Package 'sequoia'

June 20, 2019

Type Package

Title Pedigree Inference from SNPs

Version 1.3.3

Date 2019-06-20

Description Fast multi-generational pedigree inference from incomplete data on hundreds of SNPs, including parentage assignment and sibship clustering. See Huisman (2017) (DOI:10.1111/1755-0998.12665, citation('sequoia')) for more information.

License GPL-2

LazyData TRUE

Imports plyr (>= 1.8.0), stats, utils, graphics

RoxygenNote 6.1.0

Suggests xlsx, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Jisca Huisman [aut, cre]

Maintainer Jisca Huisman <jisca.huisman@gmail.com>

Repository CRAN

Date/Publication 2019-06-20 14:50:02 UTC

R topics documented:

`CheckGeno` *check GenoM*

Description

Check that the provided genotype matrix is in the correct format

Usage

```
CheckGeno(GenoM, quiet = FALSE)
```

Arguments

`GenoM`	the genotype matrix
`quiet`	suppress messages

Value

a list with low call rate individuals (<0.5, 'ExcludedInd') and low call rate SNPs (<0.1, 'ExcludedSnps'), if any, which will be excluded from pedigree reconstruction.

DyadCompare *Compare dyads*

Description

Count the number of half and full sibling pairs correctly and incorrectly assigned

Usage

```
DyadCompare(Ped1 = NULL, Ped2 = NULL, na1 = c(NA, "0"))
```

Arguments

Ped1	Original pedigree, dataframe with 3 columns: id-dam-sire
Ped2	Second (inferred) pedigree
na1	the value for missing parents in Ped1.

Value

A 3x3 table with the number of pairs assigned as full siblings (FS), half siblings (HS) or unrelated (U, including otherwise related) in the two pedigrees, with the classification in Ped1 on rows and that in Ped2 in columns

See Also

PedCompare

Examples

```
## Not run:
data(Ped_HSg5, SimGeno_example, LH_HSg5, package="sequoia")
SeqOUT <- sequoia(GenoM = SimGeno_example, LifeHistData = LH_HSg5,
                  MaxSibIter = 0)
DyadCompare(Ped1=Ped_HSg5, Ped2=SeqOUT$Pedigree)

## End(Not run)
```

EstConf *Estimate confidence probability*

Description

Estimate the assignment error rate by repeatedly simulating data from a reference pedigree using SimGeno, reconstruction a pedigree from this using sequoia, and counting the number of mismatches using PedCompare.

Usage

```
EstConf(Ped = NULL, LifeHistData = NULL, args.sim = list(nSnp = 400,
  SnpError = 0.001, ParMis = c(0.4, 0.4)), args.seq = list(MaxSibIter =
  10, Err = 1e-04, Tassign = 0.5), nSim = 10, return.PC = FALSE,
  quiet = TRUE)
```

Arguments

Ped	Reference pedigree from which to simulate, dataframe with columns id-dam-sire. Additional columns are ignored
LifeHistData	Dataframe with id, sex (1=female, 2=male, 3=unknown), and birth year.
args.sim	list of arguments to pass to SimGeno, such as nSnp (number of SNPs), SnpError (genotyping error rate) and ParMis (proportion of non-genotyped parents). Set to NULL to use all default values.
args.seq	list of arguments to pass to sequoia, such as MaxSibIter (max no. sibship clustering iterations, '0' for parentage assignment only) and Err (assumed genotyping error rate). May include (part of) SeqList, the list of sequoia output (i.e. as a list-within-a-list). Set to NULL to use all default values.
nSim	number of rounds of simulations to perform.
return.PC	return all PedCompare Counts?
quiet	suppress messages. 'very' also suppresses simulation counter, TRUE merely runs SimGeno and sequoia quietly.

Details

The confidence probability is taken as the number of correct (matching) assignments, divided by all assignments made. A confidence of '1' should be interpreted as '> 1 - 1/(sum(!is.na(Ped$dam)) * nSim)'

Value

When return.PC = FALSE, a 2x2 matrix for parentage assignment, or a 2x7x2 array for full pedigree reconstruction, with for dams and sires and per category (see PedCompare) the average and minimum number of Match/(Match + Mismatch + P2only).

When return.PC is TRUE, a list is returned with:

ConfProb	Average confidence probability across simulations, as returned when return.PC = FALSE.
SimCounts	All counts of matches, mismatches, Pedigree1-only and pedigree2-only, per simulation.
RunParams	Current call to EstConf, as well as the default parameter values for EstConf, SimGeno, and sequoia.
RunTime	sequoia runtime per simulation in seconds, as measured by system.time()['elapsed'].

See Also

SimGeno, sequoia, PedCompare

Examples

```
## Not run:
data(SimGeno_example, LH_HSg5, package="sequoia")

conf.A <- EstConf(Ped = Ped_HSg5, LifeHistData = LH_HSg5,
   args.sim = list(nSnp = 100, SnpError = 5e-3, ParMis=c(0.2, 0.5)),
   args.seq = list(MaxSibIter = 0, Err=1e-4, Tassign=0.5),
   nSim = 3, return.PC = TRUE)

# effect of tweaking AgePriors
# (only some effect due to low no. SNPs & high error rate,
#  effect of increasing no. SNPs is much larger)
AP <- MakeAgePrior(Ped = Ped_HSg5, LifeHistData = LH_HSg5,
                   Flatten = FALSE, Smooth = FALSE)
conf.B <- EstConf(Ped = Ped_HSg5, LifeHistData = LH_HSg5,
   args.sim = list(nSnp = 100, SnpError = 5e-3, ParMis=c(0.2, 0.5)),
   args.seq = list(MaxSibIter = 0, Err=1e-4, Tassign=0.5,
                   SeqList = list(AgePriors = AP)),
   nSim = 3, return.PC = TRUE)

# with sibship clustering
conf.C <- EstConf(Ped = Ped_HSg5, LifeHistData = LH_HSg5,
   args.sim = list(nSnp = 200, SnpError = 5e-3, ParMis=c(0.2, 0.5)),
   args.seq = list(MaxSibIter = 10, Err=1e-4, Tassign=0.5),
   nSim = 3, return.PC = TRUE)
conf.C$ConfProb[,"GG",]  # Genotyped individuals, Genotyped parent
conf.C$ConfProb[,"GD",]  # Genotyped individuals, Dummy parent
AR <- apply(conf.C$SimCounts, 1, function(M) M["TT","Match", ]/M["TT","Total", ])
ER <- apply(conf.C$SimCounts, 1,
       function(M) (M["TT","Mismatch", ] + M["TT","P2only", ])/M["TT","Total", ])
apply(ER, 1, mean)  # separate error rate dams & sires
mean(ER)            # overall error rate

## End(Not run)
```

FindFamilies *Assign family IDs*

Description

Add a column with family IDs (FIDs) to a pedigree, with each number denoting a cluster of connected individuals.

