Package 'CrypticIBDcheck'

July 2, 2014

Type Package

Title Identifying cryptic relatedness in genetic association studies

Version 0.3-1

Date 2012-04-14

Author Annick Joelle Nembot-Simo, Jinko Graham and Brad McNeney

Maintainer Brad McNeney <mcneney@sfu.ca>

Depends R (>= 2.14.0), rJPSGCS (>= 0.2-5), car (>= 2.0-9), ellipse (>= 0.3-5)

Imports chopsticks

Suggests parallel

Description Exploratory tools to identify closely related subjects using autosomal genetic marker data.

License GPL-3

LazyData yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-09-21 08:05:29

R topics documented:

`CrypticIBDcheck-package` *identify cryptic relatedness in genetic association studies*

Description

CrypticIBDcheck can be used to identify pairs of closely-related subjects based on genetic marker data from single-nucleotide polymorphisms (SNPs). The package is able to accommodate SNPs in linkage disequibrium (LD), without the need to thin the markers so that they are approximately independent in the population. Sample pairs are identified as related by superposing their estimated identity-by-descent (IBD) coefficients on plots of IBD coefficients for pairs of simulated subjects from one of several common close relationships. The methods are particularly relevant to candidate-gene association studies, in which dependent SNPs cluster in a relatively small number of genes spread throughout the genome.

Details

The main function in CrypticIBDcheck is `IBDcheck`, which uses SNP information to estimate IBD coefficients for pairs of study subjects and optionally for simulated pairs of subjects and returns an object of class `IBD`. The plot method for the `IBD` class, `plot.IBD`, displays the IBD coefficients for pairs of study subjects, along with prediction ellipses for known relationship pairs.

The package comes with two vignettes: `CrypticIBDcheck` gives an overview of the package and describes the methodology used, and `IBDcheck-hapmap` illustrates how to use the package to explore cryptic relatedness using genome-wide data from HapMap.

Author(s)

Annick Joelle Nembot-Simo, Jinko Graham and Brad McNeney

Maintainer: Brad McNeney <mcneney@sfu.ca>

See Also

rJPSGCS

`countIBS` *calculates counts of 0, 1 and 2 IBS*

Description

Returns counts of the number of markers at which a pair of subjects shares 0, 1 or 2 alleles identical-by-state (IBS), for all possible pairs. Counts for a given pair exclude markers where either pair member is missing data.

Usage

```
countIBS(x)
```

Arguments

x	An object of type snp.matrix

Value

IBS0	An upper triangular matrix whose (i,j)th element (i<j) is the count of non-missing markers where the ith and jth subjects share 0 alleles IBS.
IBS1	An upper triangular matrix whose (i,j)th element (i<j) is the count of non-missing markers where the ith and jth subjects share 1 allele IBS.
IBS2	An upper triangular matrix whose (i,j)th element (i<j) is the count of non-missing markers where the ith and jth subjects share 2 alleles IBS.

Author(s)

Annick Joelle Nembot-Simo, Jinko Graham and Brad McNeney

Examples

```
data(Nhlsim)
countIBS(Nhlsim$snp.data[1:5,])
```

---

filter.control *Set options for quality control filtering of data.*

Description

Set options for quality control filtering of data.

Usage

```
filter.control(filter=TRUE, snpcallrate=.9, MAF=.01, samplecallrate=.9,
               HWEp=.001)
```

Arguments

filter	Should data filtering be done? Default is TRUE.
snpcallrate	The SNP call rate is the proportion of non-missing genotypes in the study per SNP. SNPs with call rate less than snpcallrate are removed. The default value is 0.90.
MAF	The minimum minor allele frequency (MAF) for SNPs. SNPs with MAF less than MAF are removed. The default value is 0.01.

samplecallrate The proportion of SNPs for a sample that had genotype calls. Samples with call rate less than samplecallrate are removed. The default value is 0.90.

HWEp Threshold for the p-value from a 2-sided test of HWE. SNPs with p-value less than HWEp are removed. The default value is 0.001.

Value

A list whose components are the function inputs.

Author(s)

Annick Joelle Nembot-Simo, Jinko Graham and Brad McNeney

See Also

IBDcheck

Examples

```
ff<-filter.control()
ff<-filter.control(MAF=0.05)
```

IBD *Constructor function for objects of class* IBD

Description

Constructor function for objects of class IBD, used as both the input and output of the IBDcheck function. This is not intended to be called by users. Users preparing data for a call to IBDcheck should use the wrapper function new.IBD, which provides basic checks of the input.

