo.em, it is a single row of a genotype matrix.

Details

For a pair of haplotypes, if there are H sites that are heterozygous, then there are 2 raised to (H-1) possible pairs to enumerate. To achieve this, the algorithm moves across the loci that are heterozygous (after the 1st heterozygous locus), flipping alleles at heterozygous locations to enumerate all possible pairs of haplotpes, and appending results as rows of the output matrices h1, and h2.

Value

List with components:

`h1`	A matrix of enumerated haplotypes. If there are N enumerations, h1 will have dimension N x K.
`h2`	Similar to h1, a matrix of enumerated haplotypes for the second members of the pairs of haplotypes. Haplotype pairs in h1 and h2 match by the same row number.

Side Effects

References

See Also

haplo.em

Examples

---

haplo.glm *GLM Regression of Trait on Ambiguous Haplotypes*

Description

Perform glm regression of a trait on haplotype effects, allowing for ambiguous haplotypes. This method performs an iterative two-step EM, with the posterior probabilities of pairs of haplotypes per subject used as weights to update the regression coefficients, and the regression coefficients used to update the posterior probabilities.

Usage

```
haplo.glm(formula=formula(data), family=gaussian, data=sys.parent(),
          weights, na.action="na.geno.keep", start=eta, miss.val=c(0,NA),
          locus.label=NA, allele.lev=NULL, control=haplo.glm.control(),
          method="glm.fit", model=FALSE, x=FALSE, y=TRUE,
          contrasts=NULL, ...)
```

Arguments

`formula` a formula expression as for other regression models, of the form response predictors. For details, see the documentation for lm and formula.

`family` a family object. This is a list of expressions for defining the link, variance function, initialization values, and iterative weights for the generalized linear model. Supported families are: gaussian, binomial, poisson. Currently, only the logit link is implemented for binimial.

`data` a data frame in which to interpret the variables occurring in the formula. A CRITICAL element of the data frame is the matrix of genotypes, denoted here as "geno", although an informative name should be used in practice. This geno matrix is actually a matrix of alleles, such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a chromosome. If there are K loci, then ncol(geno) = 2*K. Rows represent the alleles for each subject. It is also CRITICAL that this matrix is defined as a model.matrix, in order to keep the columns of the matrix packaged together into the single matrix object. If geno is a matrix of alleles, then before adding it to the data frame, use the following command to convert it to a model.matrix: oldClass(geno) <- "model.matrix". If geno is a data.frame of alleles, you must first convert geno to a matrix, using geno <- as.matrix(geno), and then convert it to a model.matrix.

`weights`	the weights for observations (rows of the data frame). By default, all observations are weighted equally.
`na.action`	a function to filter missing data. This is applied to the model.frame. The default value of na.action=na.geno.keep will keep observations with missing alleles, but exclude observations missing any other data (e.g., response variable, other covariates, weight). The EM algorithm for ambiguous haplotypes accounts for missing alleles. Similar to the usual glm, na.fail creates an error if any missing values are found, and a third possible alternative is na.exclude, which deletes observations that contain one or more missing values for any data, including alleles.
`start`	a vector of initial values on the scale of the linear predictor.
`miss.val`	vector of values that represent missing alleles in geno matrix.
`locus.label`	vector of labels for loci.
`allele.lev`	This argument is optional ONLY for S-PLUS, but is REQUIRED for R. This is a list of vectors, each vector giving the labels of alleles for each locus. The list is made an attribute of geno<-setupGeno(geno). This is required to account for the differences in which S-PLUS and R handle character data (allele labels) in a model.frame. See its use in the example below.
`control`	list of control parameters. The default is constructed by the function haplo.glm.control. The items in this list control the regression modeling of the haplotypes (e.g., additive, dominant, recessive effects of haplotypes; which haplotype is chosen as the baseline for regression; how to handle rare haplotypes; control of the glm function - maximum number of iterations), and the EM algorithm for estimating initial haplotype frequencies. See haplo.glm.control for details.
`method`	currently, glm.fit is the only method allowed.
`model`	if model=TRUE, the model.frame is returned.
`x`	a logical flag. If x=TRUE, the model.matrix is returned. By default, x=FALSE.
`y`	a logical flag. The default value of y=TRUE causes the response variable to be returned.
`contrasts`	currently, contrasts is ignnored (so NULL, the default value, is always used).
`...`	potential other arguments that may be passed - currently ignored.

Details

Value

An object of class "haplo.glm" is returned. The output object from haplo.glm has all the components of a glm object, with a few more. It is important to note that some of the returned components correpond to the "expanded" version of the data. This means that each observation is expanded into the number of terms in the observation's posterior distribution of haplotype pairs, given the marker data. For example, when fitting the response y on haplotype effects, the value of y[i], for the ith observation, is replicated m[i] times, where m[i] is the number of pairs of haplotypes consistent with the observed marker data. The returned components that are expanded are indicated below by [expanded] in the definition of the component. These expanded components may need to be collapsed, depending on the user's objectives. For example, when considering the influence of an observation, it may make sense to examine the expanded residuals for a single observation, perhaps plotted against the haplotypes for that observation. In contrast, it would not be sensible to plot all residuals against non-genetic covaraites, without first collapsing the expanded residuals for each observation. To collapse, one can use the average residual per observation, weighted according to the posterior probabilities. The appropriate weight can be computed as wt = fit$weight.expanded * fit$haplo.post.info$post. Then, the weighted average can be calculated as tapply(fit$residuals * wt, fit$haplo.post.info$indx, sum).

`coefficients` the coefficients of the linear.predictors, which multiply the columns of the model matrix. The names of the coefficients are the names of the columns of the model matrix. For haplotype coefficients, the names are the concatentation of name of the geno matrix with a haplotype number. The haplotype number corresponds to the index of the haplotype. The default print will show the coefficients with haplotype number, along with the alleles that define the haplotype, and the estimated haplotype frequency. If the model is over-determined there will be missing values in the coefficients corresponding to inestimable coefficients.

`residuals` [expanded] residuals from the final weighted least squares fit; also known as working residuals, these are typically not interpretable without rescaling by the weights (see glm.object).

`fitted.values` [expanded] fitted mean values, obtained by transforming linear.predictors using the inverse link function (see glm.object).

`effects` [expaded] orthogonal, single-degree-of-freedom effects (see lm.object).

`R` the triangular factor of the decomposition (see lm.object).

`rank` the computed rank (number of linearly independent columns in the model matrix), which is the model degrees of freedom - see lm.object.

`assign` the list of assignments of coefficients (and effects) to the terms in the model (see lm.object).

`df.residual` [expanded] number of degrees of freedom for residuals, corresponding to the expanded data.

`weights.expanded`
[expanded] input weights after expanding according to the number of pairs of haplotypes consistent with an observation's marker genotype data.

`family` a 3 element character vector giving the name of the family, the link and the variance function; mainly for printing purposes.

`linear.predictors`
[expanded] linear fit, given by the product of the model matrix and the coefficients; also the fitted.values from the final weighted least squares fit.

`deviance` [expanded] up to a constant, minus twice the maximized log-likelihood. Similar to the residual sum of squares.