Usage

```
FindFamilies(Ped = NULL, SeqList = NULL, UseMaybeRel = FALSE)
```

Arguments

Ped	dataframe with columns id - parent1 - parent2; only the first 3 columns will be used.
SeqList	list as returned by sequoia. If 'Ped' is not provided, the element 'Pedigree' from this list will be used if present, and element 'Pedigreepar' otherwise.
UseMaybeRel	use SeqList$MaybeRel, the dataframe with probable but non-assigned relatives, to assign additional family IDs?

Details

This function repeatedly finds all ancestors and all descendants of each individual in turn, and ensures they all have the same Family ID. Not all connected individuals are related, e.g. all grandparents of an individual will have the same FID, but will typically be unrelated.

When UseMaybeRel = TRUE, probable relatives are added to existing family clusters, or existing family clusters may be linked together. Currently no additional family clusters are created.

Value

A dataframe with the provided pedigree, with a column 'FID' added.

GenoConvert *Convert genotype data*

Description

Convert genotype data in various formats to sequoia's 1-column-per-marker format or Colony's 2-column-per-marker format.

Usage

```
GenoConvert(InFile = NULL, InFormat = "raw", OutFile = NA,
  OutFormat = "seq", InData = NULL, Missing = c("-9", "??", "?",
  "NA", "NULL", c("0")[InFormat %in% c("col", "ped")]), sep = c(" ",
  "\t", ",", ";"), header = NA, IDcol = NA, FIDcol = NA,
  FIDsep = "__", dropcol = NA, quiet = FALSE)
```

Arguments

InFile	character string with name of genotype file to be converted
InFormat	One of 'single', 'double', 'col', 'ped', 'raw', or 'seq', see Details.
OutFile	character string with name of converted file. If NA, return matrix with genotypes in console (default); if NULL, write to 'GenoForSequoia.txt' in current working directory.
OutFormat	as InFormat, currently only 'seq' and 'col' are implemented.
InData	dataframe or matrix with genotypes to be converted
Missing	vector with symbols interpreted as missing data.
sep	vector with field separator strings that will be tried on InFile. The OutFile separator uses the write.table default, i.e. one blank space
header	a logical value indicating whether the file contains the names of the variables as its first line. If NA (default), set to TRUE for 'raw', and FALSE otherwise.
IDcol	single number giving the column which contains the individual IDs; 0 indicates the rownames (for InData only). If NA (default), set to 2 for InFormat 'raw' and 'ped', and otherwise to 1 for InFile and 0 (rownames) for InData, except when InData has a column labeled 'ID'.
FIDcol	column which contains the individual IDs, if any are wished to be used. This is column 1 for InFormat 'raw' and 'seq', but those are by default not used.
FIDsep	string used to paste FID and IID together into a composite-ID (value passed to paste's collapse).
dropcol	columns to exclude from the output data, on top of IDcol and FIDcol (which become rownames). When NA, defaults to columns 3-6 for InFormat 'raw' and 'seq'. Can also be used to drop some SNPs, see example below on how to do this for the 2-columns-per-SNP input formats.
quiet	suppress messages and warnings

Value

A genotype matrix in the specified output format. If 'OutFile' is specified, the matrix is written to this file and nothing is returned inside R. When converting to 0/1/2 format, 2 is the homozygote for the minor allele, and 0 the homozygote for the major allele.

Input formats

The following formats can be specified by `InFormat`:

single 1 column per marker, otherwise unspecified

double 2 columns per marker, otherwise unspecified

col (Colony) genotypes are coded as numeric values, missing as 0, in 2 columns per marker. Column 1 contains IDs.

ped (PLINK) genotypes are coded as A, C, T, G, missing as 0, in 2 columns per marker. The first 6 columns are descriptive (1:FID, 2:IID, 3 to 6 ignored).

raw (PLINK) genotypes are coded as 0, 1, 2, missing as NA, in 1 column per marker. The first 6 columns are descriptive (1:FID, 2:IID, 3 to 6 ignored), and there is a header row.

seq (sequoia) genotypes are coded as 0, 1, 2, missing as -9, in 1 column per marker. Column 1 contains IDs, there is no header row.

For each InFormat, its default values for `Missing, header, IDcol, FIDcol`, and `dropcol` can be overruled by specifying the corresponding input parameters.

Error messages

An occassional error when reading in a file with GenoConvert is that 'rows have unequal length'. GenoConvert makes use of `readLines` and `strsplit`, which is much faster than `read.table` for large datafiles, but also more sensitive to unusual line endings, unusual end-of-file characters, or invisible characters (spaces or tabs) after the end of some lines. In these cases, try to read the data from file using read.table or read.csv, and then use GenoConvert on the matrix, see example.

Author(s)

Jisca Huisman, <jisca.huisman@gmail.com>

See Also

`SnpStats`, `LHConvert`, and `PedStripFID` to reverse joining FID and IID

Examples

```
## Not run:
# Requires PLINK installed & in system PATH:

# tinker with window size, window overlap and VIF to get a set of
# 400 - 800 markers (100-200 enough for just parentage):
system("cmd", input = "plink --file mydata --indep 50 5 2")
system("cmd", input = "plink --file mydata --extract plink.prune.in
  --recodeA --out PlinkOUT")

GenoM <- GenoConvert(InFile = "PlinkOUT.raw")

# save time on file conversion next time:
write.table(GenoM, file="Geno_for_sequoia.txt", quote=FALSE,
  col.names=FALSE)
```

```
GenoM <- read.table("Geno_for_sequoia.txt", row.names=1, header=FALSE)

# drop some SNPs, e.g. after a warning of >2 alleles:
dropSNP <- c(5,68,101,128)
GenoM <- GenoConvert(ColonyFile, InFormat = "col",
                     dropcol = 1 + c(2*dropSNP-1, 2*dropSNP) )

# circumvent a 'rows have unequal length' error:
GenoTmp <- as.matrix(read.table("mydata.txt", header=TRUE, row.names=1))
GenoM <- GenoConvert(InData=GenoTmp, InFormat="single", IDcol=0)

## End(Not run)
```

`GetMaybeRel` *Find putative relatives*

Description

Identify pairs of individuals likely to be related, but not assigned as such in the provided pedigree.