Usage

```
IBD(snp.data, snp.support,subject.support, ibd.study=NULL,
    ibd.ur=NULL,ibd.mz=NULL,ibd.po=NULL,ibd.fs=NULL,ibd.hs=NULL,
    ibd.co=NULL, ibd.user=NULL, filterparams=NULL,simparams=NULL,call=NULL)
```

Arguments

snp.data A snp.matrix object containing the genotypes. Rows correspond to subjects and columns correspond to SNPs.

snp.support a data frame of SNP information

subject.support a data frame of subject information

ibd.study A data frame of estimated IBD coefficients for study subjects. Columns represent estimated probabilities of 0, 1 and 2 alleles IBD. Rows represent pairs of subjects.

ibd.ur	data frame of estimated IBD coefficients for simulated unrelated subjects
ibd.mz	data frame of estimated IBD coefficients for simulated monozygotic twins
ibd.po	data frame of estimated IBD coefficients for simulated parent-offspring subjects
ibd.fs	data frame of estimated IBD coefficients for simulated full sibling subjects
ibd.hs	data frame of estimated IBD coefficients for simulated half sibling subjects
ibd.co	data frame of estimated IBD coefficients for simulated cousins
ibd.user	data frame of estimated IBD coefficients for subjects having a user-defined relationship
filterparams	a list of data filtering options returned by filter.control
simparams	a list of data simulation options returned by sim.control
call	the IBDcheck call that created the ibd.* data frames

Details

Objects of class IBD are both the input and output of the IBDcheck function.

Value

An object of class IBD, which is a list comprised of the function arguments.

Author(s)

Annick Joelle Nembot-Simo, Jinko Graham and Brad McNeney

See Also

IBDcheck

IBDcheck *estimate IBD coefficients to identify cryptic relatedness in genetic association studies*

Description

Estimate IBD coefficients for pairs of study subjects. Optionally, IBD coefficients are estimated from simulated unrelated, monozygotic twin/duplicate, parent-offspring, full sibling, half sibling, or cousin pairs. Users may also specify their own relationships to simulate (see **Examples**). Simulations can make use of information about population linkage disequilibrium structure. The function returns an object of class IBD that can be graphically displayed by the plot method of the class, plot.IBD.

Usage

```
IBDcheck(dat, filterparams=filter.control(),simparams=sim.control())
```

Arguments

dat	An object of class IBD, created by new.IBD or by a previous call to IBDcheck.
filterparams	A list of parameters that control the filtering (e.g., quality control filtering) of the data. See filter.control for a description of these parameters.
simparams	A list of parameters that control simulation of data by gene drops. See sim.control for a description of these parameters.

Details

The required input to IBDcheck is an object of class IBD, created by new.IBD or by a previous call to IBDcheck. At a minimum, such an object includes the genetic data as a snp.matrix object from the **chopsticks** package (Leung 2012), a data frame of SNP information that includes chromosome and physical map positions of each SNP, and a data frame of subject information that includes a logical vector indicating whether (TRUE) or not (FALSE) each subject is to be used to estimate the conditional IBS probabilities and fit the LD model. Sex-chromosome SNPs in the IBD object are ignored by IBDcheck. SNPs and subjects are removed according the the filtering parameters set by the filter.control function. For SNPs and subjects that remain after filtering, IBD coefficients are estimated as described in Purcell *et al.* (2007). These proportions can be displayed graphically by the plotting function plot.IBD. When simulate=TRUE, IBDcheck simulates data that can be used to produce prediction ellipses for each simulated relationship on the graphical displays. Gene drop simulations to produce simulated pairs are done by the GeneDrops function from the **rJPSGCS** package. These simulations can be based on independent loci (fitLD=FALSE) or on a fitted LD model that accounts for inter-locus correlation (fitLD=TRUE). Parameters that control the simulations may be set by the sim.control function.

The package vignette, vignette("CrypticIBDcheck"), contains full details on the methods underlying IBDcheck. See also vignette("IBDcheck-hapmap") for an illustration of how to use IBDcheck to explore cryptic relatedness using genome-wide data from HapMap.

Value

An object of class IBD. See the help file for the constructor function IBD for a description of the components of this class.

Note

When simulate=TRUE and fitLD=TRUE, the function can be computationally demanding for data sets with more than about 1000 SNPs. In Appendix B of the vignette CrypticIBDcheck we describe strategies for making computations feasible by use of a **snow** cluster (Tierney *et al.*, 2011).