`null.deviance` the deviance corresponding to the model with no predictors.

`call` an image of the call that produced the object, but with the arguments all named and with the actual formula included as the formula argument.

`iter` the number of IRLS iterations used to compute the estimates, for the last step of the EM fit of coefficients.

`y` [expanded] response, if y=T.

contrasts a list containing sufficient information to construct the contrasts used to fit any factors occurring in the model (see lm.object).

lnlike log-likelihood of the fitted model.

lnlike.null log-likelihood of the null model that has only an intercept.

lrt likelihood ratio test statistic to test whether all coefficients (excepet intercept) are zero: 2*(lnlike - lnlike.null)

terms an object of mode expression and class term summarizing the formula, but not complete for the final model. Because this does not represent expansion of the design matrix for the haplotypes, it is typically not of direct relevance to users.

control list of all control parameters

haplo.unique the data.frame of unique haplotypes

haplo.base the index of the haplotype used as the base-line for the regression model. To see the actual haplotype definition, use the following: fit$haplo.unique[fit$haplo.base,], where fit is the saved haplo.glm object (e.g., fit <- haplo.glm(y geno, ...)).

haplo.freq the final estimates of haplotype frequencies, after completing EM steps of updating haplotype frequencies and regression coefficients. The length of haplo.freq is the number of rows of haplo.unique, and the order of haplo.freq is the same as that for the rows of haplo.unique. So, the frequencies of the unique haplotypes can be viewed as cbind(fit$haplo.unique, fit$haplo.freq).

haplo.freq.init the initial estimates of haplotype frequencies, based on the EM algorithm for estimating haplotype frequencies, ingnoring the trait. These can be compared with haplo.freq, to see the impact of using the regression model to update the haplotype frequencies.

converge.em T/F whether the initial EM algorithm for estimating haplo.freq.init converged.

haplo.common the indices of the haplotypes determined to be "common" enough to estimate their corresponding regression coefficients.

haplo.rare the indices of all the haplotypes determined to be too rare to estimate their specific regression coefficients.

haplo.rare.term T/F whether the "rare" term is included in the haplotype regression model.

haplo.names the names of the coefficients that represent haplotype effects.

haplo.post.info a data.frame of information regarding the posterior probabilites. The columns of this data.frame are: indx (the index of the input obsevation; if the ith observation is repeated m times, then indx will show m replicates of i; hence, indx will correspond to the "expanded" observations); hap1 and hap2 (the indices of the haplotypes; if hap1=j and hap2=k, then the two haplotypes in terms of alleles are fit$haplo.unique[j,] and fit$haplo.unique[k,]); post.init (the initial posterior probability, based on haplo.freq.init); post (the final posterior probability, based on haplo.freq).

x the model matrix, with [expanded] rows, if x=T.

`info` the observed information matrix, based on Louis' formula. The upper left submatrix is for the regression coefficient, the lower right submatrix for the haplotype frequencies, and the remaining is the information between regression coefficients and haplotype frequencies.

`var.mat` the variance-covariance matrix of regression coefficients and haplotype frequencies, based on the inverse of info. Upper left submatrix is for regression coefficients, lower right submatrix for haplotype frequencies.

`haplo.elim` the indices of the haplotypes eliminated from the info and var.mat matrices because their frequencies are less than haplo.min.info (the minimum haplotype frequency required for computation of the information matrix - see haplo.glm.control)

`rank.info` rank of information (info) matrix.

References

Lake S, Lyon H, Silverman E, Weiss S, Laird N, Schaid D (2002) Estimation and tests of haplotype-environment interaction when linkage phase is ambiguous. Human Heredity 55:56-65.

See Also

haplo.glm.control, haplo.em, haplo.model.frame

Examples

```
 setupData(hla.demo)
 geno <- as.matrix(hla.demo[,c(17,18,21:24)])
 keep <- !apply(is.na(geno) | geno==0, 1, any)
 hla.demo <- hla.demo[keep,]
 geno <- geno[keep,]
 attach(hla.demo)
 label <-c("DQB","DRB","B")
 y <- hla.demo$resp
 y.bin <- 1*(hla.demo$resp.cat=="low")

# set up a genotype array as a model.matrix for inserting into data frame
# Note that hla.demo is a data.frame, and we need to subset to columns
# of interest. Also also need to convert to a matrix object, so that
# setupGeno can code alleles and convert geno to 'model.matrix' class.

 geno <- setupGeno(geno, miss.val=c(0,NA))

  # geno now has an attribute 'unique.alleles' which must be passed to
  # haplo.glm as allele.lev=attributes(geno)$unique.alleles, see below

 my.data <- data.frame(geno=geno, age=hla.demo$age, male=hla.demo$male,
                      y=y, y.bin=y.bin)

 fit.gaus <- haplo.glm(y ~ male + geno, family = gaussian,  na.action=
              "na.geno.keep",allele.lev=attributes(geno)$unique.alleles,
              data=my.data, locus.label=label,
              control = haplo.glm.control(haplo.freq.min=0.02))
 fit.gaus
```

`haplo.glm.control` *Create list of control parameters for haplo.glm*

Description

Create a list of control pararameters for haplo.glm. If no parameters are passed to this function, then all default values are used.

Usage

```
haplo.glm.control(haplo.effect="add", haplo.base=NULL,haplo.freq.min=0.001,
                  sum.rare.min=0.001, haplo.min.info=0.001,
                  keep.rare.haplo=TRUE, glm.c=glm.control(maxit=500),
                                                em.c=haplo.em.control())
```

Arguments

`haplo.effect` the "effect" of a haplotypes, which determines the covariate (x) coding of haplotypes. Valid options are "additive" (causing x = 0, 1, or 2, the count of a particular haplotype), "dominant" (causing x = 1 if heterozygous or homozygous carrier of a particular haplotype; x = 0 otherwise), and "recessive" (causing x = 1 if homozygous for a particular haplotype; x = 0 otherwise).

`haplo.base` the index for the haplotype to be used as the base-line for regression. By default, haplo.base=NULL, so that the most frequent haplotype is chosen as the base-line.

`haplo.freq.min` the minimum haplotype frequency for a haplotype to be included in the regression model as its own effect. The haplotype frequency is based on the EM algorithm that estimates haplotype frequencies independent of trait.

`sum.rare.min` the sum of the "rare" haplotype frequencies must be larger than sum.rare.min in order for the pool of rare haplotypes to be included in the regression model as a separate term. If this condition is not met, then the rare haplotypes are pooled with the base-line haplotype (see keep.rare.haplo below).

`haplo.min.info` the minimum haplotype frequency for determining the contribution of a haplotype to the observed information matrix. Haplotypes with less frequency are dropped from the observed information matrix. The haplotype frequency is that from the final EM that iteratively updates haplotype frequencies and regression coefficients.

`keep.rare.haplo` TRUE/FALSE to determine if the pool of rare haplotype should be kept as a separate term in the regression model (when keep.rare.haplo=TRUE), or pooled with the base-line haplotype (when keep.rare.haplo=FALSE).

`glm.c` list of control parameters for the usual glm.control (see glm.control).