Usage

```
GetMaybeRel(GenoM = NULL, SeqList = NULL, Ped = NULL,
  LifeHistData = NULL, ParSib = "par", Complex = "full",
  Err = 1e-04, MaxMismatch = 3, Tassign = 0.5, MaxPairs = 7 *
  nrow(GenoM), DumPrefix = c("F0", "M0"), quiet = FALSE)
```

Arguments

`GenoM`	matrix with genotype data, size nInd x nSnp
`SeqList`	list with output from `sequoia`. If provided, the elements 'Specs', 'AgePriors' and 'LifeHist' are used, and all other input parameters except 'GenoM', 'ParSib' and 'quiet' are ignored.
`Ped`	dataframe with pedigree, with id - dam - sire in columns 1-3
`LifeHistData`	dataframe with columns id - sex (1=female, 2=male, 3=unknown) - birth year
`ParSib`	either 'par' to check for putative parent-offspring pairs only, or 'sib' to check for all types of first and second degree relatives. When 'par', all pairs are returned that are more likely parent-offspring than unrelated, including pairs that are even more likely to be otherwise related.
`Complex`	either "full" (default), "simp" (simplified, no explicit consideration of inbred relationships), "mono" (monogamous) or "herm" (hermaphrodites, otherwise like "full").
`Err`	estimated genotyping error rate. The error model aims to deal with scoring errors typical for SNP arrays.

MaxMismatch	maximum number of loci at which candidate parent and offspring are allowed to be opposite homozygotes. Setting a more liberal threshold can improve performance if the error rate is high, at the cost of decreased speed.
Tassign	minimum LLR required for acceptance of proposed relationship, relative to next most likely relationship. Higher values result in more conservative assignments. Must be zero or positive.
MaxPairs	The maximum number of putative pairs to return.
DumPrefix	character vector of length 2 with prefixes for dummy dams (mothers) and sires (fathers) used in Ped.
quiet	suppress messages

Value

A list with

MaybeParent or MaybeRel	Non-assigned likely relatives
MaybeTrio	Non-assigned parent-parent-offspring trios

Examples

```
## Not run:
data(SimGeno_example, LH_HSg5, package="sequoia")
SeqOUT <- sequoia(GenoM = SimGeno_example,
                  LifeHistData = LH_HSg5, MaxSibIter = 0)
MaybePO <- GetMaybeRel(GenoM = SimGeno_example,
                       SeqList = SeqOUT)

# age differences limit which relationships are considered:
MaybePO3 <- GetMaybeRel(GenoM = SimGeno_example,
                       Ped = SeqOUT$PedigreePar)

## End(Not run)
```

GetRelCat *Pairwise relationship*

Description

Determine the relationship between individual X and all other individuals in the pedigree, going up to 1 or 2 generations back.

Usage

```
GetRelCat(x, Ped, GenBack = 2)
```

Arguments

x	The focal individual, either its rownumber in the pedigreee or ID.
Ped	dataframe with pedigree; columns id - dam - sire.
GenBack	Number of generations back to consider; 1 returns parent-offspring and sibling relationships, 2 also returns grandparental, avuncular and first cousins.

Details

if

Value

A named vector of length equal to the number of rows in Ped, with for each ID its relationship to the focal individual:

S	Self
M	Mother
P	Father
O	Offspring
FS	Full sibling
MHS	Maternal half-sibling
PHS	Paternal half-sibling
MGM	Maternal grandmother
MGF	Maternal grandfather
PGM	Paternal grandmother
PGF	Paternal grandfather
GO	Grand-offspring
FA	Full avuncular; maternal or paternal aunt or uncle
HA	Half avuncular
FN	Full nephew/niece
HN	Half nephew/niece
FC1	Full first cousin
DFC1	Double full first cousin
U	Unrelated (or otherwise related)

Examples

```
## Not run:
# make NxN matrix with relationship categories:
RCM <- sapply(1:nrow(Ped), GetRelCat, Ped)
# table is considerably faster on a factor than a character vector:
table(factor(RCM, levels=c("M","P","FS","MHS","PHS","MGM")))
# note that sibling & cousin pairs are counted twice!

## End(Not run)
```

Inherit *Inheritance patterns*

Description

Inheritance patterns used by SimGeno for non-autosomal SNPs, identical to those in Inherit.xlsx

Usage

```
data(Inherit)
```

Format

An array with the following dimensions:

d1 type: autosomal, x-chromosome, y-chromosome, or mtDNA

d2 offspring sex: female, male, or unknown

d3 offspring genotype: aa (0), aA (1), Aa (1), or AA (2)

d4 mother genotype

d5 father genotype

Author(s)

Jisca Huisman, <jisca.huisman@gmail.com>

See Also

SimGeno

LHConvert *Extract sex and birthyear from PLINK file*

Description

Convert the first six columns of a PLINK .fam, .ped or .raw file into a three-column lifehistory file for sequoia. Optionally FID and IID are combined.

Usage

```
LHConvert(PlinkFile = NULL, UseFID = FALSE, SwapSex = TRUE,
  FIDsep = "__", LifeHistData = NULL)
```

Arguments

PlinkFile	character string with name of genotype file to be converted
UseFID	Use the family ID column. The resulting ids (rownames of GenoM) will be in the form FID__IID
SwapSex	change the coding from PLINK default (1=male, 2=female) to sequoia default (1=female, 2=male); any other numbers are set to NA
FIDsep	characters inbetween FID and IID in composite-ID. By default a double underscore is used, to avoid problems when some IIDs contain an underscore. Only used when UseFID=TRUE.
LifeHistData	dataframe with additional sex and birth year info. In case of conflicts, LifeHistData takes priority, with a warning. If UseFID=TRUE, IDs in LifeHistData are assumed to be already as FID__IID.

Details

The first 6 columns of PLINK .fam, .ped and .raw files are by default FID - IID - father ID (ignored) - mother ID (ignored) - sex - phenotype.

When additionally a

Value

a dataframe with id, sex and birth year, which can be used as input for sequoia

See Also

GenoConvert, PedStripFID to reverse UseFID

Examples

```
## Not run:
# combine FID and IID in dataframe with additional sex & birth years
ExtraLH$FID_IID <- paste(ExtraLH$FID, ExtraLH$IID, sep = "__")
LH.new <- LHConvert(PlinkFile, UseFID = TRUE, FIDsep = "__",
                    LifeHistData = ExtraLH)

## End(Not run)
```

LH_HSg5 *Example life history file*

Description

This is the lifehistory file associated with Ped_HSg5, which is Pedigree II in the paper.

Usage

```
data(LH_HSg5)
```

Format

A data frame with 1000 rows and 3 variables: ID, Sex (1=female, 2=male), and BirthYear

Author(s)

Jisca Huisman, <jisca.huisman@gmail.com>

References

Huisman, J. (2017) Pedigree reconstruction from SNP data: Parentage assignment, sibship clustering, and beyond. Molecular Ecology Resources 17:1009–1024.

See Also

Ped_HSg5 sequoia

MakeAgePrior *Age priors*

Description

Calculate age-difference based prior probability ratios $P(A|R)/P(A)$ for various categories of pairwise relatives.

