Package 'ibdreg'

February 20, 2015

Version 0.2.5

Date 2013-4-16

Title Regression Methods for IBD Linkage With Covariates

Author Jason P. Sinnwell and Daniel J. Schaid

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

Description A method to test genetic linkage with covariates by regression methods with response IBD sharing for relative pairs. Account for correlations of IBD statistics and covariates for relative pairs within the same pedigree.

License GPL-2 | file LICENSE

Depends R (>= 2.15.0)

URL http://mayoresearch.mayo.edu/mayo/research/schaid_lab/software.cfm

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-04-20 07:47:38

R topics documented:

`align.ibd.var` *align ibd.var object with relative pairs within pairs.frame object*

Description

align ibd.var object with relative pairs within pairs.frame object

Usage

```
align.ibd.var(id.df, ibd.var, epsilon=1e-5)
```

Arguments

`id.df`	a data.frame containing the three id columns (ped.id, person1.id, person2.id) from a pairs.frame object
`ibd.var`	an ibd.var object
`epsilon`	cutoff for singular values in a generalized inverse (Ginv)

Value

a list resembling an ibd.var object

chibar3.w *Compute mixture proportions, w, for mixture chi-square distribution*

Description

Compute mixture proportions, w, for mixture chi-square distribution when k=0..3, where k = df of chi-squares

Usage

```
chibar3.w(sigma)
```

Arguments

sigma	covariance matrix (3 x 3) of the 3 distributions

Value

mixing proportions, with element w[k+1] the mixture proportion for k, k=0..3.

See Also

pchibar

Examples

```
chibar3.w(sigma=diag(c(1,1,1)))
```

create.ibd.dat *Create an ibd.dat object*

Description

Combine a posterior IBD probabilities file with a prior IBD probabilities file

Usage

```
create.ibd.dat(postfile, priorfile, software="merlin", x.linked=FALSE,
               cov.data=NULL, rm.noninform=TRUE)
```

Arguments

postfile	full path and name of the file with posterior IBD probabilities
priorfile	full path and name of the file with prior IBD probabilities
software	character string of which software was used to create IBD probability files
x.linked	logical, is the chromosome X-linked. If TRUE, cov.data is required
cov.data	name of data.frame containing covariates, specifically ped.id, person.id, and sex. Required when x.linked=TRUE.
rm.noninform	logical, if TRUE, remove relative pairs that are not informative for linkage

Details

Perl scripts are provided within ibdreg/perl/ for creating IBD probability files, and are explained in the user manual. Prior probability files are made using a homozygous marker on the pedigree structure, or a "dummy" marker. When the chromosome is X-linked, the Merlin software treats males as homozygous for their X chromosome, and thus the probability of sharing 1 and 2 alleles IBD with any other relative are switched within create.ibd.dat.

Value

An object with class ibd.dat, which contains the following elements:

ped.id	pedigree identifier code
person1.id	identifier to person 1 of the relative pair
person2.id	identifier to person 2 of the relative pair
post0	data.frame with probability of sharing zero (0) alleles ibd between relative pairs (rows) at each position (columns)
post1	data.frame with probability of sharing one (1) alleles ibd between relative pairs (rows) at each position (columns)
post2	data.frame with probability of sharing two (2) alleles ibd between relative pairs (rows) at each position (columns)
prior0	vector with probability of sharing zero (0) alleles ibd between relative pairs, given no genotype data
prior1	vector with probability of sharing one (1) alleles ibd between relative pairs, given no genotype data
prior2	vector with probability of sharing two (2) alleles ibd between relative pairs, given no genotype data

See Also

ibdreg, ibd.df.merlin, mergeIBD

Examples

```
## do not run example in testing
## uncomment to run for demo

## ibd file for 1 chromosome
# ibdfile.ch20 <- "post.ibd"

## ibd file for 1 locus for prior probs.
# prior.ibdfile <- "prior.ibd"

# ibd.dat.obj <- create.ibd.dat(postfile=post.ibd,
#                               priorfile=prior.ibd,
#                               rm.noninform=FALSE)

# names(ibd.dat.obj)
```

`create.pairs.frame` *create a data frame for relative pairs*

Description

create a data frame for relative pairs containing identifiers, response, and covariates

Usage

```
create.pairs.frame(ibd.dat, model.mat, formula, c.scale)
create.pairs.frame.cvec(c.scale, prior2, prior1)
```

Arguments

`ibd.dat`	object of class ibd.dat
`model.mat`	a model matrix containing covariates for each person; to be used in making covariates for the relative pair
`formula`	format is either ~1 or ~pairs.fun(cov1). If the latter, the columns of the returned x.mat will be the result of the function operated on the covariate for each relative in the relative pair.
`c.scale`	scaling factor for covariates. Either "nodom" for no dominance variance or "minimax".
`prior2`	Prior probability of relative pairs to share 2 alleles ibd (from ibd.dat object)
`prior1`	Prior probability of relative pairs to share 1 allele ibd (from ibd.dat object)

Value

a list of 3 data frames all with the same number of rows. These data frames are id.df (3-columns of ped.id, person1.id, person2.id), y.mat (estimated ibd allele sharing, which is the regression response), status.df (3-columns for AA, UU, or AU status for the pair), x.mat (model matrix with intercept and, if covariates in formula, columns for covariates).