`em.c` list of control parameters for the EM algorithm to estimate haplotype frequencies, independent of trait (see haplo.em.control).

Value

the list of above components

See Also

glm.control, haplo.em.control

Examples

```
# using the data set up in the example for haplo.glm,
# the control function is used in haplo.glm as follows
#  > fit <- haplo.glm(y ~ male + geno, family = gaussian,
#  >          na.action="na.geno.keep",
#  >          data=my.data, locus.label=locus.label,
#  >          control = haplo.glm.control(haplo.freq.min =
#  >          0.02,em.c=haplo.em.control(n.try=1)))
```

`haplo.group` *Frequencies for Haplotypes by Grouping Variable*

Description

Calculate maximum likelihood estimates of haplotype probabilities for the entire dataset and separately for each subset defined by the levels of a group variable. Only autosomal loci are considered.

Usage

```
haplo.group(group, geno, locus.label=NA, miss.val=0,
            control=haplo.em.control())
```

Arguments

`group`	Group can be of logical, numeric, character, or factor class type.
`geno`	Matrix of alleles, such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a chromosome. If there are K loci, then geno has 2*K columns. Rows represent all observed alleles for each subject.
`locus.label`	Vector of labels for loci, of length K (see definition of geno matrix).
`miss.val`	Vector of codes for allele missing values.
`control`	list of control parameters for haplo.em (see haplo.em.control).

Details

Haplo.em is used to compute the maximum likelihood estimates of the haplotype frequencies for the total sample, then for each of the groups separately.

Value

A list as an object of the haplo.group class. The three elements of the list are described below.

`group.df` A data frame with the columns described as follows. -haplotype: Names for the K columns for the K alleles in the haplotypes. -total: Estimated frequencies for haplotypes from the total sample. -group.name.i: Estimated haplotype frequencies for the haplotype if it occurs in the group referenced by 'i'. Frequency is NA if it doesn't occur for the group. The column name is the actual variable name joined with the ith level of that variable.

`group.count` Vector containing the number of subjects for each level of the grouping variable.

`n.loci` Number of loci occuring in the geno matrix.

Side Effects

References

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. "Score tests for association of traits with haplotypes when linkage phase is ambiguous." Amer J Hum Genet. 70 (2002): 425-434.

See Also

print.haplo.group, haplo.em

Examples

```
  setupData(hla.demo)
  geno <- as.matrix(hla.demo[,c(17,18,21:24)])

# remove any subjects with missing alleles for faster examples,
# but you may keep them in practice
  keep <- !apply(is.na(geno) | geno==0, 1, any)
  hla.demo <- hla.demo[keep,]
  geno <- geno[keep,]
  attach(hla.demo)

  y.ord <- as.numeric(resp.cat)
  y.bin <-ifelse(y.ord==1,1,0)
  group.bin <- haplo.group(y.bin, geno, miss.val=0)
  print.haplo.group(group.bin)
```

`haplo.hash` *Integer Rank Codes for Haplotypes*

Description

Create a vector of integer codes for the input matrix of haplotypes. The haplotypes in the input matrix are converted to character strings, and if there are C unique strings, the integer codes for the haplotypes will be 1, 2, ..., C.

Usage

```
haplo.hash(hap)
```

Arguments

`hap`	A matrix of haplotypes. If there are N haplotypes for K loci, hap have dimensions N x K.

Details

The alleles that make up each row in hap are pasted together as character strings, and the unique strings are sorted so that the rank order of the sorted strings is used as the integer code for the unique haplotypes.

Value

List with elements:

`hash`	Vector of integer codes for the input data (hap). The value of hash is the row number of the unique haplotypes given in the returned matrix hap.mtx.
`hap.mtx`	Matrix of unique haplotypes.

Side Effects

References

See Also

haplo.em

Examples

`haplo.model.frame` *Sets up a model frame for haplo.glm*

Description

For internal use within the haplo.stats library

Usage

```
haplo.model.frame(m, locus.label=NA, allele.lev=NULL, miss.val=c(0,NA),
                  control=haplo.glm.control())
```

Arguments

`m`

`locus.label`

`allele.lev`

`miss.val`

`control`

Details

Value

Side Effects

References

See Also

Examples

`haplo.score` *Score Statistics for Association of Traits with Haplotypes*

Description

Compute score statistics to evaluate the association of a trait with haplotypes, when linkage phase is unknown and diploid marker phenotypes are observed among unrelated subjects. For now, only autosomal loci are considered.

Usage

```
haplo.score(y, geno, trait.type="gaussian", offset = NA,
            x.adj = NA, skip.haplo=.005, locus.label=NA,
            miss.val=c(0,NA), simulate=FALSE, sim.control=score.sim.control(),
            em.control=haplo.em.control())
```

Arguments

`y` Vector of trait values. For trait.type = "binomial", y must have values of 1 for event, 0 for no event.

`geno` Matrix of alleles, such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a chromosome. If there are K loci, then ncol(geno) = 2*K. Rows represent alleles for each subject.

`trait.type` Character string defining type of trait, with values of "gaussian", "binomial", "poisson", "ordinal".

`offset` Vector of offset when trait.type = "poisson"

`x.adj` Matrix of non-genetic covariates used to adjust the score statistics. Note that intercept should not be included, as it will be added in this function.

`skip.haplo` Skip score statistics for haplotypes with frequencies < skip.haplo

`locus.label` Vector of labels for loci, of length K (see definition of geno matrix)

`miss.val` Vector of codes for missing values of alleles

`simulate` Logical: if [F]alse, no empirical p-values are computed; if [T]rue, simulations are performed. Specific simulation parameters can be controlled in the sim.control parameter list.

`sim.control` A list of control parameters to determine how simulations are performed for simulated p-values. The list is created by the function score.sim.control and the default values of this function can be changed as desired. See score.sim.control for details.

`em.control` A list of control parameters to determine how to perform the EM algorithm for estimating haplotype frequencies when phase is unknown. The list is created by the function haplo.em.control - see this function for more details.

Details

Compute the maximum likelihood estimates of the haplotype frequencies and the posterior probabilities of the pairs of haplotypes for each subject using an EM algorithm. The algorithm begins with haplotypes from a subset of the loci and progressively discards those with low frequency before inserting more loci. The process is repeated until haplotypes for all loci are established. The posterior probabilities are used to compute the score statistics for the association of (ambiguous) haplotypes with traits. The glm function is used to compute residuals of the regression of the trait on the non-genetic covariates.

Value

List with the following components:

`score.global` Global statistic to test association of trait with haplotypes that have frequencies >= skip.haplo.

`df` Degrees of freedom for score.global.

`score.global.p`
P-value of score.global based on chi-square distribution, with degrees of freedom equal to df.

`score.global.p.sim`
P-value of score.global based on simulations (set equal to NA when simulate=F).

`score.haplo` Vector of score statistics for individual haplotypes that have frequencies >= skip.haplo.

`score.haplo.p` Vector of p-values for score.haplo, based on a chi-square distribution with 1 df.

`score.haplo.p.sim` Vector of p-values for score.haplo, based on simulations (set equal to NA when simulate=F).