Users may also need to increase the amount of java heap space for some computations. The computation for fitting the LD model is done in java, using functions from Alun Thomas' suite of Java Programs for Statistical Genetics and Computational Statistics (JPSGCS), available at the website http://balance.med.utah.edu/wiki/index.php/JPSGCS. The JPSGCS java programs are accessed by R-wrappers provided by the **rJPSGCS** R package. When **rJPSGCS** is loaded (automatically by loading **CrypticIBDcheck**), it initializes the java Virtual Machine (JVM) via the **rJava** package, if not already done so by another package. One can set the amount of memory java can use for heap space by initilizing the JVM *before* loading **CrypticIBDcheck** as follows:

```
options(java.parameters="-Xmx2048m") #set max heap space to 2GB
library(rJava)
.jinit() #initialize the JVM
library(CrypticIBDcheck) # now load rJPSGCS by loading CrypticIBDcheck
```

Author(s)

Annick Joelle Nembot-Simo, Jinko Graham and Brad McNeney

References

Leung H-T (2012). chopsticks: The snp.matrix and X.snp.matrix classes. R package version 1.20.0. URL http://outmodedbonsai.sourceforge.net/

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MA, Bender D, Maller J, Sklar P, de Bakker PI, Daly MJ, Sham PC. PLINK: a tool set for whole-genome association and population-based linkage analyses. Am J Hum Genet. 2007 Sep;81(3):559-75.

Tierney L, Rossini AJ, Li N, Sevcikova H (2011). snow: Simple Network of Workstations. R package version 0.3-8, URL http://CRAN.R-project.org/package=snow.

See Also

plot.IBD, SNPgenmap

Examples

```
# These examples use the package dataset Nhlsim. For examples of how to
# prepare data in other formats for use by IBDcheck(), please see the
# documentation for new.IBD().
##################################################3
# Part I: No simulations
# Example 1) use all default settings; i.e., filter SNPs but do not simulate data
data(Nhlsim)
popsam<-Nhlsim$csct==0 # controls
dat<-new.IBD(Nhlsim$snp.data,Nhlsim$chromosome,Nhlsim$physmap,popsam)
cibd<-IBDcheck(dat)
##################################################3
## Not run:
# Part II: Simulate data assuming SNPs are in linkage equilibrium (no LD model fitted).
# Example 2) Simulate pairs of subjects of each of the default relationships:
# unrelated, MZ twins/duplicate, parent-offspring, full sibling, half sibling
# Use chromosomes 20, 21 and 22 only.
cind<-(Nhlsim$chromosome == 20 | Nhlsim$chromosome == 21 | Nhlsim$chromosome == 22)
dat<-new.IBD(Nhlsim$snp.data[,cind],Nhlsim$chromosome[cind],
                Nhlsim$physmap[cind],popsam)
ss<-sim.control(simulate=TRUE,fitLD=FALSE)
cibd2<-IBDcheck(dat,simparams=ss)

# Example 3) Add 100 more simulated unrelated pairs to the IBD object cibd2
ss<-sim.control(simulate=TRUE,fitLD=FALSE,rships="unrelated",nsim=100)
ff<-filter.control(filter=FALSE) # No need to re-filter the SNP data in cibd2
```

```
cibd3<-IBDcheck(cibd2,filterparams=ff,simparams=ss)

# Example 4) Add simulated first cousin pairs to the IBD object, cibd3, without any
# simulated first cousins. Simulate the default number of 200 pairs of first cousins.
ss<-sim.control(simulate=TRUE,fitLD=FALSE,rships="cousins")
ff<-filter.control(filter=FALSE) # No need to re-filter the SNP data in cibd3
cibd4<-IBDcheck(cibd3,filterparams=ff,simparams=ss)

# Example 5) Add simulated pairs for the user-specified relationship of
# mother-daughter, with mother and father who are first cousins. See the package
# vignette "CrypticIBDcheck", Figure 4, for a picture of this pedigree. Simulate
# the default number of 200 pairs of this user-specified relationship.
userdat<-data.frame(ids=1:9,
                    dadids=c(3,5,7,0,9,9,0,0,0),
                    momids=c(2,4,6,0,8,8,0,0,0),
                    gender=c(2,2,1,2,1,2,1,2,1))
ss<-sim.control(simulate=TRUE,fitLD=FALSE, rships=c("user"), userdat=userdat)
ff<-filter.control(filter=FALSE) # No need to re-filter the SNP data in cibd4
cibd5<-IBDcheck(cibd4,simparams=ss,filterparams=ff)
################################################
# Part III: Simulations based on a fitted LD model
# Example 6) Simulate pairs of subjects of each of the default relationships:
# unrelated, MZ twins/duplicate, parent-offspring, full sibling, half sibling.
# Use IBD object "dat" with SNPs from chromosomes 20, 21 and 22 only.
ss<-sim.control(simulate=TRUE,fitLD=TRUE)
cibd6<-IBDcheck(dat,simparams=ss)
# Save names of LD files for future simulations.
LDfiles<-cibd6$simparams$LDfiles

# Example 7) Use the fitted LD model from cibd6 to add 100 more simulated
# unrelated pairs
ss<-sim.control(simulate=TRUE,fitLD=TRUE,LDfiles=LDfiles,
rships="unrelated",nsim=100)
ff<-filter.control(filter=FALSE) # No need to re-filter the SNP data in cibd6
cibd7<-IBDcheck(cibd6,filterparams=ff,simparams=ss)
# NB: names of the LD files will be copied from cibd6 to cibd7

# Example 8) Use the fitted LD model from cibd6 to add simulated first cousins
# to an IBD object without any simulated first-cousin pairs. Add the default
# number of 200 simulated pairs of first cousins.
ss<-sim.control(simulate=TRUE,fitLD=TRUE,LDfiles=LDfiles,rships=c("cousins"))
ff<-filter.control(filter=FALSE) # No need to re-filter the SNP data in cibd7
cibd8<-IBDcheck(cibd7,simparams=ss,filterparams=ff)

# Example 9) Use the fitted LD model from cibd6 to add simulated pairs for the
# user-specified mother-daughter relationship, with mother and father who are
# first cousins. See the package vignette "CrypticIBDcheck", Figure 4, for a
# picture of this pedigree. Simulate the default number of 200 pairs of this
# user-specified relationship.
200 pairs.
userdat<-data.frame(ids=1:9,
                    dadids=c(3,5,7,0,9,9,0,0,0),
                    momids=c(2,4,6,0,8,8,0,0,0),
```

```
                    gender=c(2,2,1,2,1,2,1,2,1))
ss<-sim.control(simulate=TRUE,fitLD=TRUE, LDfiles=LDfiles, rships=c("user"),
userdat=userdat)
ff<-filter.control(filter=FALSE) # No need to re-filter the SNP data in cibd8
cibd9<-IBDcheck(cibd8,simparams=ss,filterparams=ff)

# Example 10) Distribute fitting of LD models and gene drop simulations for
# chromosomes 20, 21 and 22 across a snow cluster running on a local computer.
# See the package vignette, vignette("CrypticIBDcheck") for an example of
# running IBDcheck in batch mode on a compute cluster. Simulate pairs of
# subjects from each of the default relationships: unrelated, MZ twins/duplicate,
# parent-offspring, full sibling, half sibling.
cind<-(Nhlsim$chromosome == 20 | Nhlsim$chromosome == 21 | Nhlsim$chromosome == 22)
dat<-new.IBD(Nhlsim$snp.data[,cind],Nhlsim$chromosome[cind],
               Nhlsim$physmap[cind],popsam)
library(snow)
cl<-makeCluster(3,type="SOCK")
clusterEvalQ(cl,library("CrypticIBDcheck"))
ss<-sim.control(simulate=TRUE,fitLD=TRUE,cl=cl)
cibd3<-IBDcheck(dat,simparams=ss)
stopCluster(cl)


## End(Not run)
```