See Also

create.ibd.dat

---

exact.ibd.var *create an ibd.var object*

Description

create an ibd.var object from a temporary output file produced by exact.ibd.var.pl perl script, which uses exact computations from merlin

Usage

```
exact.ibd.var(file)
```

Arguments

file	a temporary output file created by exact.ibd.var.pl, a perl scripts provided in /ibdreg/perl/ in the ibdreg package

Value

an ibd.var object (e.g. ret) that contains the following elements for each pedigree: <pre> ret\$ped.id: pedigree id

ret\$person1.id: vector of ids for first person in the relative pair

ret\$person2.id: vector of ids for second person in the relative pair

ret\$sm: mean vector of ibd sharing between relative pairs {person1.id, person2.id}

ret\$sv: variance-covariance matrix for ibd sharing between pairs of relative pairs. </pre

See Also

sim.ibd.var

Examples

```
## create a temporary file using perl script

# unix% exact.ibd.var.pl chrom1.pre 1 chr1.var.tmp

## make an ibd.var object from chr1.var.tmp file

# RorS> chr1.ibd.var <- exact.ibd.var("chr1.var.tmp")
```

`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, eps=1e-6)
```

Arguments

`x`	A matrix.
`eps`	minimum cutoff for singular values in svd of x

Details

The svd function uses the LAPACK standard library 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. For S-PLUS, the Matrix library is required (Ginv loads Matrix if not already done so).

Value

List with components:

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

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.

Anderson, E., et al. (1994). LAPACK User's Guide, 2nd edition, SIAM, Philadelphia.

See Also

svd

Examples

```
# for matrix x, extract the generalized inverse and
# rank of x as follows
    x <- matrix(c(1,2,1,2,3,2),ncol=3)
    save <- Ginv(x)
    ginv.x <- save$Ginv
    rank.x <- save$rank
```

ibd.df.merlin *Create and return a data.frame with ibd data from merlin*

Description

Create and return a dataframe of ped.id, person1.id, person2.id, and ibd information from merlin

Usage

```
ibd.df.merlin(ibd.dat)
```

Arguments

ibd.dat	data.frame with posterior IBD data from Merlin, with column names: FAMILY, ID1, ID2, MARKER

Value

The returned data.frame, call it df, with the following elements:

df\$ped.id: pedigree id

df\$person1.id: df\$person2.id: identifiers for the relative pair within a pedigree

df\$post0: matrix objects that can be referenced from df. Each df\$post1: is a matrix with posterior ibd sharing probabilities for df\$post2: relative pairs (rows) at each chromosome position (columns)

ibd.moments *calculate moments (mean, variance) for simulated ibd vectors*

Description

calculate moments (mean, variance) for simulated ibd vectors for an individual predigree from sim.mark.prop

Usage

```
ibd.moments(simped, person.indx, male, x.linked=FALSE)
```

Arguments

simped	The object returned from sim.mark.prop
person.indx	person identifier
male	binary code (1/0) for if person is male
x.linked	logical, if TRUE, apply the method for an x-linked chromosome

Value

a list with person1.id person2.id sm and sv

See Also

sim.mark.prop

---

ibdreg *Regression Methods to Test for Linkage With Covariates*

Description

A method to test genetic linkage with covariates by regression models that use the IBD status for relative pairs as the dependent variable and pair-specific covariates as the independent variables. Correlations of IBD statistics and covariates for relative pairs within the same pedigree are accounted for.

Usage