`score.max.p.sim` P-value of maximum score.haplo, based on simulations (set equal to NA when simulate=F).

`haplotype` Matrix of hapoltypes analyzed. The ith row of haplotype corresponds to the ith item of score.haplo, score.haplo.p, and score.haplo.p.sim.

`hap.prob` Vector of haplotype probabilies, corresponding to the haplotypes in the matrix haplotype.

`locus.label` Vector of labels for loci, of length K (same as input argument).

`simulate` Same as function input parameter. If [T]rue, simulation results are included in the haplo.score object.

`n.val.global` Vector containing the number of valid simulations used in the global score statistic simulation. The number of valid simulations can be less than the number of simulations requested (by sim.control) if simulated data sets produce unstable variances of the score statistics.

`n.val.haplo` Vector containing the number of valid simulations used in the p-value simulations for maximum-score statistic and scores for the individual haplotypes.

Side Effects

References

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. "Score tests for association of traits with haplotypes when linkage phase is ambiguous." Amer J Hum Genet. 70 (2002): 425-434.

See Also

`haplo.em`, `plot.haplo.score`, `print.haplo.score`, `haplo.em.control`, `score.sim.control`

Examples

```
# establish all hla.demo data, remove genotypes with missing alleles
# so haplo.score runs faster

setupData(hla.demo)
geno <- as.matrix(hla.demo[,c(17,18,21:24)])
keep <- !apply(is.na(geno) | geno==0, 1, any)
hla.demo <- hla.demo[keep,]
geno <- geno[keep,]
attach(hla.demo)
label <- c("DQB","DRB","B")
```

```
# For quantitative, normally distributed trait:

  score.gaus <- haplo.score(resp, geno, locus.label=label,
                            trait.type = "gaussian")
  print(score.gaus)

# For ordinal trait:
  y.ord <- as.numeric(resp.cat)
  score.ord <- haplo.score(y.ord, geno, locus.label=label,
                           trait.type="ordinal")
  print(score.ord)

# For a  binary trait and simulations,
# limit simulations to 500 in score.sim.control, default is 20000
  y.bin <-ifelse(y.ord==1,1,0)
  score.bin.sim <- haplo.score(y.bin, geno, trait.type = "binomial",
                   locus.label=label, simulate=TRUE, sim.control=
                   score.sim.control(min.sim=200,max.sim=500))

  print(score.bin.sim)

# For a binary trait, adjusted for sex and age:

  x <- cbind(male, age)
  score.bin.adj <- haplo.score(y.bin, geno, trait.type = "binomial",
                              locus.label=label, x.adj=x)
  print(score.bin.adj)
```

`haplo.score.glm` *Compute haplotype score statistics for GLM*

Description

This function is used by haplo.score when analyzing traits by a GLM score.

Usage

```
haplo.score.glm(y, mu, a, v, x.adj, nreps, x.post, post, x)
```

Arguments

`y`	Vector of trait values.
`mu`	Expected value of y.
`a`	scale parameter
`v`	v= b"/a for a GLM.
`x.adj`	Matrix of non-genetic covariates used to adjust the score statistics. Note that intercept should be included in this matrix.
`nreps`	Vector for the count of haplotype pairs that map to each subject's marker genotypes (see haplo.em).
`x.post`	Matrix for posterior mean of x per subject.
`post`	Vector of posterior probabilities of pairs of haplotypes for a person, given thier marker phenotypes (see haplo.em).

`x`	Matrix of scores for enumerated haplotypes for each subject, with elements 0, 1, 2 (counts of specific haplotypes).

None.

Details

Using posterior probabilities of pairs of haplotypes, the "design" matrix for the haplotype effects, and the GLM residuals, compute the score vector and its variance matrix, adjusted for the non-genetic covariates.

Value

List with components:

`u.score`	Vector of scores for the chosen haplotypes
`v.score`	Covariance matrix for u.score

Side Effects

References

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. Score tests for association of traits with haplotypes when linkage phase is ambiguous. Submitted to Amer J Hum Genet.

See Also

haplo.score

Examples

`haplo.score.merge` *Merge haplo.score And haplo.group Objects*

Description

Combine information from returned objects of haplo.score and haplo.group, 'score' and 'group' respectively. 'score' and 'group' are sorted differently and 'score' keeps a subset of all the haplotypes while 'group' has all of them. To combine results from the two objects, merge them by haplotype and sort by score of the haplotype. The merged object includes all haplotypes; i.e. those appearing in 'group', but the print default only shows haplotypes which have a score.

Usage

```
haplo.score.merge(score, group)
```

Arguments

`score`	Object returned from haplo.score of class "haplo.score".
`group`	Object returned from haplo.group of class "haplo.group".

Details

Haplo.score returns score statistic and p-value for haplotypes with an overall frequency above the user-specified threshold, skip.haplo. For haplotypes with frequencies below the threshold, the score and p-value will be NA. Overall haplotype frequencies and for sub-groups are estimated by haplo.group.

Value

Data frame including haplotypes, score-statistics, score p-value, estimated haplotype frequency for all subjects, and haplotype frequency from group subsets.

Side Effects

Warning: The merge will not detect if the group and score objects resulted from different subject phenotypes selected by memory-usage parameters, rm.geno.na and enum.limit. Users must use the same values for these parameters in haplo.score and haplo.group so the merged objects are consistent.

See Also

`haplo.score`, `haplo.group`

Examples

```
setupData(hla.demo)
geno <- as.matrix(hla.demo[,c(17,18,21:24)])
keep <- !apply(is.na(geno) | geno==0, 1, any)
hla.demo <- hla.demo[keep,]
geno <- geno[keep,]
attach(hla.demo)
y.ord <- as.numeric(resp.cat)
y.bin <-ifelse(y.ord==1,1,0)

group.bin <- haplo.group(y.bin, geno, miss.val=0)
score.bin <- haplo.score(y.bin, geno, trait.type="binomial")
score.merged <- haplo.score.merge(score.bin, group.bin)

print(score.merged)
```

`haplo.score.podds` *Compute Haplotype Score Statistics for Ordinal Traits with Proportional Odds Model*

Description

This function is used by haplo.score when analyzing ordinal traits by a proportional odds model score statistic.

Usage

```
haplo.score.podds(y, alpha, beta=NA, x.adj=NA, nreps, x.post, post, x)
```

Arguments

`y`	Vector of ordinal trait values.
`alpha`	Intercept parameters for ordinal logistic regression model.
`beta`	Regression parameters for adjusted covariates (x.adj).
`x.adj`	Matrix of non-genetic covariates used to adjust the score statistics. Note that intercept should NOT be included in this matrix.
`nreps`	Vector for the count of haplotype pairs that map to each subject's marker genotypes (see haplo.em).
`x.post`	Matrix for posterior mean of x per subject.
`post`	Vector of posterior probabilities of pairs of haplotypes for a person, given thier marker phenotypes (see haplo.em).
`x`	Matrix of scores for enumerated haplotypes for each subject, with elements 0, 1, 2 (counts of specific haplotypes).

None.

Details

Using posterior probabilities of pairs of haplotypes, the "design" matrix for the haplotype effects, and the proportional odds model, compute the score vector and its variance matrix, adjusted for the non-genetic covariates.