`new.IBD` *Create a data structure suitable for input to* `IBDcheck`

Description

Create a data structure suitable for input to `IBDcheck`. This is a wrapper function for the constructor function `IBD` that creates the object. The wrapper provides basic checks of the input and creates the `snp.support` and `subject.support` data frames required for an `IBD` object.

Usage

```
new.IBD(snp.data, Chromosome, Position, popsam, Gen_loc=NULL,
        pvalue_HWE=NULL, subids=NULL, ...)
```

Arguments

`snp.data`	A `snp.matrix` object containing the genotypes. Rows correspond to subjects and columns correspond to SNPs.
`Chromosome`	a vector containing the chromosome numbers of the SNPs
`Position`	a vector of physical map positions
`popsam`	A logical vector indicating whether each subject can be considered part of a random sample (`TRUE`) or not (`FALSE`). See **Details** for more information. Only those subjects for which `popsam==TRUE` are used for estimating conditional IBS probabilities and fitting LD models.

Gen_loc	A vector of genetic map positions in centiMorgans. If NULL (the default), they will be inferred using the function SNPgenmap.
pvalue_HWE	A vector of p-values from tests of Hardy-Weinberg proportions for each SNP. If NULL (the default), they will be filled in using all population sample subjects (popsam==TRUE) in snp.data.
subids	a vector of subject IDs
...	additional arguments to be passed to the constructor IBD

Details

The arguments snp.data, Chromosome, Position and popsam are required. Only subjects with popsam==TRUE are used for estimating conditional IBS probabilities and fitting LD models; those with popsam==FALSE are excluded. A typical use of popsam is to exclude cases when the data are from case control study of a rare disease, where cases are oversampled relative to their frequency in the population but controls may be regarded as a population sample.