```
ibdreg(formula, status.method, c.scale='nodom', data,
                  status, ped.id, person.id,
                  ibd.dat, ibd.var,
                  subset, weights, na.action,
                  min.pairs=1, epsilon=1e-5, ...)
```

Arguments

formula	either "~1" for no covariates to get only linkage tests, or a function of the paired covariates (e.g., "~pairs.sum(cov1, cov1)") for tests that include covariates.
status.method	Character string indicating which relative pairs to apply the ibdreg method: "AA", "UU", "AU", or "ALL". These correspond to: Affected-Affected, Unaffected-Unaffected, Affected-Unaffected, and all pairs, respectively.
ibd.dat	an ibd.dat object, created by create.ibd.dat() with elements: ped.id, person1.id, person2.id, which collectively identify relative pairs, post0, post1, post2, which are the posterior probabilities of sharing 0, 1, and 2 alleles IBD at each chromosome position, and prior0, prior1, prior2, which are the null prior probabilities of sharing 0, 1, and 2 alleles ibd.
ibd.var	a list containing an element for each pedigree. The elements include ped.id, person1.id, person2.id, and the variance-covariance matrix of ibd statistics between pairs of subject pairs.
status	column name of data that has affection status (1=unaffected; 2=affected; NA=missing)
ped.id	column name of data that has pedigree id
person.id	column name of data that has a person's id, which only have to be unique within pedigrees

data	a data frame containing all variables in the model formula, in addition to status, ped.id, and person.id
c.scale	choice for scaling factor on covariates, "minimax" (same as LODPAL) and "nodom" (no dominance-variance)
subset	an optional vector specifying a subset of observations in data
weights	an optional vector of weights to be used in the regression method
na.action	a function that defines how missing values (NA) are handled. The default is set by the 'na.action' setting of 'options', and is 'na.fail' by default.
min.pairs	minimum number of relative pairs needed for linkage tests to be calculated on a specific status group (e.g. AA pairs)
epsilon	minimum value for singular values in generalized inverse calculations
...	further arguments passed to or from other methods

Details

The tests for genetic linkage use quasi-likelihood score statistics, formulated in terms of weighted least squares regression. The covariates in the regression framework are scaled according to the degree of relationship between relative pairs within the pedigree (c.scale). The method yields the following tests for linkage and/or covariate effect for relative pairs: (i) Linkage only; (ii) Linkage with covariate effects; (iii) Covariate effect on IBD sharing (e.g, heterogeneity), assuming either a model-based variance-covariance matrix or a robust variance-covariance matrix. Tests (i) - (iii) are evaluated as unconstrained (two-sided test) and constrained for excess allele sharing in one direction. If formula is \~1, there are no covariates, so the tests will be for linkage only.

Another test (iv) is for linkage using all relative pairs. This test imposes constraints on allele sharing for one-sides tests favoring linkage, achieved by constraining IBD allele sharing for pairs as AA > UU > null, and AU < null, where null is the expected IBD sharing without linkage.

Tests (i) - (iv) are performed using quasi-likelihood scores, and their distributions are asymptotically chi-square or mixture chi-square.

Another set of tests (v) are z-scores measuring the departure from expected allele sharing for each pedigree, at each position. These are provided for diagnostic purposes in the linkage.tests results for AA, UU, and AU pairs.

Correlations of relative pairs from the same pedigree are accounted for by a covariance matrix for ibd statistics within pedigrees. This variance-covariance matrix is calculated using results from merlin (with options "–ibd –matrices").

Value

A list with the following components:

Call	function call to ibdreg
relpair.tbl	table with number of relative pairs in each pedigree
status.tbl	table of counts for AA, UU, and AU relative pairs
AA.linkage	an object of class linkage.tests; includes score statistics for linkage for AA relative pairs that includes tests (i) - (iii), (v) (see details)

UU.linkage	an object of class linkage.tests; includes score statistics for linkage for UU relative pairs that includes tests (i) - (iii), (v) (see details)
AU.linkage	an object of class linkage.tests; includes score statistics for linkage for AU relative pairs that includes tests (i) - (iii), (v) (see details)
ALL.linkage	object of class linkage.all; includes score statistics for linkage with ALL pairs (test iv).

References

Schaid DJ, Sinnwell JP, Thibodeau SN. Testing Genetic Linkage with Affected Relative Pairs and Covariates by Quasi-Likelihood Score Statistics. Submitted.

See Also

create.ibd.dat, sim.ibd.var, exact.ibd.var, linkage.tests, linkage.all, print.ibdreg, plot.ibdreg

Examples

```
## see manual for examples, the data requirements are here:

# make a data.frame with ped.id, person.id, status, and covariates
# make ibd.dat object using create.ibd.dat
# make ibd.var object using create.ibd.var

# call ibdreg
```

---

linkage.all *Linkage test statistics for ALL relative pairs*

Description

Calculate and store linkage test statistics for ALL relative pairs. The tests include

Usage

```
linkage.all(y.mat, x.adj, ibdvar.lst, epsilon=1e-5)
linkage.all.pedloop(y.vec, x.adj, ibdvar.lst)
```

Arguments

y.mat	matrix of ibd sharing statistics, the response in the regression model; each column represents one chromosome position.
y.vec	vector of ibd sharing statistics (at one position), the response in the regression model (pedloop only)
x.adj	matrix of covariates, x, which are indicators for which status group the relative pairs belong, adjusted by c.scale.

ibdvar.lst	a list containing an element for each pedigree. The elements include ped.id, person1.id, person2.id, and the variance-covariance matrix of pairs of pairs osf subjects.
epsilon	cutoff for singular values in generalized inverse (Ginv)

Details

linkage.all sets up the linkage calculations for each position, and the calculations are performed over all pedigrees in linkage.all.pedloop.

Value

a linkage.all object, containing tests for linkage on all relative pairs

See Also

ibdreg, print.linkage.all, plot.linkage.all

---

linkage.tests *Linkage tests for relative pair groups*

Description

Linkage tests, with and without covariates, for relative pair status groups Affected-Affected (AA), Unaffected-Unaffected (UU), or Affected-Unaffected (AU)

Usage