Value

List with components:

`u.score`	Vector of scores for the chosen haplotypes
`v.score`	Covariance matrix for u.score

Side Effects

Warning

To analyze an ordinal trait with adjustment for x.adj covariates, the user will need to have Frank Harrell's librarys (Design and Hmisc). However, the unadjusted ordinal trait works fine without these libraries.

References

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. Score tests for association of traits with haplotypes when linkage phase is ambiguous. Submitted to Amer J Hum Genet.

See Also

haplo.score

Examples

`haplo.score.slide` *Score Statistics for Association of Traits with Haplotypes*

Description

Used to identify sub-haplotypes from a group of loci. Run haplo.score on all contiguous subsets of size n.slide from the loci in a genotype matrix (geno). From each call to haplo.score, report the global score statistic p-value. Can also report global and maximum score statistics simulated p-values.

Usage

```
haplo.score.slide(y, geno, trait.type="gaussian", n.slide=2,
                  offset = NA, x.adj = NA, skip.haplo=.005,
                  locus.label=NA, miss.val=c(0,NA),
                  simulate=FALSE, sim.control=score.sim.control(),
                  em.control=haplo.em.control())
```

Arguments

`y`	Vector of trait values. For trait.type = "binomial", y must have values of 1 for event, 0 for no event.
`geno`	Matrix of alleles, such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a chromosome. If there are K loci, then ncol(geno) = 2*K. Rows represent alleles for each subject.
`trait.type`	Character string defining type of trait, with values of "gaussian", "binomial", "poisson", "ordinal".
`n.slide`	Number of loci in each contiguous subset. The first subset is the ordered loci numbered 1 to n.slide, the second subset is 2 through n.slide+1 and so on. If the total number of loci in geno is n.loci, then there are n.loci - n.slide + 1 total subsets.
`offset`	Vector of offset when trait.type = "poisson"
`x.adj`	Matrix of non-genetic covariates used to adjust the score statistics. Note that intercept should not be included, as it will be added in this function.
`skip.haplo`	Skip score statistics for haplotypes with frequencies < skip.haplo
`locus.label`	Vector of labels for loci, of length K (see definition of geno matrix).
`miss.val`	Vector of codes for missing values of alleles.
`simulate`	Logical, if [F]alse (default) no empirical p-values are computed. If [T]rue simulations are performed. Specific simulation parameters can be controlled in the sim.control parameter list.
`sim.control`	A list of control parameters used to perform simulations for simulated p-values in haplo.score. The list is created by the function score.sim.control and the default values of this function can be changed as desired.

`em.control` A list of control parameters used to perform the em algorithm for estimating haplotype frequencies when phase is unknown. The list is created by the function haplo.em.control and the default values of this function can be changed as desired.

Details

Haplo.score.slide is useful for a series of loci where little is known of the association between a trait and haplotypes. Using a range of n.slide values, the region with the strongest association will consistently have low p-values for locus subsets containing the associated haplotypes. The global p-value measures significance of the entire set of haplotypes for the locus subset. Simulated maximum score statistic p-values indicate when one or a few haplotypes are associated with the trait.

Value

List with the following components:

`df` Data frame with start locus, global p-value, simulated global p-value, and simulated maximum-score p-value.

`n.loci` Number of loci given in the genotype matrix.

`simulate` Same as parameter description above.

`n.slide` Same as parameter description above.

`locus.label` Same as parameter description above.

`n.val.haplo` Vector containing the number of valid simulations used in the maximum-score statistic p-value simulation. The number of valid simulations can be less than the number of simulations requested (by sim.control) if simulated data sets produce unstable variables of the score statistics.

`n.val.global` Vector containing the number of valid simulations used in the global score statistic p-value simulation.

Side Effects

References

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. "Score tests for association of traits with haplotypes when linkage phase is ambiguous." Amer J Hum Genet. 70 (2002): 425-434.

See Also

`haplo.score`, `plot.haplo.score.slide`, `score.sim.control`

Examples

```
  setupData(hla.demo)

# Continuous trait slide by 2 loci on all 11 loci, uncomment to run it.
# Takes > 20 minutes to run
#  geno.11 <- hla.demo[,-c(1:4)]
#  label.11 <- c("DPB","DPA","DMA","DMB","TAP1","TAP2","DQB","DQA","DRB","B","A")
```

```
#  slide.gaus <- haplo.score.slide(resp, geno.11, trait.type = "gaussian",
#                                  locus.label=label.11, n.slide=2)

#  print(slide.gaus)
#  plot(slide.gaus)

# Run shortened example on 9 loci
# For an ordinal trait, slide by 3 loci, and simulate p-values:
  geno.9 <- hla.demo[,-c(1:6,15,16)]
  label.9 <- c("DPA","DMA","DMB","TAP1","DQB","DQA","DRB","B","A")

  y.ord <- as.numeric(hla.demo$resp.cat)

# data is set up, to run, run these lines of code on the data that was
# set up in this example. It takes > 15 minutes to run
#  slide.ord.sim <-  haplo.score.slide(y.ord, geno.9, trait.type = "ordinal",
#                      n.slide=3, locus.label=label.9, simulate=TRUE,
#                      sim.control=score.sim.control(min.sim=200, max.sim=500))

  # note, results will vary due to simulations
#  print(slide.ord.sim)
#  plot(slide.ord.sim)
#  plot(slide.ord.sim, pval="global.sim")
#  plot(slide.ord.sim, pval="max.sim")
```

`hla.demo` *HLA Loci and Serologic Response to Measles Vaccination.*

Description

Eleven HLA-region loci genotyped for 220 subjects, phase not known. Contains measles vaccination response with covariate data.

Usage

```
data(hla.demo)
```

Format

Data Frame with the following columns:

resp Quantitative response to Measles Vaccination

resp.cat Category of response as low, normal, or high; based on 'resp'

male Binary indicator of gender, 1=male, 0=female

age Age of the subject

allele columns 5 - 26 Pairs of columns represent the allele pairs for each subject at the locus.

References

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. "Score tests for association of traits with haplotypes when linkage phase is ambiguous." Amer J Hum Genet. 70 (2002): 425-434.

Source

Data set kindly provided by Gregory A. Poland, M.D. and the Mayo Clinic Vaccine Research Group for illustration only, and my not be used for publication.

`locator.haplo` *Find Location from Mouse Clicks and Print Haplotypes on Plot*

Description

Much like the Splus locator() is used to find x-y coordinates on a plot, locator.haplo() finds all x-y coordinates that are clicked on by a user, and then prints haplotypes at the chosen positions.

Usage

```
locator.haplo(obj)
```

Arguments

`obj`	An object (of class haplo.score) which contains the analysis results that are returned from the function haplo.score.

Details

After plotting the results in obj, as from plot(obj), the function locator.haplo is used to place on the plot the text strings for haplotypes of interest. After the function call (e.g., locator.haplo(obj)), the user can click, with the left mouse button, on as many points in the plot as desired. Then, clicking with the middle mouse button will cause the haplotypes to be printed on the plot. The format of a haplotype is "a:b:c", where a, b, and c are alleles, and the separator ":" is used to separate alleles on a haplotype. The algorithm chooses the closest point that the user clicks on, and prints the haplotype either above the point (for points on the lower-half of the plot) or below the point (for points in the upper-half of the plot).