If Gen_loc is missing it is inferred from Position by the SNPgenmap function. Currently SNPgenmap assumes physical map positions are on build 36 of the human genome. If subids is missing, the row names of snp.data are used as subject identifiers.

Value

An object of class IBD. See the help file for the constructor function IBD for details.

Author(s)

Annick Joelle Nembot-Simo, Jinko Graham and Brad McNeney

See Also

IBDcheck

Examples

```
data(Nhlsim)
popsam<-Nhlsim$csct==0 #controls
dat<-new.IBD(Nhlsim$snp.data,Nhlsim$chromosome,Nhlsim$physmap,popsam)

## Not run:
# Read PLINK-formatted data via the read.snps.pedfile() function of the
# chopsticks package
# [source("http://bioconductor.org/biocLite.R"); biocLite("chopsticks") to install]
# Assume PLINK data is in files mydata.ped and mydata.map.
require(chopsticks)
mydata = read.snps.pedfile("mydata.ped")
# mydata is now a list with elements mydata$snp.data, mydata$snp.support and
# mydata$subject.support. Chromosome number and physical position of the SNPs
# are in mydata$snp.support$chromosome and mydata$snp.support$position, respectively.
dat<-new.IBD(mydata$snp.data,mydata$snp.support$chromosome,
             mydata$snp.support$position,popsam=rep(1,nrow(mydata$snp.support)))
```

```
## End(Not run)
```

Nhlsim *Example data for the CrypticIBDcheck package*

Description

A dataset that contains genotypes simulated by gene drop based on a model of linkage disequilibrium fit to data from a candidate-gene case-control study. Several close relative pairs have been included in the simulated data.

Usage

```
data(Nhlsim)
```

Format

A list comprised of the following four objects:

[[1]]	snp.data	a `snp.matrix` object containing genotypes of 208 subjects (108 controls and 100 cases). Rows correspond to subjects and columns correspond to SNPs
[[2]]	chromosome	a numeric vector containing the chromosome numbers of the SNPs
[[3]]	physmap	a numeric vector of physical positions
[[4]]	csct	a vector of case-control status (1=case, 0=control)

Details

The dataset contains mostly unrelated individuals, but includes two parent-offspring pairs and three full-siblings pairs to show how the CrypticIBDcheck package can be used to uncover cryptic relatedness.

Source

The genotypes were simulated based on data from a candidate-gene case-control study described in Schuetz et al. (2012).

References

Schuetz JM, Daley D, Graham J, Berry BR, Gallagher RP, Connors JM, Gascoyne RD, Spinelli JJ, Brooks-Wilson AR (2012). Genetic Variation in Cell Death Genes and Risk of Non-Hodgkin Lymphoma. PLoS ONE, 7(2), e31560. doi:10.1371/journal.pone.0031560.

Examples

```
data(Nhlsim)
```

plot.IBD *plot estimated IBD coefficients for pairs of study subjects, along with prediction ellipses based on simulated pairs of known relationships*

Description

Interactive graphical display of an IBD object.

Usage

```
## S3 method for class 'IBD'
plot(x,kinshipth=NULL, ellipse.coverage=.95,...)
```

Arguments

x	an IBD object returned by IBDcheck
kinshipth	Kinship coefficient threshold. If NULL (the default), all study pairs will be included on the first plot summarizing the study pairs. If a numeric value, study pairs with estimated kinship coefficient less than this threshold value will be suppressed on the first plot. When simulate=TRUE and unrelated pairs are simulated, users may specify kinshipth="empirical" to use the 99th percentile of estimated kinship coefficients in simulated unrelated pairs as the threshold value.
ellipse.coverage	Prediction ellipse coverage probability. Simulated pairs of subjects from a given relationship are used to construct prediction ellipses with approximate coverage probability ellipse.coverage. See **Details** for details.
...	optional arguments passed to plot

Details

Overview: When simulate=FALSE, the function produces an interactive plot of estimated IBD coefficients for pairs of study subjects whose estimated kinship coefficients exceed the user-specified threshold in kinshipth. Plots are of the estimated probability of 1 IBD versus the estimated probability of 0 IBD for pairs of study subjects, with prediction ellipses for known relationships superposed, if requested by the user with simulate=TRUE. The prediction ellipses are produced from estimated IBD coefficients for a user-specified number (default 200) of simulated pairs of known relationships, assuming the distribution of estimated IBD coefficients is approximately bivariate Normal. When simulated pairs are omitted (simulate=FALSE), plotting produces a single interactive display of estimated IBD coefficients for pairs of study subjects specified by kinshipth, on which points may be identified by clicking with the mouse. By contrast, when the IBD object includes simulated pairs, the function returns a series of plots, which the user is prompted to view and interact with successively. The first plot to appear is non-clickable and shows the estimated IBD coefficients for pairs of study subjects specified by kinshipth, along with the prediction ellipse for unrelated, simulated pairs. Subsequent plots are clickable and correspond to each relationship requested in the call to IBDcheck(). These relationship-specific plots are for identifying pairs of