Usage

```
MakeAgePrior(Ped = NULL, LifeHistData = NULL, MaxAgeParent = NULL,
  Flatten = TRUE, lambdaNW = -log(0.5)/100, Smooth = TRUE,
  Plot = TRUE, Return = "LR", quiet = FALSE)
```

Arguments

Ped	dataframe with pedigree, with id - dam - sire in columns 1-3, and optional column with birth years. Other columns are ignored.
LifeHistData	dataframe with columns id - sex (not used) - birth year (unknown: negative number or NA). Column names are ignored, so the column order is important. "Birth year" may be in any arbitrary discrete time unit relevant to the species (day, month, decade), as long as parents are never born in the same time unit as their offspring.
MaxAgeParent	maximum age of a parent (max across dams and sires). If NULL, will be estimated from the data. If there are fewer than 100 parents of either sex assigned, MaxAgeParent is set to the maximum age difference in the birth year column of Ped or LifeHistData.

`Flatten`	Calculate weighed average between the observed age difference distribution among the relative pairs with known age difference and a completely flat distribution. The weights are a function of the number of pairs and `lambdaNW`, see `Details`. Advisable if the relative pairs with known age difference non-typical of the pedigree as a whole or when their number is limited, and therefore automatically set to TRUE when there are fewer than 20 parents of either sex assigned. Not advisable when generations do not overlap.
`lambdaNW`	When `Flatten=TRUE`, weighing factors of the data-estimated age-difference distribution versus a flat distribution are calculated as $W(R) = 1-exp(-lambdaNW* N(R))$, where $N(R)$ is the number of pairs with relationship R for which the age difference is known. See Details below.
`Smooth`	Smooth the tails of and any dips in the distribution? Sets dips (<10% of average of neighbouring ages) to the average of the neigbouring ages, sets the age after the end (oldest observed age) to LR(end)/2, and assigns a small value (0.001) to the ages before the front (youngest observed age) and after the new end. Set to FALSE when generations do not overlap.
`Plot`	plot a 2-panel overview of the results?
`Return`	return only a matrix with the likelihood-ratio $P(A\|R)/P(A)$ ("LR") or a list including also various intermediate statistics ("all") ?
`quiet`	suppress messages

Details

Using Bayes' theorem,

$$P(Relationship|Agedifference) = P(Agedifference|Relationship)/P(Agedifference)*P(Relationship)$$

or for short $P(R|A) = P(A|R)/P(A)*P(R)$. During pedigree reconstruction, the ratios $P(A|R)/P(A)$ calculated here are multiplied by the age-independent genetic-only $P(Relationship)$ to obtain a probability that the pair are relatives of type R conditional on both their age difference and their genotypes. This age-difference prior is used not only for pairs of genotyped individuals, but also between genotyped and dummy individuals and between pairs of dummy individuals. Therefore, $P(Agedifference)$ is taken in the broadest sense possible and calculated across all pairs of individuals in Ped and LifeHistData. When P(A) is 0 but The resulting distribution is by default flattened and smoothed to account for finite sample size.

Sometimes no age information is available for one or more relative categories, for example when all candidate fathers are unsampled and/or of unknown age. To account for the amount of information available, if Flatten=TRUE a weighed average of $P(A|R)/P(A)$ as estimated from the data and a completely flat distribution is returned. The weighing factor $W(R)$ for each relationship is a function of the number of relative pairs with a known age difference $N(R)$, following the sigmoid curve $W(R) = 1 - exp(-lambdaNW * N(R))$. By default, $lambdaNW = -log(0.5)/100$, corresponding to a weight of <50% if there are <100 pairs, and >50% if there are >100 pairs. See example below for a plot of this curve.

Value

A matrix with the probability ratio of the age difference between two individuals conditional on them being a certain type of relative ($P(A|R)$) versus being a random draw from the sample ($P(A)$). For siblings and avuncular pairs, this is the absolute age difference.

The matrix has one row per age difference (0 - nAgeClasses) and nine columns, one for each relationship type, with abbreviations:

M	Mothers
P	Fathers
FS	Full siblings
MS	Maternal half-siblings
PS	Paternal half-siblings
MGM	Maternal grandmother
PGF	Paternal grandfather
MGF	Maternal grandfathers and paternal grandmothers
UA	Avuncular (aunt/uncle - niece/nephew)

When Return='all', a list is returned with in addition to this matrix ('LR.RU.A') the following elements:

BirthYearRange	vector length 2
MaxAgeParent	single number, as estimated from the data or provided
tblA.R	matrix with the counts per age difference (0 - nAgeClasses) and the five relationship types as for 'LR.RU.A', plus a column 'X' with age differences across all pairs of individuals, including those in LifeHistData but not in Ped.
Weights	vector length 4, the weights used to flatten the distributions
LR.RU.A.unweighed	matrix with nAgeClasses+1 rows and 9 columns; LR.RU.A prior to flattening and smoothing

Discrete generations

When generations do not overlap, Flatten and Smooth should both be set to FALSE. This is done automatically when it is detected that all parents are 1 year older than their offspring, and siblings are always born in the same year.

Single cohort

When no birth year information is given, or all individuals have the same birthyear, it is assumed that a single cohort has been analysed and a simple 2 by 9 matrix with 0's and 1's is returned. Note that by default it is assumed that no avuncular pairs exist within this single cohort; if they may exist, change LR.RU.A["0","UA"] from 0 to 1.

Other time units

"Birth year" may be in any arbitrary time unit relevant to the species (day, month, decade), as long as parents are never born in the same time unit as their offspring. Time of birth/hatching/germination must be expressed as whole, positive numbers.

See Also

sequoia, PlotAgePrior

Examples

```
data(LH_HSg5, Ped_HSg5, package="sequoia")
## Not run:
MakeAgePrior(Ped_HSg5, LH_HSg5, Flatten=FALSE, Smooth=FALSE)
APlist <- MakeAgePrior(Ped_HSg5, LH_HSg5, Flatten=FALSE, Smooth=TRUE,
  Return="all")
## End(Not run)

# effect of lambdaNW on weights when Flatten=TRUE:
lambda1 <- -log(0.5)/100
lambda2 <- -log(1-.9)/100
curve(1-exp(-lambda1*x), lty=2, lwd=2, from=1, to=1000, log="x",
      xlab="N known age difference", ylab="W", las=1)
curve(1-exp(-lambda2*x), lty=2, lwd=2, col="blue", add=TRUE)
abline(h=c(0,.1,.25,.5,.75,.9,1), col="grey", lty=3)

N = c(1,10,42,100,200, 432, 1e3, 1e4)
data.frame(N = N,
           W1 = round(1-exp(-lambda1*N), 3),
           W2 = round(1-exp(-lambda2*N), 3))
```

MkGenoErrors *Simulate genotyping errors*

Description

Generate errors and missing values in a (simulated) genotype matrix

Usage

```
MkGenoErrors(SGeno, CallRate = 0.99, SnpError = 5e-04,
  ErrorFM = function(E) {     matrix(c(1 - E - (E/2)^2, E, (E/2)^2, E/2,
  1 - E, E/2, (E/2)^2, E, 1 - E - (E/2)^2), 3, 3, byrow = TRUE) },
  Error.shape = 0.5, CallRate.shape = 1)
```

Arguments

SGeno	Matrix with genotype data in Sequoia's format: 1 row per individual, 1 column per SNP, and genotypes coded as 0/1/2.
CallRate	Either a single number for the mean call rate (genotyping success), OR a vector with the call rate at each SNP, OR a named vector with the call rate for each individual. In the third case, ParMis is ignored, and individuals in the pedigree (as id or parent) not included in this vector are presumed non-genotyped.
SnpError	mean per-locus genotyping error rate across SNPs, and a beta-distribution will be used to simulate the number of missing cases per SNP, OR a vector with the genotyping error for each SNP.