```
linkage.tests(y.mat, x.adj, ibdvar.lst, status.method, epsilon=1e-5)
linkage.tests.B0(y.mat, xvec, ibdvar.lst)
linkage.tests.pedloop(y.vec, x.adj, ibdvar.lst,
                      B0, concordant=TRUE, epsilon)
```

Arguments

y.mat	matrix of ibd sharing statistics, which are the response in the regression model; each column represents one chromosome position.
y.vec	vector of ibd sharing statistics (at one position), which are the response in the regression model (linkage.tests.pedloop only)
x.adj	matrix of covariates, x, adjusted by c.scale.
xvec	the intercept column from x.adj, needed for pre-calculating Beta-0
ibdvar.lst	a list containing an element for each pedigree. The elements include ped.id, person1.id, person2.id, and the variance-covariance matrix of pairs of pairs of subjects.
status.method	Character string indicating which relative pairs to apply the linkage tests: "AA", "UU", or "AU".

B0	The intercept in the regression model, ("beta-zero").
concordant	logical; indicate whether to perform tests for linkage on concordant pairs (AA, UU), or discordant (AU).
epsilon	cutoff for singular values in generalized inverse (Ginv)

Details

The calculations for linkage tests require a pre-calculation of beta-zero (linkage.tests.B0) and then proceed with calculating the linkage test statistics for each position by looping over all pedigrees (linkage.tests.pedloop). Linkage tests can only be applied to one status group at a time, because the tests are restricted to whether the relative pairs are all concordant or discordant. Unexpected allele sharing can be detected when unconstrained linkage > constrained linkage tests. The method also returns z scores for pedigree-specific contributions to unexpected allele sharing indicated in linkage tests.

Value

an object of class linkage.tests

Side Effects

none

See Also

ibdreg, print.linkage.tests, plot.linkage.tests

---

lsConstrain.fit *Minimize Inequality Constrained Mahalanobis Distance*

Description

Find the vector z that solves:

min{ (x - z)'inv(S)(x - z); Az <= b },

where x is an input vector, S its covariance matrix, A is a matrix of known contrasts, and b is a vector of known constraint constants.

Usage

```
lsConstrain.fit(x, b, s, a, iflag, itmax=4000, eps=1e-06, eps2=1e-06)
```

Arguments

x	vector of length n
b	vector of length k, containing constraint constants
s	matrix of dim n x n, the covariance matrix for vector x
a	matrix of dim k x n, for the contraints
iflag	vector of length k; an item = 0 if inequality constraint, 1 if equality constraint
itmax	scalar for number of max interations
eps	scalar of accuracy for convergence
eps2	scalar to determine close to zero

Value

List with the following components:

itmax: (defined above)

eps: (defined above)

eps2: (defined above)

iflag: (defined above)

xkt: vector of length k, for the Kuhn-Tucker coefficients.

iter: number of completed iterations.

supdif: greatest difference between estimates across a full cycle

ifault: error indicator: 0 = no error 1 = itmax exceeded 3 = invalid constraint function for some row ASA'=0.

a: (defined above)

call: function call

x.init: input vector x.

x.final: the vector "z" that solves the equation (see z in description).

s: (defind above)

min.dist: the minimum value of the function (see description).

References

Wollan PC, Dykstra RL. Minimizing inequality constrained mahalanobis distances. Applied Statistics Algorithm AS 225 (1987).

Examples

```
# An simulation example with linear regression with 3 beta's,
# where we have the contraints:
#
# b[1] > 0
# b[2] - b[1] < 0
# b[3] < 0
```

```
set.seed(111)

n <- 100
x <- rep(1:3,rep(n,3))
x <- 1*outer(x,1:3,"==")

beta <- c(2,1,1)

y <- x%*%beta + rnorm(nrow(x))

fit <- lm(y ~-1 + x)

s <- solve( t(x) %*% x )

bhat <- fit$coef


a <-  rbind(c(-1, 0, 0),
            c(-1, 1, 0),
            c( 0, 0, 1))

# View expected constraints (3rd not met):

a %*% bhat
#            [,1]
# [1,] -1.8506811
# [2,] -0.9543320
# [3,]  0.8590827

b <- rep(0, nrow(a))
iflag <- rep(0,length(b))

save <- lsConstrain.fit(x=bhat,b=b, s=s, a=a, iflag=iflag, itmax=500,
                        eps=1e-6, eps2=1e-6)

save
```

`mergeIBD` *Merge ibd and covariate data*

Description

Merge ibd probabilities data frame and an individual's sex on pedigree id and person id

Usage

```
mergeIBD(ibd.dat, sex.dat)
```

Arguments

ibd.dat	A data frame with ped.id, person1.id, person2.id, and the estimated ibd sharing values at each chromosome position for the two people.
sex.dat	a data frame containing ped.id, person.id, and sex

Value

a data frame with identifiers for relative pairs, their ibd data, and the two sex codes for the pair

---

minpRows *find the rows of a data frame with the minimum pvalue*

Description

find the rows of a data frame with the minimum pvalue, best used in a print or summary function to summarize a group of test statistics.

Usage

```
minpRows(obj, colnames=NULL, rowname=NULL, col.indx=ncol(obj))
```

Arguments

obj	a data frame for a group of statistical tests, with pvalues
colnames	new names to give to columns
rowname	new names to give the row(s). If multiple rows contain the minimum pvalue, rowname is assigned to each of them, appending a digit at the end (1, 2, etc).
col.indx	the index of the columns that contains the pvalue, default is the last column of obj

Value

a subset of the obj data.frame rows

Examples

```
## create a data frame of chi-square(1) tests
tests <- data.frame(pos=1:3, chitest=c(3,4,2), df=rep(1,3),
                    pval=1-pchisq(c(3,4,2), df=1))

# find the rows of the best tests
best.tests <- minpRows(tests, colnames=c("position", "chi.test", "df","pvalue"))
```

pairSum *Combine person-specific covariates to pair-specific*

Description

Functions to combine person-specific covariates to pair-specific

Usage