study subjects which could have the relationship. The plotting regions are restricted to the neighborhood of the prediction ellipse for the simulated pairs of that relationship, which is also drawn. If, however, the plotting region overlaps with the prediction ellipse for simulated unrelated pairs, the ellipse for simulated unrelated pairs is drawn as well. Points falling within the prediction ellipse for the relationship and outside the prediction ellipse for unrelated pairs are automatically flagged. In addition, users may click on points of study pairs that appear to be related but are not automatically flagged. The plot method produces a data frame of information on pairs that have been flagged on the different plots, either automatically or interactively by the user through clicking the mouse.

Additional details: The showLabels function in the **car** package is used to implement identification of points by left-clicking the mouse on interactive plots. When finished identifying points, users should right-click the plotting region to move to the next plot. A Bonferroni-type adjustment is applied when computing the ellipse for unrelated pairs to account for the fact that, typically, most study pairs will be unrelated. Specifically, the coverage probability for the simulated unrelated pairs ellipse is taken to be $1 - (1 - \texttt{ellipse.coverage})/n_p$, where n_p is the number of pairs of study subjects.

Value

A data frame of information on pairs that are flagged, either automatically or by user mouse-clicks, on the different plots provided by the function. The columns of this data frame are:

member1	ID of the first member of the study pair
member2	ID of the second member of the study pair
pz0	estimated proportion of markers with zero alleles IBD
pz1	estimated proportion of markers with one allele IBD
relationship	If simulate=TRUE, this column indicates which relationship plot each pair was identified on. This column is absent from the data frame when simulate=FALSE.

Author(s)

Annick Joelle Nembot-Simo, Jinko Graham and Brad McNeney

See Also

IBDcheck, showLabels

Examples

```
data(Nhlsim)
## Not run:
# Example with simulate=FALSE (default)
popsam<-Nhlsim$csct==0 # controls
dat<-new.IBD(Nhlsim$snp.data,Nhlsim$chromosome,Nhlsim$physmap,popsam)
cibd<-IBDcheck(dat)
plot(cibd)
# Example with simulate=TRUE. Use chromosomes 20, 21 and 22 only.
cind<-(Nhlsim$chromosome == 20 | Nhlsim$chromosome == 21 | Nhlsim$chromosome == 22)
dat<-new.IBD(Nhlsim$snp.data[,cind],Nhlsim$chromosome[cind],
               Nhlsim$physmap[cind],popsam)
```

```
ss<-sim.control(simulate=TRUE,fitLD=TRUE)
cibd2<-IBDcheck(dat,simparams=ss)
plot(cibd2)

# Example use of kinshipth argument: On the plot of study subjects, only plot
# those with kinship coefficient greater than the 99th percentile of the
# kinship coefficients of simulated unrelated individuals.
plot(cibd2,kinshipth="empirical")

## End(Not run)
```

`RutgersMapB36` *physical and genetic map positions for a collection of markers on chromosome 1 to 22*

Description

This dataset is list comprised of data frames corresponding to chromosomes 1 to 22. Each data frame has information on physical and genetic map positions for a collection of markers on the corresponding chromosome.

Usage

```
data(RutgersMapB36)
```

Format

The format of the dataset is a list of 22 elements. Each element is a data frame corresponding to a chromosome with nine columns:

[,1]	Markers_name	factor	marker name
[,2]	Type	factor	marker type
[,3]	Primer.SNP_ref_name	factor	primer or SNP name
[,4]	Informative_meioses	integer	number of informative meioses
[,5]	Heterozygosity	numeric	marker heterozygosity
[,6]	Build36_map_physical_position	integer	physical position (in base pairs) on build 36 of the human genome
[,7]	Sex.averaged_map_position	numeric	sex-averaged map position
[,8]	Female_map_position	numeric	female map position
[,9]	Male_map_position	numeric	male map position

Source

The data were obtained from the Rutgers map (see http://compgen.rutgers.edu/RutgersMap/DownloadMap.aspx)

References

A second-generation combined linkage physical map of the human genome. Matise TC, Chen F, Chen W, De La Vega FM, Hansen M, He C, Hyland FC, Kennedy GC, Kong X, Murray SS, Ziegle JS, Stewart WC, Buyske S. Genome Res. 2007 Dec;17(12):1783-6. Epub 2007 Nov 7.