ErrorFM	function taking the error rate (scalar) as argument and returning a 4x4 or 3x3 matrix with probabilities that actual genotype i (rows) is observed as genotype j (columns).
Error.shape	first shape parameter (alpha) of beta-distribution of per-SNP error rates. A higher value results in a flatter distribution.
CallRate.shape	as Error.shape, for per-SNP call rates.

Value

The input genotype matrix, with some genotypes replaced, and some set to missing (-9)

Examples

```
data(Ped_HSg5)
GenoM <- SimGeno(Ped = Ped_HSg5, nSnp = 100, ParMis = 0.2,
                 SnpError=0, CallRate=1)
GenoM.actual <- GenoM
LowQ <- sample.int(nrow(GenoM), 42)  # low-quality samples
GenoM[LowQ, ] <- MkGenoErrors(GenoM[LowQ, ], SnpError = 0.05)
GenoM[-LowQ, ] <- MkGenoErrors(GenoM[-LowQ, ], SnpError = 0.001)
ErrorCount <- sapply(1:nrow(GenoM), function(i) {
  sum(GenoM.actual[i,] != GenoM[i,] & GenoM[i,] != -9) } )
mean(ErrorCount[LowQ])
mean(ErrorCount[-LowQ])
```

PedCompare *Compare two Pedigrees*

Description

Compare an inferred pedigree (Ped2) to a previous or simulated pedigree (Ped1), including comparison of sibship clusters and sibship grandparents.

Usage

```
PedCompare(Ped1 = NULL, Ped2 = NULL, na1 = c(NA, "0"),
  DumPrefix = c("F0", "M0"), SNPd = NULL)
```

Arguments

Ped1	original pedigree, dataframe with columns id-dam-sire; only the first 3 columns will be used.
Ped2	infered pedigree, e.g. SeqOUT$Pedigree, with columns id-dam-sire.
na1	the value for missing parents in Ped1 (assumed NA in Ped2).
DumPrefix	character vector of length 2 with the dummy prefices in Pedigree 2; all IDs not starting with the Dummy prefix are taken as genotyped.
SNPd	character vector with IDs of genotyped individuals.

Details

The comparison is divided into different classes of 'assignable' parents. This includes cases where the focal individual and parent according to Ped1 are both Genotyped (G-G), as well as cases where the non-genotyped parent according to Ped1 can be lined up with a sibship Dummy parent in Ped2 (G-D), or where the non-genotyped focal individual in Ped1 can be matched to a dummy individual in Ped2 (D-G and D-D). If SNPd is NULL (the default), and DumPrefix is set to NULL, the intersect between the IDs in Pedigrees 1 and 2 is taken as the vector of genotyped individuals.

Value

A list with

`Counts`	A 7 x 5 x 2 named numeric array with the number of matches and mismatches
`MergedPed`	A side-by-side comparison of the two pedigrees
`ConsensusPed`	A consensus pedigree, with Pedigree 2 taking priority over Pedigree 1
`DummyMatch`	Dataframe with all dummy IDs in Pedigree 2 (id), and the best-matching individual in Pedigree 1 (id.r)
`Mismatch`	A subset of MergedPed with mismatches between Ped1 and Ped2, as defined below. The two additional columns are Cat (category, 'GG', 'GD', 'DG' or 'DD', as described below) and Parent ('dam' or 'sire' indicating which is mismatching)
`Ped1only`	as Mismatches, with parents in Ped1 that were not assigned in Ped2
`Ped2only`	as Mismatches, with parents in Ped2 that were missing in Ped1

The first dimension of `Counts` denotes the following categories:

`GG`	Genotyped individual, assigned a genotyped parent in either pedigree
`GD`	Genotyped individual, assigned a dummy parent, or at least 1 genotyped sibling or a genotyped grandparent in Pedigree 1)
`GT`	Genotyped individual, total
`DG`	Dummy individual, assigned a genotyped parent (i.e., grandparent of the sibship in Pedigree 2)
`DD`	Dummy individual, assigned a dummy parent (i.e., avuncular relationship between sibships in Pedigree 2)
`DT`	Dummy total
`TT`	Total total, includes all genotyped individuals, plus non-genotyped individuals in Pedigree 1, plus non-replaced dummy individuals (see below) in Pedigree 2

The dummy individual count includes all non-genotyped individuals in Pedigree 1 who have, according to either pedigree, at least 2 genotyped offspring, or at least one genotyped offspring and a genotyped parent.

The second dimension of `Counts` gives the outcomes:

`Total`	The total number of individuals with a parent assigned in either or both pedigrees
`Match`	The same parent is assigned in both pedigrees (non-missing). For dummy parents, it is considered a match if the inferred sibship which contains the most offspring of a non-genotyped parent, consists for more than half of this individual's offspring.

Mismatch	Different parents assigned in the two pedigrees. When a sibship according to Pedigree 1 is split over two sibships in Pedigree 2, the smaller fraction is included in the count here.
P1only	Parent in Pedigree 1 but not 2; includes non-assignable parents (e.g. not genotyped and no genotyped offspring).
P2only	Parent in Pedigree 2 but not 1.

The third dimension Counts separates between maternal and paternal assignments, where e.g. paternal 'DR' is the assignment of fathers to both maternal and paternal sibships.

'MergedPed' provides the following columns:

id	All ids in both Pedigree 1 and 2
dam.1, sire.1	parents in Pedigree 1
dam.2, sire.2	parents in Pedigree 2
id.r, dam.r, sire.r	when in Pedigree 2 a dummy parent is assigned, this column gives the best-matching non-genotyped individual according to Pedigree 1, or "nomatch". If a sibship in Pedigree 1 is divided over 2 sibships in Pedigree 2, the smaller one will be denoted as "nomatch"

In 'ConsensusPed', the priority used is parent.r (if not "nomatch) > parent.2 > parent.1. The columns 'dam.cat' and 'sire.cat' give a 2-letter code denoting whether the focal individual (first letter) and its assigned parent (2nd letter) are

G	Genotyped
D	Dummy individual (in Pedigree 2)
R	Dummy individual in pedigree 2 replaced by best matching non-genotyped individual in pedigree 1
U	Ungenotyped (in Pedigree 1, with no dummy match)
X	No parent in either pedigree

Author(s)

Jisca Huisman, <jisca.huisman@gmail.com>

See Also

DyadCompare, sequoia, EstConf.