```
pairSum(cov1, cov2)
pairDiff(cov1, cov2)
```

Arguments

cov1

cov2

Details

For use within the ibdreg package to make covariates that are a function of the two covariates in a relative pair. These functions are to be used in the formula parameter of ibdreg, and the pair-specific covariates can be expressed as pairSum(cov) or pairSum(cov, cov).

Value

Either the sum or the difference (absolute) of the two covariates

See Also

ibdreg

Examples

```
pairSum(20,30)

pairDiff(20,30)
```

pchibar *cumulative probability of a chibar distribution*

Description

cumulative probability of a chi-bar distribution (mixed chi-square)

Usage

```
pchibar(x, df, wt)
```

Arguments

x	quantile of chi-bar distribution
df	vector degrees of freedom
wt	vector of mixing proportions (weights)

Value

cumulative density of chi-bar distribution

See Also

pchisq

Examples

```
pch = pchibar(3, df=c(0,1), wt=c(.5,.5))
1-pch  # compare to 1-pchisq(3,1)
```

plot.ibdreg *plot an ibdreg object*

Description

plot test results within an ibdreg object

Usage

```
 ## S3 method for class 'ibdreg'
plot(x, ...)
```

Arguments

x	an ibdreg object
...	optional parameters for plot function, title is expected

Details

If status.method was AA, UU, or AU, only tests for linkage on the chosen group of relative pairs will be plotted. If status.method was ALL, then the linkage test in linkage.all object is plotted. If status.method was ALL and only linkage tests were done, 4 lines for linkage of AA, UU, AU, ALL linkage are plotted together.

Value

nothing is returned

See Also

ibdreg, plot.ibdreg.unexpect, plot.linkage.tests, plot.linkage.all

---

plot.ibdreg.unexpect *Plot unconstrained and constrained linkage tests to indicate areas of unexpected allele sharing*

Description

Plot the -log10(pvalue) of the unconstrained and constrained linkage tests to indicate areas of unexpected allele sharing

Usage

```
 ## S3 method for class 'ibdreg.unexpect'
plot(x, status.method="AA", ...)
```

Arguments

x	an ibdreg object
status.method	Character string indicating which affection status for relative pairs ("AA", "AU", or "UU") is to be examined for unexpected IBD sharing. Character string indicating which relative pairs to display linkage tests for: "AA", "AU", or "UU". These correspond to: Affected-Affected, Affected-Unaffected, and Unaffected-Unaffected, respectively.
...	extra parameters for the plot method. This method looks for a 'title' parameter.

Value

nothing is returned

See Also

plot.ibdreg, ibdreg

plot.linkage.all *Plot a linkage.all object*

Description

Plot a linkage.all object

Usage

```
## S3 method for class 'linkage.all'
plot(x, ...)
```

Arguments

x	A linkage.all object
...	additional plot parameters, this method checks for 'title'

Value

nothing is returned

See Also

ibdreg, linkage.all, print.linkage.all

plot.linkage.tests *Plot a linkage.tests object*

Description

Plot a linkage tests object, which was made within ibdreg. Plot the -log10(pvalue) for pvalues from the score test statistics for linkage with and without covariates, constrained for one-sided tests.

Usage

```
## S3 method for class 'linkage.tests'
plot(x, ...)
```

Arguments

x	An object of class linkage.tests
...	additional plot parameters, this method checks for 'title'

Value

nothing is returned

See Also

ibdreg, linkage.tests

Examples

```
# reg.out <- ibdreg(formula=~1, ped.id=ped.id, person.id=person.id,
#                   status=status, status.method="AA", data=data,
#                   ibd.dat=ibd.dat.obj, ibd.var=ibd.var.obj)

# plot.linkage.tests(reg.out$AA.linkage, title="AA, formula: ~1")
```

---

plotpval *Plot the -log10 of pvalues*

Description

Plot lines connecting the the -log10 of pvalues for a series of tests

Usage

```
plotpval(pos, pmat, lty=1, lwd=1, col=1, title="", legend,
         xlab="Position(cM)", ylab="-log10(pvalue)")
```

Arguments

pos	an index for the series of tests, usually assuming adjacent tests are correlated.
pmat	matrix of p-values, each column represents a different test, rows correspond to the pos[ition] index
lty	line type, see par()
lwd	line width, see par()
col	line color, see par()
title	title for the top-center of the plot
legend	vector of text strings explaning the tests in each column of pmat
xlab	x axis label
ylab	y axis label

Details

The default height for the y-axis is 3, which corresponds to a p-value of 1e-3. If any p-value is smaller, the height is -log10 of the smallest p-value in pmat.

Value

nothing is returned

Side Effects

none

See Also

par, title, legend

Examples

```
plotpval(pos=1:10, pmat=cbind(runif(10)/5, runif(10)/10),
         lty=c(1,2), col=c(1,2), legend=c("runif/5", "runif/10"))
```

`print.ibd.var` *print an ibd.var object*

Description

Print results stored in an ibd.var object. If ped.id is NULL and sinkfile is NULL, print everything to a made-up sink file because the output is most likely too big for standard output. If ped.id is not NULL, print the ibd.var elements that match ped.ids. If sinkfile is not NULL, sink results to that file.

Usage