Value

List with the following components:

`x.coord`	Vector of x-coordinates.
`y.coord`	Vector of y-coordinates.
`hap.txt`	Vector of character strings for haplotypes.

See Also

haplo.score

Examples

```
# follow the pseudo-code
#  score.out <-  haplo.score(y, geno, trait.type = "gaussian")

#  plot(score.out)

#  locator.haplo(score.out)
```

---

loci — *Create a group of locus objects from a genotype matrix, assign to 'model.matrix' class.*

Description

The function makes each pair of columns a locus object, which recodes alleles to numeric and saves the original alleles as an attribute of the model.matrix.

Usage

```
loci(geno, locus.names, chrom.label=NULL, x.linked=FALSE, sex=NULL,
     male.code="M", female.code="F", miss.val=NA, map=NA)
```

Arguments

`geno`	Matrix of alleles, such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a chromosome. If there are K loci, then ncol(geno) = 2*K. Rows represent alleles for each subject.
`locus.names`	A vector containing the locus name for each locus.
`chrom.label`	Chromosome Label
`x.linked`	A logical value denoting whether the chromosome is X-linked.
`sex`	A vector containing the sex of each individual. If x.linked=F then argum ent sex is not required and may be left as the default value of NULL.
`male.code`	The code denoting a male in the sex vector.
`female.code`	The code denoting a female in the sex vector.
`miss.val`	A vector of codes denoting missing values for the allele labels. Note that NA will always be treated as a missing value, and alleles matching miss.val are assigned NA. Also note that the original missing value code for a specific individual can not be retrieved from the returned object.
`map`	An optional chromosome map of class "cmap"

Details

Value

An object of class "model.matrix", with all alleles recoded to a numeric value. It contains the following attributes:

- `locus.names` A vector of labels for the loci, of length nloci.
- `map` Will be better defined later.
- `x.linked` A logical value denoting whether the chromosome is X-linked.
- `unique.alleles` The original allele labels are stored in the 'unique.alleles' attribute. The ith item of the unique.alleles list is a vector of unique alleles for the ith locus.
- `male.code` The code denoting a male in the sex vector.
- `female.code` The code denoting a female in the sex vector.
- `chrom.label` Chromosome Label

Side Effects

References

Note

A matrix that contains all elements of mode character will be sorted in alphabetic order.

See Also

locus, setupGeno

Examples

```
# Create some loci to work with
a1 <- 1:6
a2 <- 7:12

b1 <- c("A","A","B","C","E","D")
b2 <-c("A","A","C","E","F","G")

c1 <- c("101","10","115","132","21","112")
c2 <- c("100","101","0","100","21","110")

myloci <- data.frame(a1,a2,b1,b2,c1,c2)
myloci <- loci(myloci, locus.names=c("A","B","C"),miss.val=c(0,NA))
myloci

attributes(myloci)
```

`locus` *Creates an object of class "locus"*

Description

Creates an object containing genotypes for multiple individuals. The object can then use method functions developed for objects of class "locus".

Usage

```
locus(allele1, allele2, chrom.label=NULL,locus.alias=NULL,
     x.linked=FALSE, sex=NULL, male.code="M", female.code="F", miss.val=NA)
```

Arguments

`allele1`	A vector containing the labels for 1 allele for a set of individuals, or optionally a matrix with 2 columns each containing an allele for each person.
`allele2`	A vector containing the labels for the second allele for a set of individuals. If allele 1 is a matrix, allele 2 need not be specified.
`chrom.label`	A label describing the chromosome the alleles belong to
`locus.alias`	A vector containing one or more aliases describing the locus. The first alias in the vector will be used as a label for printing in some functions such as multilocus.print().
`x.linked`	A logical value denoting whether the chromosome is x linked
`sex`	A vector containing the gender of each individual (required if x.linked=T)
`male.code`	The code denoting a male in the sex vector
`female.code`	The code denoting a female in the sex vector
`miss.val`	a vector of codes denoting missing values for allele1 and allele2. Note that NA will always be treated as a missing value, even if not specified in miss.val. Also note that if multiple missing value codes are specified, the original missing value code for a specific individual can not be retrieved from the locus object.

Details

Value

Returns an object of class locus which inherits from class model.matrix containing the following elements:

`geno`	a matrix with 2 columns where each row contains numeric codes for the 2 alleles for an individual.
`chrom.label`	a chromosome label
`locus.alias`	a vector of aliases for the locus
`x.linked`	a logical value specifying if the locus is x-linked or not

`allele.labels` a vector of labels corresponding to the numeric codes in matrix geno (similar to levels in a factor)

`male.code` a code to be used to identify males for an x.linked locus.

`female.code` a code to be used to identify females for an x.linked locus.

Side Effects

References

See Also

Examples

```
b1 <- c("A","A","B","C","E","D")
b2 <- c("A","A","C","E","F","G")
loc1 <- locus(b1,b2,chrom=4,locus.alias="D4S1111")

loc1

# a second example which uses more parameters, some may not be supported.
# c1 <- c("101","10","115","132","21","112")
# c2 <- c("100","101","0","100","21","110")

# gender <- rep(c("M","F"),3)
# loc2 <- locus(c2,c2,chrom="X",locus.alias="DXS1234",x.linked=T,sex=gender)
```

---

louis.info *Louis Information for haplo.glm*

Description

For internal use within the haplo.stats library

Usage

```
louis.info(fit)
```

Arguments

`fit`

Details

Value

Side Effects

References

See Also

Examples

`mf.gindx` *Model Frame Genotype Index to Account for Missing Data in haplo.glm*

Description

For internal use within the haplo.stats library

Usage

```
mf.gindx(m)
```

Arguments

`m`

Details

Value

Side Effects

References

See Also

Examples

`na.geno.keep` *Find non-missing rows in the genotype matrix of the model.frame*

Description

An internal function for the haplo.stats package

Usage

```
na.geno.keep(m)
```

Arguments

`m`

Details

Value

Side Effects

References

See Also

Examples

`plot.haplo.score` *Plot Haplotype Frequencies versus Haplotype Score Statistics*

Description

Method function to plot a class of type haplo.score

Usage

```
plot.haplo.score(x, ...)
```

Arguments

x	The object returned from haplo.score (which has class haplo.score).
...	Dynamic parameter for the values of additional parameters for the plot method.

Details

This is a plot method function used to plot haplotype frequencies on the x-axis and haplotype-specific scores on the y-axis. Because haplo.score is a class, the generic plot function can be used, which in turn calls this plot.haplo.score function.

Value

Nothing is returned.

Side Effects

References

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. "Score tests for association of traits with haplotypes when linkage phase is ambiguous." Amer J Hum Genet. 70 (2002): 425-434.