Examples

```
## Not run:
data(Ped_HSg5, SimGeno_example, LH_HSg5, package="sequoia")
SeqOUT <- sequoia(GenoM = SimGeno_example, LifeHistData = LH_HSg5)
compare <- PedCompare(Ped1=Ped_HSg5, Ped2=SeqOUT$Pedigree)
compare$Counts   # 2 mismatches, due to simulated genotyping errors
head(compare$MergedPed)
```

```
PedM <- compare$MergedPed
# find mismatching mothers:
with(PedM, PedM[which(dam.1!=dam.2 & dam.1!=dam.r),])

# find mothers in Ped1 which are genotyped but not assigned in Ped2:
with(PedM, PedM[which(is.na(dam.2) & !is.na(dam.1) &
                      !is.na(id) & dam.1 %in% id),])

## End(Not run)
```

`PedStripFID` *backtransform IDs*

Description

Reverse the joining of FID and IID in `GenoConvert` and `LHConvert`

Usage

```
PedStripFID(Ped, FIDsep = "__")
```

Arguments

`Ped`	Pedigree as returned by sequoia (e.g. `SeqOUT$Pedigree`)
`FIDsep`	characters inbetween FID and IID in composite-ID

Details

Note that the family IDs are the ones provided, and not automatically updated. New, numeric ones can be obtained with `FindFamilies`

Value

a pedigree with 6 columns

`FID`	family ID of focal individual (offspring).
`id`	within-family of focal individual
`dam.FID`	original family ID of assigned dam
`dam`	within-family of dam
`sire.FID`	original family ID of assigned sire
`sire`	within-family of sire

Ped_HSg5 *Example pedigree*

Description

This is Pedigree II in the paper.

Usage

```
data(Ped_HSg5)
```

Format

A data frame with 1000 rows and 3 variables (id, dam, sire)

Author(s)

Jisca Huisman, <jisca.huisman@gmail.com>

References

Huisman, J. (2017) Pedigree reconstruction from SNP data: Parentage assignment, sibship clustering, and beyond. Molecular Ecology Resources 17:1009–1024.

See Also

LH_HSg5 SimGeno_example sequoia

PlotAgePrior *Plot age priors*

Description

visualise the age-difference based prior probability ratios

Usage

```
PlotAgePrior(AP = NULL, GPlines = TRUE)
```

Arguments

AP	matrix with age priors of dimension nAgeclasses rows by 9 columns, OR a list returned by MakeAgePrior when called with Return='all', OR a list returned by sequoia containing an element 'AgePriors'.
GPlines	show the lines for grandparental & avuncular relationships? Can sometimes make the plot overly crowded.

Details

When called with an output list from `MakeAgePrior`, the raw distributions are shown too in a separate panel, in addition to the the scaled, flattened & smoothened distributions used for pedigree reconstruction.

See Also

`MakeAgePrior`, `SummarySeq`

`sequoia` *Pedigree Reconstruction*

Description

Perform pedigree reconstruction based on SNP data, including parentage assignment and sibship clustering.

Usage

```
sequoia(GenoM = NULL, LifeHistData = NULL, SeqList = NULL,
  MaxSibIter = 10, Err = 1e-04, MaxMismatch = 3, Tfilter = -2,
  Tassign = 0.5, MaxSibshipSize = 100, DummyPrefix = c("F", "M"),
  Complex = "full", UseAge = "yes", args.AP = list(Flatten = TRUE,
  Smooth = TRUE), FindMaybeRel = FALSE, CalcLLR = TRUE,
  quiet = FALSE)
```

Arguments

`GenoM`	numeric matrix with genotype data: One row per individual, and one column per SNP, coded as 0, 1, 2 or -9 (missing). Use `GenoConvert` to convert genotype files created in PLINK using –recodeA or in Colony's 2-column format to this format.
`LifeHistData`	Dataframe with 3 columns: • ID: max. 30 characters long, • Sex: 1 = females, 2 = males, other = unkown, except 4 = hermaphrodite, • BirthYear: (birth or hatching year) Integer, negative numbers are interpreted as missing values. If the species has multiple generations per year, use an integer coding such that the candidate parents' 'Birth year' is at least one smaller than their putative offspring's. Column names are ignored, so ensure column order is ID - sex - birth year.
`SeqList`	list with output from a previous run, containing the elements 'Specs', 'AgePriors' and/or 'PedigreePar', as described below, to be used in the current run. If `SeqList$Specs` is provided, all other input parameter values except `MaxSibIter` are ignored.

MaxSibIter	number of iterations of sibship clustering, including assignment of grandparents to sibships and avuncular relationships between sibships. Set to 0 to not (yet) perform this step, which is by far the most time consuming and may take several hours for large datasets. Clustering continues until convergence or until MaxSibIter is reached.
Err	estimated genotyping error rate. The error model aims to deal with scoring errors typical for SNP arrays.
MaxMismatch	maximum number of loci at which candidate parent and offspring are allowed to be opposite homozygotes. Setting a more liberal threshold can improve performance if the error rate is high, at the cost of decreased speed.
Tfilter	threshold log10-likelihood ratio (LLR) between a proposed relationship versus unrelated, to select candidate relatives. Typically a negative value, related to the fact that unconditional likelihoods are calculated during the filtering steps. More negative values may decrease non-assignment, but will increase computational time.
Tassign	minimum LLR required for acceptance of proposed relationship, relative to next most likely relationship. Higher values result in more conservative assignments. Must be zero or positive.
MaxSibshipSize	maximum number of offspring for a single individual (a generous safety margin is advised).
DummyPrefix	character vector of length 2 with prefixes for dummy dams (mothers) and sires (fathers); maximum 20 characters each.
Complex	either "full" (default), "simp" (simplified, no explicit consideration of inbred relationships), "mono" (monogamous) or "herm" (hermaphrodites, otherwise like "full").
UseAge	either "yes" (default), "no", or "extra" (additional rounds with extra reliance on ageprior, may boost assignments but increased risk of erroneous assignments); used during full reconstruction only.
args.AP	list with arguments to be passed on to MakeAgePrior.
FindMaybeRel	identify pairs of non-assigned likely relatives after pedigree reconstruction. Can be time-consuming in large datasets. NOTE: from v1.2 default changed from TRUE to FALSE; GetMaybeRel can now be called separately.
CalcLLR	calculate log-likelihood ratios for all assigned parents (is parent vs. is otherwise related). Time-consuming in large datasets.
quiet	suppress messages: TRUE/FALSE/"verbose".

Details

For each pair of candidate relatives, the likelihoods are calculated of them being parent-offspring (PO), full siblings (FS), half siblings (HS), grandparent-grandoffspring (GG), full avuncular (niece/nephew - aunt/uncle; FA), half avuncular/great-grandparental/cousins (HA), or unrelated (U). Assignments are made if the likelihood ratio (LLR) between the focal relationship and the most likely alternative exceed the threshold Tassign.