```
 ## S3 method for class 'ibd.var'
print(x, ped.id=NULL, sinkfile=NULL, digits=max(options()$digits - 2, 5), ...)
```

Arguments

x	An ibd.var object, created by either exact.ibd.var or sim.ibd.var
ped.id	Vector of pedigree IDs to specify a subset of the pedigrees to print
sinkfile	Name of file to sink too. Printed output from ibd.var is large, especially if ped.id is not given, so enforce a file name for where to sink print results. If ped.id is NULL and sinkfile is not given, default is set to name of x with ".sink".
digits	number of significant digits to print for numeric values
...	optional parameters for the print method

Value

nothing is returned

See Also

sim.ibd.var, exact.ibd.var, ibdreg

print.ibdreg *print and ibdreg object*

Description

Print results stored in an ibdreg object, showing the function call, the counts of relative pairs used, and the tests performed for linkage with and without covariates on the relative pairs. A table is given to show the test at the chromosome position which had the minimum p-value.

Usage

```
## S3 method for class 'ibdreg'
print(x, digits=max(options()$digits - 2, 5), ...)
```

Arguments

x	An ibdreg object
digits	number of significant digits to print for numeric values
...	optional parameters for the print method

Value

nothing is returned

See Also

ibdreg

print.linkage.all *Print or summarize a linkage.all object*

Description

Print a linkage.all object, made within ibdreg. For the linkage test, print the test statistic, degrees of freedom, and pvalue at the position with the smallest pvalue.

Usage

```
## S3 method for class 'linkage.all'
print(x, digits=max(options()$digits - 2, 5), ...)
```

Arguments

x	A linkage.all object
digits	number of significant digits to print for numeric values
...	additional display parameters

Value

nothing is returned

See Also

ibdreg, linkage.all, plot.linkage.all

`print.linkage.tests` *Print a linkage.tests object*

Description

Print a linkage.tests object, made within ibdreg. For each test, print the score test statistic, degree(s) of freedom, and pvalue at the position with the smallest pvalue.

Usage

```
## S3 method for class 'linkage.tests'
print(x, digits=max(options()$digits - 2, 5),
                   show.model.tests=FALSE, ...)
```

Arguments

x	A linkage.tests object
digits	The number of significant digits to print for numeric values
show.model.tests	logical, if TRUE, show tests with model-based covariance, in addition to the other tests
...	Additional print parameters

Value

nothing is returned

See Also

ibdreg, linkage.tests, plot.linkage.tests

Examples

```
# reg.out <- ibdreg(formula=~1, ped.id=ped.id, person.id=person.id,
#                   status=status, status.method="AA", data=data,
#                   ibd.dat=ibd.dat.obj, ibd.var=ibd.var.obj)

# print.linkage.tests(reg.out$AA.linkage, digits=4)
```

printBanner *Print a nice banner*

Description

Print a nice banner with a border above and below the text. It centers the text, and adjusts to the width system option by breaking into multiple lines when needed.

Usage

```
printBanner(str, banner.width=options()$width, char.perline=.75*banner.width, border="=")
```

Arguments

str	character string - a title within the banner
banner.width	width of banner, the default is set to fit current options
char.perline	number of characters per line for the title, the default is 75% of the banner.width parameter
border	type of character for the border

Details

This function prints a nice banner in both R and S-PLUS

Value

nothing is returned

See Also

options

Examples

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

# the output looks like this:
# ========================================
#         This is a pretty banner
# ========================================
```

`resources` *track system resources*

Description

track system resources during evaluation of an expression

Usage

```
resources(expr, doPrint=TRUE)
```

Arguments

`expr`	An expression to be evaluated by eval() within the sys.parent() evaluation frame.
`doPrint`	Logical, print the resource information in the call?

Details

The resources given by Venebles and Ripley had reported both Cache and Working memory usage. S-PLUS 7 no longer tracks Cache usage, so we only report the Working. Similarly, R has Ncells and Vcells, and the more important measure for users is Vcells, the memory heap.

Value

Return the data frame that is printed by default if the object is saved to a value (via invisible()). The expression is evaluated and details are printed for CPU time, overall time, child process time, and heap space memory usage.

References

Venebles WN, Ripley BD. "Statistics and Computing." Springer-Verlag, New York, NY, 2000. (page 151.)

See Also

`proc.time`, R: gc, S-PLUS: mem.tally.reset, mem.tally.report

Examples

```
resources({
  norm.dat = rnorm(10000)
  norm.mat = matrix(rnorm(10000), nrow=100)
})
```

sim.ibd.setup *set-up functions for sim.mark.prop*

Description

set-up functions for sim.mark.prop

Usage

```
sim.ibd.setup(ped, miss.val = c(NA, 0), x.linked=FALSE)
setFounderAlleles(ped, x.linked=FALSE)
```

Arguments

ped	a data.frame with data to characterize one pedigree. The columns contain id codes for person, mother, and father; also a column for sex.
miss.val	code(s) for missing values ped
x.linked	logical; if TRUE, apply method to x chromosome

Value

A ped object to be used in sim.mark.prop

See Also

sim.mark.prop

sim.ibd.var *Create an ibd.var object via simulations using gene-dropping*

Description

Create an ibd.var object, containing a mean vector and covariance matrix for the ibd sharing values between each pair of relative pairs in each pedigree. Use a gene-dropping approach with 2 unique alleles for each founder, and simulate n.sim times. Then calculate the moments based on the vector of allele sharing between all relative pairs within the pedigree.