Examples

```
data(RutgersMapB36)
```

---

sim.control *Set options that control gene drop simulation of relationship pairs.*

Description

Set options that control gene drop simulation of relationship pairs.

Usage

```
sim.control(simulate=FALSE,
    rships=c("unrelated","MZtwins","parent-offspring","full-sibs","half-sibs"),
    nsim=rep(200,length(rships)), userdat=NULL,
    geno.err=1/1000, hom2hom.err=0, fitLD=TRUE, LDfiles=NULL, cl=NULL)
```

Arguments

simulate	Should data be simulated by gene drop to allow the user to assign relationships to outlying pairs? Default is FALSE. See **Details** for more information.
rships	A character vector specifying the relationships to simulate. The choices are currently "unrelated", "MZtwins", "parent-offspring", "full-sibs", "half-sibs", "cousins", or "user", for unrelated, monozygotic-twin/duplicate, parent-offspring, full-sibling, half-sibling, first-cousin, or user-defined relationship pairs, respectively. The default relationships are "unrelated", "MZtwins", "parent-offspring", "full-sibs" and "half-sibs". Please note that half-sibling, avuncular and grandparent-grandchild relationships cannot be distinguished on the basis of IBD coefficients. Partial matches (e.g., "par" rather than "parent-offspring") are allowed, as shown in the **Examples**.
nsim	Numeric vector of numbers of pairs to be simulated by gene drop for each relationship. The default is 200 for each relationship listed in rships.
userdat	A data frame of information on the pedigree to simulate for a user-defined relationship pair. The columns of this data frame must be named as follows: ids, the IDs of the members of the pedigree; dadids, the IDs of each pedigree member's father, or zero if the father is not in the pedigree; momids, the IDs of each pedigree member's mother, or zero if the mother is not in the pedigree; gender, the gender of each subject coded as 1 for male and 2 for female. See **Examples** for an example. Default is NULL.

geno.err	Genotyping error rate. Each genotype is sampled with probability geno.err to be measured incorrectly, according to a simple error model. In the error model, heterozygous genotypes are equally likely to be called as either of the homozygous genotypes. Homozygous genotypes are called as the other homozygous genotype with probability hom2hom.err and as heterozygous with probability 1-hom2hom.err. The default value of geno.err is 0.001.
hom2hom.err	The probability a homozygous genotype that is miscalled is miscalled as the other homozygous genotype. Default is 0, so that miscalled homozyous genotypes are always called heterozygous.
fitLD	Should an LD model be fit to the data for use in gene drop simulations? Default is TRUE. Ignored if simulate=FALSE. See **Details** for more information.
LDfiles	Character vector of the names of files containing LD models fit by FitGMLD. These may be present from a previous call to IBDcheck. There must be one file name for each chromosome of data in snp.data, and the files must be ordered by chromosome number. Default is NULL. Ignored if fitLD=FALSE or simulate=FALSE.
cl	A SNOW cluster that can be used to split fitting of LD models and gene drop simulations across a compute cluster. Default is NULL.

Details

When simulate=TRUE, IBDcheck simulates data from pairs with known relationship that can be used to generate prediction ellipses on the graphical displays as a reference. Unrelated, parent-offspring, full sibling, half sibling, cousin, or user-defined relationships are simulated by gene drop and their estimated IBD coeficients are computed as for pairs of study subjects. Monozygotic twins/duplicates are not simulated by gene drop. Rather, they are simulated by randomly sampling a study individual and then applying the genotyping error model twice to make two copies. Gene drop simulations can be based on loci in linkage equilibrium (fitLD=FALSE) or on a fitted LD model that accounts for inter-locus correlation (fitLD=TRUE).

Value

A list whose components are the function inputs.

Author(s)