Further explanation of the various options and interpretation of the output is provided in the vignette.

Value

A list with some or all of the following components:

AgePriors	Matrix with age-difference based prior probability ratios, used for full pedigree reconstruction. See MakeAgePrior for details.
DummyIDs	Dataframe with pedigree for dummy individuals, as well as their sex, estimated birth year (point estimate, upper and lower bound of 95% confidence interval), number of offspring, and offspring IDs (genotyped offspring only).
DupGenotype	Dataframe, duplicated genotypes (with different IDs, duplicate IDs are not allowed). The specified number of maximum mismatches is used here too. Note that this dataframe may include pairs of closely related individuals, and monozygotic twins.
DupLifeHistID	Dataframe, rownumbers of duplicated IDs in life history dataframe. For convenience only, but may signal a problem. The first entry is used.
ExcludedInd	Individuals in GenoM which were excluded because of a too low genotyping success rate (<50%).
ExcludedSNPs	Column numbers of SNPs in GenoM which were excluded because of a too low genotyping success rate (<10%).
LifeHist	Provided dataframe with sex and birth year data.
MaybeParent	Dataframe with pairs of individuals who are more likely parent-offspring than unrelated, but which could not be phased due to unknown age difference or sex, or for whom LLR did not pass Tassign.
MaybeRel	Dataframe with pairs of individuals who are more likely to be first or second degree relatives than unrelated, but which could not be assigned.
MaybeTrio	Dataframe with non-assigned parent-parent-offspring trios (both parents are of unknown sex), with similar columns as the pedigree
NoLH	Vector, IDs in genotype data for which no life history data is provided.
Pedigree	Dataframe with assigned genotyped and dummy parents from Sibship step; entries for dummy individuals are added at the bottom.
PedigreePar	Dataframe with assigned parents from Parentage step.
Specs	Named vector with parameter values.
TotLikParents	Numeric vector, Total likelihood of the genotype data at initiation and after each iteration during Parentage.
TotLikSib	Numeric vector, Total likelihood of the genotype data at initiation and after each iteration during Sibship clustering.

List elements PedigreePar and Pedigree both have the following columns:

id	Individual ID
dam	Assigned mother, or NA
sire	Assigned father, or NA
LLRdam	Log10-Likelihood Ratio (LLR) of this female being the mother, versus the next most likely relationship between the focal individual and this female (see Details for relationships considered)

LLRsire	idem, for male parent
LLRpair	LLR for the parental pair, versus the next most likely configuration between the three individuals (with one or neither parent assigned)
OHdam	Number of loci at which the offspring and mother are opposite homozygotes
OHsire	idem, for father
MEpair	Number of Mendelian errors between the offspring and the parent pair, includes OH as well as e.g. parents being opposing homzygotes, but the offspring not being a heterozygote. The offspring being OH with both parents is counted as 2 errors.

Disclaimer

While every effort has been made to ensure that sequoia provides what it claims to do, there is absolutely no guarantee that the results provided are correct. Use of sequoia is entirely at your own risk.

Author(s)

Jisca Huisman, <jisca.huisman@gmail.com>

References

Huisman, J. (2017) Pedigree reconstruction from SNP data: Parentage assignment, sibship clustering, and beyond. Molecular Ecology Resources 17:1009–1024.

See Also

GenoConvert, SnpStats, GetMaybeRel, EstConf, SummarySeq, writeSeq, vignette("sequoia")

Examples

```
# == EXAMPLE 1 ==
data(SimGeno_example, LH_HSg5, package="sequoia")
head(SimGeno_example[,1:10])
head(LH_HSg5)
SeqOUT <- sequoia(GenoM = SimGeno_example,
                  LifeHistData = LH_HSg5, MaxSibIter = 0)
names(SeqOUT)
SeqOUT$PedigreePar[34:42, ]

## Not run:
SeqOUT2 <- sequoia(GenoM = SimGeno_example,
                  LifeHistData = LH_HSg5, MaxSibIter = 10)
SeqOUT2$Pedigree[34:42, ]

# == EXAMPLE 2 ==
# ideally, select 400-700 SNPs: high MAF & low LD
# save in 0/1/2/NA format (PLINK's --recodeA)
GenoM <- GenoConvert(InFile = "inputfile_for_sequoia.raw")
SNPSTATS <- SnpStats(GenoM)
```

```
# perhaps after some data-cleaning:
write.table(GenoM, file="MyGenoData.txt", row.names=T, col.names=F)
# later:
GenoM <- as.matrix(read.table("MyGenoData.txt", row.names=1, header=F))
LHdata <- read.table("LifeHistoryData.txt", header=T) # ID-Sex-birthyear
SeqOUT <- sequoia(GenoM, LHdata, Err=0.005, MaxMismatch=10)
SummarySeq(SeqOUT)
writeSeq(SeqOUT, OutFormat = "xls")  # needs library xlsx

## End(Not run)
```

SimGeno *Simulated genotypes*

Description

Simulate SNP genotype data from a pedigree, with optional missingess and errors.

Usage

```
SimGeno(Ped, nSnp = 400, ParMis = 0.4, MAF = 0.3, CallRate = 0.99,
  SnpError = 5e-04, ErrorFM = "SNPchip", ReturnStats = FALSE,
  OutFile = NA, Inherit = "autosomal", InheritFile = NA,
  quiet = FALSE, PropLQ, MisHQ, MisLQ, ErHQ, ErLQ)
```

Arguments

Ped	Dataframe, pedigree with the first three columns being id - dam - sire. Column names are ignored, as are additional columns, with the exception of a 'Sex' column when Inherit is not 'autosomal'.
nSnp	number of SNPs to simulate.
ParMis	Single number or vector length two with proportion of parents with fully missing genotype. Ignored if CallRate is a named vector.
MAF	minimum minor allele frequency, and allele frequencyes will be sampled uniformly between this minimum and 0.5, OR a vector with minor allele frequency at each locus. In both cases, this is the MAF among pedigree founders, the MAF in the sample will deviate due to drift.
CallRate	Either a single number for the mean call rate (genotyping success), OR a vector with the call rate at each SNP, OR a named vector with the call rate for each individual. In the third case, ParMis is ignored, and individuals in the pedigree (as id or parent) not included in this vector are presumed non-genotyped.
SnpError	mean per-locus genotyping error rate across SNPs, and a beta-distribution will be used to simulate the number of missing cases per SNP, OR a vector with the genotyping error for each SNP.
ErrorFM	function taking the error rate (scalar) as argument and returning a 3x3 matrix with probabilities that actual genotype i (rows) is observed as genotype j (columns). See details.