Usage

```
sim.ibd.var(pedfile,
            male.code=1,
            female.code=2,
            x.linked=FALSE,
            n.sim=1000,
            print.resources=FALSE)
```

Arguments

pedfile	file containing ids for pedigree, person, father, and mother, and gender. Also known as a pre-MAKEPED format file.
male.code	The code that denotes a sex classification of "male".
female.code	The code that denotes a sex classification of "female".
x.linked	A flag that denotes whether the inheritance mode should be autosomal (x.linked=FALSE, the default) or X-linked (x.linked=TRUE).
n.sim	Number of simulations
print.resources	logical, set to TRUE to print CPU and memory usage for calculations on each pedigree.

Details

The bulk of the work for simulating ibd sharing vectors is in sim.mark.prop(). These methods will handle practically any correctly formed pedigree structure. This includes pedigrees containing loops, multiple pairs of ancestoral founders, and both loops and multiple pairs of ancestoral founders. Thus, maximum flexibility may be obtained. Both autosomal and X-linked methods of inheritance are supported through the use of the x.linked parameter.

Value

The return value, call it ret, is an ibd.var object that contains the following elements for each pedigree:

ret\$ped.id: pedigree id

ret\$person1.id: vector of ids for first person in the relative pair

ret\$person2.id: vector of ids for second person in the relative pair

ret\$sm: mean vector of ibd sharing between relative pairs {person1.id, person2.id}

ret\$sv: variance-covariance matrix for ibd sharing between pairs of relative pairs.

References

Schaid DJ, Sinnwell JP, Thibodeau SN. Testing Genetic Linkage with Relative Pairs and Covariates by Quasi-Likelihood Score Statistics. Submitted.

See Also

exact.ibd.var, sim.ibd.setup, sim.mark.prop

Examples

```
## See manual(s) for full example usage
## Below is a basic example for one pedigree

# Since input parameter is a "pre" file, create a "pre" file for one pedigree

# make a data.frame to write to a file
```

```
ped.id <- rep(7,7)
person <- c(1,2,3,4,5,6,7)
father <- c(0,0,0,1,0,3,5)
mother <- c(0,0,0,2,0,4,4)
sex    <- c(1,2,1,2,1,1,1)
chrom1 <- c(1,3,5,0,7,0,0)
chrom2 <- c(2,4,6,0,8,0,0)
ped7.df <- data.frame(ped.id, person, father, mother,
                      sex, chrom1, chrom2)

# write the file
write.table(ped7.df, file="ped7.pre", row.names=FALSE, col.names=FALSE)

ped7.ibdVar <- sim.ibd.var("ped7.pre", n.sim=1000)

# results are long and difficult to interpret
# here is an example of how to view
# > print(ped7.ibdVar[[1]], digits=2)

# demonstrate for x.linked=TRUE
ped7.ibdVarX <- sim.ibd.var("ped7.pre", n.sim=1000, x.linked=TRUE)
```

`sim.mark.prop` *Simulate Marker Propagation*

Description

Simulates inheritance-by-descent in pedigree data through both autosomal and X-linked modes of inheritance.

Usage

```
sim.mark.prop(ped,
              iseed=NULL,
              miss.val=c(NA, 0),
              male.code=1,
              female.code=2,
              x.linked=FALSE,
              proband=0,
              n.iter=1)
```

Arguments

`ped`	The ped argument is an object of class "list" or class "data.frame" and it must have the following six fields: ped\$person: A vector or factor. The person ID of each pedigree member. ped\$father: A vector or factor. The father ID of each pedigree member. ped\$mother: A vector or factor. The mother ID of each pedigree member.

ped\$sex: A vector or factor. The sex of each pedigree member.

ped\$chrom1: A vector or factor. The first marker of the marker data pair for each pedigree member (only founder data is required; non-founder data will be ignored in the simulations). If x.linked=TRUE, then male pedigree members must be homozygous at the specified marker site.

ped\$chrom2: A vector or factor. The second marker of the marker data pair for each pedigree member (only founder data is required; non-founder data will be ignored in the simulations). If x.linked=TRUE, then male pedigree members must be homozygous at the specified marker site.

`iseed` An integer or a saved copy of .Random.seed. This allows simulations to be reproduced by using the same initial seed. If no value is given, the current copy of .Random.seed will be used.

`miss.val` Codes that denote missing values in the pedigree data given by ped.

`male.code` The code that denotes a sex classification of "male".

`female.code` The code that denotes a sex classification of "female".

`x.linked` A flag that denotes whether the inheritance mode should be autosomal (x.linked=FALSE, the default) or X-linked (x.linked=TRUE).

`proband` An integer denoting which person should be the first person encountered by the traversal algorithm. This parameter is used for testing and need not be set by the user. It's value will not have any effect on the statistical properties of the return value.