See Also

haplo.score

Examples

```
  setupData(hla.demo)
  geno <- as.matrix(hla.demo[,c(17,18,21:24)])
  keep <- !apply(is.na(geno) | geno==0, 1, any)
  hla.demo <- hla.demo[keep,]
  geno <- geno[keep,]
  attach(hla.demo)
  label <- c("DQB","DRB","B")

# For quantitative, normally distributed trait:

  score.gaus <- haplo.score(resp, geno, locus.label=label,
                            trait.type = "gaussian")

  plot.haplo.score(score.gaus)
```

plot.haplo.score.slide

Plot a haplo.score.slide Object

Description

Method function to plot an object of class haplo.score.slide. The p-values from haplo.score.slide are for sub-haplotypes of a larger chromosomal region, and these are plotted to visualize the change in p-values as the sub-haplotype "slides" over a chromosome. Plot -log10(p-value) on the y-axis vs. the loci over which it was computed on the x-axis.

Usage

```
plot.haplo.score.slide(x, pval="global", dist.vec=1:x$n.loci,
                       cex=.8, srt=270, ...)
```

Arguments

`x`	The object returned from haplo.score.slide
`pval`	Character string for the choice of p-value to plot. Options are: "global" (the global score statistic p-value based on an asymptotic chi-square distribution), "global.sim" (the global score statistic simulated p-value), and "max.sim" (the simulated p-value for the maximum score statistic).
`dist.vec`	Numeric vector for position (i.e. in cM) of the loci along a chromosome. Distances on x-axis will correspond to these positions.
`cex`	Character expansion size.
`srt`	String rotation in degrees measured counterclockwise from horizontal. Applies to x-axis (locus) labels.
`...`	Dynamic parameter for the values of additional parameters for the plot method.

Details

The x-axis has tick marks for all loci. The y-axis is the -log10() of the selected p-value. For each haplo.score result, plot a horizontal line at the height of -log10(p-value) drawn across the loci over which it was calculated. Therefore a p-value of 0.001 for the first 3 loci will plot as a horizontal line plotted at y=3 covering the first three tick marks.

Value

Nothing is returned.

Side Effects

References

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. "Score tests for association of traits with haplotypes when linkage phase is ambiguous." Amer J Hum Genet. 70 (2002): 425-434.

See Also

haplo.score.slide

Examples

```
#This example is run completely in the haplo.score.slide

#   setupData(hla.demo)
#   attach(hla.demo)
#   geno.11 <- hla.demo[,-c(1:4)]
#   label.11 <- c("DPB","DPA","DMA","DMB","TAP1","TAP2","DQB","DQA","DRB","B","A")

#For an ordinal trait, slide by 3 loci, and simulate p-values:
#   y.ord <- as.numeric(resp.cat)
#   slide.ord.sim <-  haplo.score.slide(y.ord, geno.11, trait.type = "ordinal",
#                             n.slide=3, locus.label=label.11, simulate=TRUE,
#                             sim.control=score.sim.control(min.sim=500))

#   print(slide.ord.sim)
#   plot(slide.ord.sim)
#   plot(slide.ord.sim, pval="global.sim")
#   plot(slide.ord.sim, pval="max.sim")
```

---

print.haplo.em *Print contents of a haplo.em object*

Description

Print a data frame with haplotypes and their frequencies. Also print likelihood information.

Usage

```
print.haplo.em(x, nlines=NULL, ...)
```

Arguments

x	A haplo.em object
nlines	To shorten output, print the first 1:nlines rows of the large data frame.
...	optional arguments for print

Details

Value

Nothing is returned

Side Effects

References

See Also

haplo.em

Examples

---

print.haplo.glm *Print a contents of a haplo.glm object*

Description

Print model information and then haplotype information.

Usage

```
print.haplo.glm(x, print.all.haplo=FALSE, digits =
                 max(options()$digits - 4, 3), ...)
```

Arguments

`x`	A haplo.glm object
`print.all.haplo`	Logical. If TRUE, print all haplotypes considered in the model.
`digits`	Number of numeric digits to print.
`...`	Optional arguments for print method

Details

Value

Nothing is returned

Side Effects

References

See Also

haplo.glm

Examples

print.haplo.group *Print a haplo.group object*

Description

Method function to print a class of type haplo.group

Usage

```
print.haplo.group(x, digits=max(options()$digits-2, 5), nlines=NULL, ...)
```

Arguments

`x`	The object returned from haplo.group (which has old class haplo.group).
`digits`	Set the number of significant digits to print for haplotype probabilities.
`nlines`	For shorter output, print first 1:nlines rows of the large data frame
`...`	Optional arguments for the print method

Details

This is a print method function used to print information from the haplo.group class, with haplotype-specific information given in a table. Because haplo.score is a class, the generic print function can be used, which in turn calls this print.haplo.group function.

Value

Nothing is returned.

Side Effects

References

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. Expected haplotype frequencies for association of traits with haplotypes when linkage phase is ambiguous. Submitted to Amer J Hum Genet.

See Also

haplo.score, haplo.group, haplo.em

Examples

`print.haplo.score` *Print a haplo.score object*

Description

Method function to print a class of type haplo.score

Usage

```
print.haplo.score(x, digits, nlines=NULL, ...)
```

Arguments

`x`	The object returned from haplo.score (which has class haplo.score).
`digits`	Number of digits to round the numeric output.
`nlines`	Print the first 'nlines' rows of the large data frame for fast, short view of the results.
`...`	Dynamic parameter for the values of additional parameters for the print method.

Details

This is a print method function used to print information from haplo.score class, with haplotype-specific information given in a table. Because haplo.score is a class, the generic print function can be used, which in turn calls this print.haplo.score function.

Value

Nothing is returned.

Side Effects

See Also

haplo.score

Examples

`print.haplo.score.merge` *Print a haplo.score.merge object*

Description

Method function to print a class of type haplo.score.merge

Usage

```
print.haplo.score.merge(x, order.by="score", all.haps=FALSE,
                digits=max(options()$digits-2, 5), nlines=NULL, ...)
```

Arguments

`x`	The object returned from haplo.score.merge (which has old class {S} haplo.score.merge).
`order.by`	Column of the haplo.score.merge object by which to order the results.
`all.haps`	Logical, if (T)rue prints a row for all haplotypes. If (F)alse, the default, only prints the haplotypes kept in haplo.score for modelling.
`digits`	Set the number of significant digits to print for the numeric output.
`nlines`	Print the first 'nlines' rows of the large data frame for a short view of the results.
`...`	Dynamic parameter for the values of additional parameters for the print method.

Details

This is a print method function used to print information from the haplo.score.merge class. Because haplo.score.merge is a class, the generic print function can be used, which in turn calls this print.haplo.score.merge function.

Value

Nothing is returned.

Side Effects

References

Schaid DJ, Rowland CM, Tines DE, Jacobson RM, Poland GA. Expected haplotype frequencies for association of traits with haplotypes when linkage phase is ambiguous. Submitted to Amer J Hum Genet.