Annick Joelle Nembot-Simo, Jinko Graham and Brad McNeney

See Also

IBDcheck

Examples

```
# Set simulation parameters to simulate unrelated, parent-offspring and
# full-sibling pairs. Leave other simulation parameters at their
# default values (e.g., nsim=rep(200,length(rships)), fitLD=TRUE).
ss<-sim.control(simulate=TRUE,rships=c("unrel","parent","full"))
## Not run:
```

```
# Create an IBD object to use as input to IBDcheck.
data(Nhlsim)
popsam<-Nhlsim$csct==0 # controls
# Use chromosomes 20, 21 and 22 only.
cind<-(Nhlsim$chromosome==20|Nhlsim$chromosome==21|Nhlsim$chromosome==22)
dat<-new.IBD(Nhlsim$snp.data[,cind],Nhlsim$chromosome[cind],
                Nhlsim$physmap[cind],popsam)
# Run IBDcheck
cibd<-IBDcheck(dat,simparams=ss)
# Save the names of the LD files for future simulations.
LDfiles<-cibd$simparams$LDfiles

# Use the fitted LD model from cibd to add 100 more simulated, unrelated pairs
# and save the updated IBD object in cibd2.
ss<-sim.control(simulate=TRUE,LDfiles=LDfiles,rships="unrelated",nsim=100)
ff<-filter.control(filter=FALSE) # No need to re-filter the SNP data in cibd
cibd2<-IBDcheck(cibd2,filterparams=ff,simparams=ss)

# Add 200 simulated first-cousin pairs to cibd2, an IBD object which has no
# simulated first-cousin pairs. Save the updated IBD object in cibd3.
ss<-sim.control(simulate=TRUE,LDfiles=LDfiles,rships="cousins")
ff<-filter.control(filter=FALSE) # No need to re-filter the SNP data in cibd2
cibd3<-IBDcheck(cibd2,filterparams=ff,simparams=ss)

# Add 200 simulated pairs having the user-specified mother-daughter relationship,
# with mother and father being first cousins. See the package vignette
# "CrypticIBDcheck", Figure 4, for a picture of this pedigree. Save the updated
# IBD object in cibd4.
userdat<-data.frame(ids=1:9,
                    dadids=c(3,5,7,0,9,9,0,0,0),
                    momids=c(2,4,6,0,8,8,0,0,0),
                    gender=c(2,2,1,2,1,2,1,2,1))
ss<-sim.control(simulate=TRUE,LDfiles=LDfiles,rships=c("user"),userdat=userdat)
ff<-filter.control(filter=FALSE) # No need to re-filter the SNP data in cibd3
cibd4<-IBDcheck(cibd3,simparams=ss,filterparams=ff)

# Distribute fitting of LD models and gene drop simulations for each
# chromosome across a snow cluster running on a local computer.  See the
# package vignette, vignette("CrypticIBDcheck") for an example of running
# IBDcheck in batch mode on a compute cluster. Save the updated IBD object
# in cibd5.
library(snow)
cl<-makeCluster(3,type="SOCK")
clusterEvalQ(cl,library("CrypticIBDcheck"))
ss<-sim.control(simulate=TRUE,cl=cl) # Leave all other sim params at defaults
cibd5<-IBDcheck(dat,simparams=ss)
stopCluster(cl)

## End(Not run)
```

SNPgenmap *convert physical map positions on build 36 of the genome to genetic map positions*

Description

Convert physical map positions on build 36 of the genome to genetic map positions by linear interpolation of the Rutgers combined linkage-physical map. The markers in the Rutgers map are a small subset of markers for which genetic map positions have been determined. Linear interpolation is done for points in between.

Usage

```
SNPgenmap(physmap, chromosomes)
```

Arguments

physmap	a vector of physical map positions on build 36 of the human genome
chromosomes	a vector containing the corresponding chromosome numbers

Details

Genetic map positions are inferred from physical positions by linear interpolation of the Rutgers Combined Linkage-Physical Map for build 36 of the human genome, contained in the data object RutgersMapB36. Users who want some other form of interpolation can do so themselves using RutgersMapB36, as illustrated in the **Examples**. *NB:* The order of markers in RutgersMapB36 is the same for both physical and genetic maps. In order for an interpolated genetic map to preserve the ordering of physical map positions, the interpolant must be monotone increasing. Linear interpolation *is* monotone increasing, but other forms, such as spline interpolation, may not be.

Value

The function returns a vector of genetic map positions.

Note

The function interpolates the Rutgers map and does not attempt to extrapolate for SNPs outside the map. Genetic map positions for SNPs outside the Rutgers map are set to NA.

Author(s)

Annick Joelle Nembot-Simo, Jinko Graham and Brad McNeney

See Also

RutgersMapB36

Examples

```
data(Nhlsim)
gmap <- SNPgenmap(Nhlsim$physmap,Nhlsim$chromosome)

# Example of using RugtersMapB36 to do spline rather than linear
# interpolation of genetic map positions on chromosome 1.
# NB: Interpolant is not necessarily monotone increasing, which can lead to a
# genetic map on which markers are re-ordered relative to the physical map.
chrmap<-splinefun(RutgersMapB36[["chr1"]]$Build36_map_physical_position,
                  RutgersMapB36[["chr1"]]$Sex.averaged_map_position)
c1ind<-(Nhlsim$chromosome=="chr1")
gmap[c1ind]<-chrmap(Nhlsim$physmap[c1ind])
```

Index