`n.iter` An integer stipulating the number of simulated inheritance-by-descent calculations that should be computed (and returned) by the sim.mark.prop() program.

Details

The sim.mark.prop() program will handle practically any correctly formed pedigree structure. This includes pedigrees containing loops, multiple pairs of ancestoral founders, and both loops and multiple pairs of ancestoral founders. Thus, maximum flexibility may be obtained.

Both autosomal and X-linked methods of inheritance are supported through the use of the x.linked parameter. When X-linked inheritance is specified, the input marker data ("ped\$chrom1" and "ped\$chrom2") must be homozygous for male pedigree members, and the simulated output marker data ("ret\$mark1" and "ret\$mark2") will be homozygous for male pedigree members.

When more than one set of simulated marker data is desired, it is advisable to set the "n.iter" parameter to the number of simulated marker sets that are desired, rather than calling sim.mark.prop() in a loop n.iter times, as the underlying compiled C program that does the simulations forms the inputted pedigree one time, and then perfoms n.iter simulations upon that single pedigree. Calling sim.mark.prop() one time for each simulation results in proportionately large increases in redundant work (both in S-PLUS and C) and call overhead. The only thing to be aware of is that the amount of memory used by a call to sim.mark.prop() grows linearly with (n.subjects*n.iter). Therefore, if the product of the number of pedigree subjects with the number of simulations requested is unusually large, it may be advisable to split the simulations into more than one call to sim.mark.prop() in order to avoid exhausting the amount of primary memory afforded to S-PLUS (both to avoid the unintended termination of the host S-PLUS process and to avoid the significant increase in computational time that is seen when disk paging is required).

The non-founder members of the pedigree will have the first marker of each simulated marker pair (ret\$mark1) inherited from the father, and the second marker of each simulated marker pair (ret\$mark2) inherited from the mother. Thus, it is possible to follow maternal and paternal allele transmission through dependent pedigree members.

Value

The return value, call it ret, is an object of class "list" or class "data.frame" (the same class as the class of the "ped" parameter that the sim.mark.prop() function was called with, and it has the follwing fields:

ret\$person: A vector or factor. The person ID of each pedigree member.

ret\$father: A vector or factor. The father ID of each pedigree member.

ret\$mother: A vector or factor. The mother ID of each pedigree member.

ret\$sex: A vector or factor. The sex of each pedigree member.

ret\$chrom1: A vector or factor. The first marker of the marker data pair for each pedigree member input into the sim.mark.prop() program.

ret\$chrom2: A vector or factor. The second marker of the marker data pair for each pedigree member input into the sim.mark.prop() program.

ret\$mark1: A vector if n.iter=1, otherwise an object of class "model.matrix". Letting n.subjects denote the number of pedigree members, "mark1" is an n.subject-by-n.iter matrix with the ith column listing the first simulated marker of each simulated marker pair computed by the ith simulation for the corresponding pedigree member. If X-linked inheritance was specified (x.linked=TRUE), each male will be homozygous at the simulated marker location.

ret\$mark2: A vector if n.iter=1, otherwise an object of class "model.matrix". Letting n.subjects denote the number of pedigree members, "mark2" is an n.subject-by-n.iter matrix with the ith column listing the second simulated marker of each simulated marker pair computed by the ith simulation for the corresponding pedigree member. If X-linked inheritance was specified (x.linked=TRUE), each male will be homozygous at the simulated marker location

ret\$x.linked: A flag denoting whether the simulated "mark1" and "mark2" data was simulated to conform to autosomal inheritance or X-linked inheritance.

Side Effects

The sim.mark.prop() program has no side effects.

Examples

```
#################################################
### The following examples use a list as input: ###
#################################################

ped7 <- list(NULL)
ped7$person <- c(1,2,3,4,5,6,7)
ped7$father <- c(0,0,0,1,0,3,5)
ped7$mother <- c(0,0,0,2,0,4,4)
ped7$sex    <- c(1,2,1,2,1,1,1)
ped7$chrom1 <- c(1,3,5,0,7,0,0)
```

```
ped7$chrom2 <- c(2,4,6,0,8,0,0)

#ped7:

#       1-------2
#     1/2   |   3/4
#           |
#           |
#   3-------4-------5
# 5/6   |  0/0  |   7/8
#       |       |
#       |       |
#       6       7
#     0/0       0/0

prop7 <- sim.mark.prop(ped7, n.iter=5)

prop7


## an x.linked example

ped7.x <- list(NULL)
ped7.x$person <- c(1,2,3,4,5,6,7)
ped7.x$father <- c(0,0,0,1,0,3,5)
ped7.x$mother <- c(0,0,0,2,0,4,4)
ped7.x$sex    <- c(1,2,1,2,1,1,1)
ped7.x$chrom1 <- c(1,2,4,0,5,0,0)
ped7.x$chrom2 <- c(1,3,4,0,5,0,0)

# ped7.x:
#
#       1-------2
#     1/1   |   2/3
#           |
#           |
#   3-------4-------5
# 4/4   |  0/0  |   5/5
#       |       |
#       |       |
#       6       7
#     0/0       0/0

prop7X <- sim.mark.prop(ped7.x, x.linked=TRUE, n.iter=5)

prop7X
```

Index