See Also

haplo.score.merge, haplo.score, haplo.group

Examples

```
#see example for haplo.score.merge
```

`print.haplo.score.slide` *Print the contents of a haplo.score.slide object*

Description

Print the data frame returned from haplo.score.slide

Usage

```
print.haplo.score.slide(x, digits=max(options()$digits - 2, 5), ...)
```

Arguments

`x`	A haplo.score.slide object
`digits`	Number of digits to print for numeric output
`...`	Optional arguments for the print method

Details

Value

Side Effects

References

See Also

Examples

`printBanner` *Print a nice banner*

Description

Usage

```
printBanner(str, banner.width=80, char.perline=60, border="=")
```

Arguments

`str`	character string - a title within the banner
`banner.width`	width of banner
`char.perline`	number of characters per line for the title
`border`	type of character for the border

Details

Value

Side Effects

References

See Also

Examples

```
printBanner("This is a pretty banner", banner.width=40, char.perline=30)

#========================================
#         This is a pretty banner
#========================================
```

residScaledGlmFit *Scaled Residuals for GLM fit*

Description

For internal use within the haplo.stats library

Usage

```
residScaledGlmFit(fit)
```

Arguments

`fit`

Details

Value

Side Effects

References

See Also

Examples

score.sim.control *Create the list of control parameters for simulations in haplo.score*

Description

In the call to haplo.score, the sim.control parameter is a list of parameters that control the simulations. This list is created by this function, score.sim.control, making it easy to change the default values.

Usage

```
score.sim.control(p.threshold=0.25, min.sim=1000, max.sim=20000.,verbose=FALSE)
```

Arguments

`p.threshold`	A paremeter used to determine p-value precision from Besag and Clifford (1991). For a p-value calculated after min.sim simulations, continue doing simulations until the p-value's sample standard error is less than p.threshold * p-value. The dafault value for p.threshold = 1/4 corresponds approximately to having a two-sided 95% confidence interval for the p-value with a width as wide as the p-value itself. Therefore, simulations are more precise for smaller p-values. Additionally, since simulations are stopped as soon as this criteria is met, p-values may be biased high.
`min.sim`	The minimum number of simulations to run.
`max.sim`	The upper limit of simulations allowed. When the number of simulations reaches max.sim, p-values are approximated based on simulation results at that time.
`verbose`	Logical, if (T)rue, print updates from every simulation to the screen. If (F)alse, do not print these details.

Details

In simulations for haplo.score, employ the simulation p-value precision criteria of Besag and Clifford (1991). The criteria ensures both the global and the maximum score statistic simulated p-values be precise for small p-values. First, perform min.sim simulations to guarantee sufficient precision for the score statistics on individual haplotypes. Then continue simulations as needed until simulated p-values for both the global and max score statistics meet precision requirements set by p.threshold.

Value

A list of the control parameters:

`p.threshold`	As described above
`min.sim`	As described above. If run-time is an issue, a lower minimum (e.g. 500) may be useful.
`max.sim`	As described above
`verbose`	As described above

Side Effects

References

Besag, J and Clifford, P. "Sequential Monte Carlo p-values." Biometrika. 78, no. 2 (1991): 301-304.

See Also

`haplo.score`

Examples

```
# it would be used in haplo.score as appears below
#
# score.sim.500 <- haplo.score(y, geno, trait.type="gaussian", simulate=T,
#                sim.control=score.sim.control(min.sim=500, max.sim=2000)
```

---

setupData — *Set up an example dataset provided within the library.*

Description

This function defines an alias function to run exactly as data() in R and does nothing in Splus. R keeps a data set within the working data frame, so we only want to load data it when calling an example. Splus keeps it in background, so it is already loaded upon library(mypkg).

Usage

```
setupData(...)
```

Arguments

...	The name of a dataset provided within the Splus/R library.

Details

Value

Side Effects

References

See Also

Examples

`setupGeno` *Create a group of locus objects from a genotype matrix, assign to 'model.matrix' class.*

Description

The function makes each pair of columns a locus object, which recodes alleles to numeric and saves the original alleles as an attribute of the model.matrix.

Usage

```
setupGeno(geno, miss.val=c(0,NA))
```

Arguments

`geno`	Matrix of alleles, such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a chromosome. If there are K loci, then ncol(geno) = 2*K. Rows represent alleles for each subject.
`miss.val`	A vector of codes denoting missing values for allele1 and allele2. Note that NA will always be treated as a missing value, even if not specified in miss.val. Also note that if multiple missing value codes are specified, the original missing value code for a specific individual can not be retrieved from the loci object.

Details

Value

A 'model.matrix' object with the alleles recoded to numeric values, and the original values are stored in the 'unique.alleles' attribute. The ith item of the unique.alleles list is a vector of unique alleles for the ith locus.

Side Effects

References

Note

A matrix that contains all elements of mode character will be sorted in alphabetic order.

See Also

locus, loci, haplo.glm

Examples

```
# Create some loci to work with
a1 <- 1:6
a2 <- 7:12

b1 <- c("A","A","B","C","E","D")
b2 <-c("A","A","C","E","F","G")

c1 <- c("101","10","115","132","21","112")
c2 <- c("100","101","0","100","21","110")

myGeno <- data.frame(a1,a2,b1,b2,c1,c2)
myGeno <- setupGeno(myGeno)
myGeno

attributes(myGeno)$unique.alleles
```

`summary.haplo.em` *Summarize contents of a haplo.em object*

Description

Display haplotype pairs and their posterior probabilities by subject. Also display a table with number of max haplotype pairs for a subject versus how many were kept (max vs. used).

Usage

```
summary.haplo.em(object, show.haplo=FALSE, nlines=NULL, ...)
```

Arguments

`object`	A haplo.em object
`show.haplo`	Logical. If TRUE, show the alleles of the haplotype pairs, otherwise show only the recoded values.
`nlines`	To shorten output, print the first 1:nlines rows of the large data frame.
`...`	Optional arguments for the summary method

Details

Value

Side Effects

References

See Also

`haplo.em`

Examples

`summaryGeno` *Summarize Full Haplotype Enumeration on Genotype Matrix*

Description

Provide a summary of missing allele information for each individual in the genotype matrix. The number of loci missing zero, one, or two alleles is computed, as well as the total number of haplotype pairs that could result from the observed phenotype.

Usage

```
summaryGeno(geno, miss.val=0)
```

Arguments

`geno`	Matrix of alleles, such that each locus has a pair of adjacent columns of alleles, and the order of columns corresponds to the order of loci on a chromosome. If there are K loci, then geno has 2*K columns. Rows represent all observed alleles for each subject.
`miss.val`	Vector of codes for allele missing values.

Details

After getting information on the individual loci, this function makes a call to geno.count.pairs(). The E-M steps to estimate haplotype frequencies considers haplotypes that could result from a phenotype with a missing allele. It will not remove a subject's phenotype, only the unlikely haplotypes that result from it.

Value

Data frame with columns representing the number of loci with zero, one, and two missing alleles, then the total haplotype pairs resulting from full enumeration of the phenotype.

Side Effects

See Also

`geno.count.pairs`, `haplo.em`

Examples

`varfunc.glm.fit` *Variance Function for GLM*

Description

For internal use within the haplo.stats library

Usage

```
varfunc.glm.fit(fit)
```

Arguments

`fit`

Details

Value

Side Effects

References

See Also

Examples

Index