ReturnStats	in addition to the genotype matrix, return the input parameters and mean & quantiles of MAF, error rate and call rates.
OutFile	filename for simulated genotypes. If NA (default), return results within R.
Inherit	inheritance pattern, scalar or vector of length nSnp, Defaults to 'autosomal'. An excel file included in the package has inheritance patterns for the X and Y chromosome and mtDNA, and allows custom inheritance patterns. Note that these are NOT currently supported by the pedigree reconstruction with sequoia !
InheritFile	filename for excel file with inheritance patterns, requires library xlsx.
quiet	suppress messages.
PropLQ	[deprecated] proportion of low-quality samples.
MisHQ	[deprecated] average missingness for high-quality samples, assuming a beta-disstribution with alpha = 1.
MisLQ	[deprecated] average missingness in low-quality samples.
ErHQ	[deprecated] error rate in high quality samples (defaults to 0.005).
ErLQ	[deprecated] error rate in low quality samples.

Details

Please ensure the pedigree is a valid pedigree, for example by first running fixPedigree() from library Pedantics. Genotypes are drawn assuming Hardy-Weinberg equilibrium and the provided MAF among founders, i.e. individuals with no known parents, including parents who do not also occur in the ID column. Offspring genotypes are generated following Mendelian inheritance, assuming all loci are completely independent.

Genotyping errors are generated following a user-definable 3x3 matrix with probabilities that actual genotype i (rows) is observed as genotype j (columns). This is specified as ErrorFM, which is a function of SnpError. By default (ErrorFM = "SNPchip"), SnpError is interpreted as a locus-level error rate (rather than allele-level), and equals the probability that a homozygote is observed as heterozygote, and the probability that a heterozygote is observed as either homozygote (i.e., the probability that it is observed as AA = probability that observed as aa = SnpError/2). The probability that one homozygote is observed as the other is $(\texttt{SnpError}/2)^2$.

Note that this differs from versions up to 1.1.1, where a proportion of SnpError*3/2 of genotypes were replaced with random genotypes. This corresponds to ErrorFM = "Version111".

Error rates differ between SNPs, but the same error pattern is used across all SNPs, even when inheritance patterns vary. When two or more different error patterns are required, SimGeno should be run on the different SNP subsets separately, and results combined.

Variation in call rates is assumed to follow a highly skewed (beta) distribution, with many samples having call rates close to 1, and a narrowing tail of lower call rates. The first shape parameter defaults to 1 (but see MkGenoErrors), and the second shape parameter is defined via the mean as CallRate. For 99.9 rate of 0.8 (0.9; 0.95) or higher, use a mean call rate of 0.969 (0.985; 0.993).

Variation in call rate between samples can be specified by providing a named vector to CallRate, which supersedes PropLQ in versions up to 1.1.1. Otherwise, variation in call rate and error rate between samples occurs only as side-effect of the random nature of which individuals are hit by per-SNP errors and drop-outs. Finer control is possible by first generating an error-free genotype matrix, and then calling MkGenoErrors directly on subsets of the matrix.

Value

if ReturnStats=FALSE (the default), a matrix with genotype data in sequoia's input format, encoded as 0/1/2/-9. if ReturnStats=TRUE, a named list with list 'ParamsIN', list 'StatsOUT', and matrix 'SGeno'.

Disclaimer

This simulation is highly simplistic and assumes that all SNPs segregate completely independently, and that the SNPs are in Hardy-Weinberg equilibrium in the pedigree founders. Results based on this simulated data will provide an minimum estimate of the number of SNPs required, and an optimistic estimate of pedigree reconstruction performance.

Author(s)

Jisca Huisman, <jisca.huisman@gmail.com>

See Also

EstConf, MkGenoErrors

Examples

```
data(Ped_HSg5)
GenoM <- SimGeno(Ped = Ped_HSg5, nSnp = 100, ParMis = c(0.2, 0.7))

## Not run:
# Alternative genotyping error model
EFM <- function(E) {   # Whalen, Gorjanc & Hickey 2018
 matrix(c(1-E*3/4, E/4, E/4,
          E/4, 1/2-E/4, 1/2-E/4, E/4,
          E/4, E/4, 1-E*3/4),
          3,3, byrow=TRUE)  }
EFM(0.01)
GenoM <- SimGeno(Ped = Ped_HSg5, nSnp = 100, ParMis = 0.2,
 SnpError = 5e-3, ErrorFM = EFM)

## End(Not run)
```

SimGeno_example *Example genotype file*

Description

Simulated genotype data for cohorts 1+2 in Pedigree Ped_HSg5

Usage

```
data(SimGeno_example)
```

Format

A data frame with 214 rows and 201 columns: id, followed by 1 column per SNP coded as 0/1/2 or -9 for missing values.

Author(s)

Jisca Huisman, <jisca.huisman@gmail.com>

See Also

Ped_HSg5, SimGeno

SnpStats *SNP summary statistics*

Description

Estimate allele frequency (AF), missingness and Mendelian errors per SNP.

Usage

```
SnpStats(GenoM, Ped = NULL, Plot = TRUE)
```

Arguments

GenoM	Genotype matrix, in sequoia's format: 1 column per SNP, 1 row per individual, genotypes coded as 0/1/2/-9, and rownames giving individual IDs.
Ped	a dataframe with 3 columns: ID - parent1 - parent2. Additional columns and non-genotyped individuals are ignored. Only used to estimate the error rate.
Plot	show histograms of the results?

Details

Calculation of these summary statistics can be done in PLINK, and SNPs with low minor allele freuqency or high missigness should be filtered out using PLINK prior to pedigree reconstruction. This function is merely provided as an aid to inspect the relationship between AF, missingness and error to find a suitable combination of thresholds to use.

The error count includes both the number of parent-offspring pairs that are opposing homozygotes (parent is AA and offspring is aa), as Mendelian errors in parent-parent-offspring trios (e.g. parents AA and aa, but offspring not Aa).

The underlying genotyping error can not be easily estimated from the number of Mendelian errors, as many errors may go undetected and a single error in a prolific individual can result in a high number of Mendelian errors. Moreover, a high error rate may interfere with pedigree reconstruction, and succesful assignment will be biased towards parents with lower error count.

Value

a matrix with a number of rows equal to the number of SNPs (=number of columns of GenoM) and 2 or 3 columns:

AF	Allele frequency of the 'second allele' (the one for which the homozygote is coded 2)
Mis	Proportion of missing calls
ER	(only when Ped provided) number of Mendelian errors in parent- offspring pairs (i.e. the number of opposing homozygotes, 'OHdam' & 'OHsire' in pedigree) and parent-parent-offspring trios ('MEpairs' in pedigree).

See Also

GenoConvert

SummarySeq *Summarise sequoia output or pedigree*

Description

Number of assigned parents and grandparents and sibship sizes, split by genotyped, dummy, and 'observed'.

Usage

```
SummarySeq(SeqList = NULL, Ped = NULL, DumPrefix = c("F0", "M0"),
  SNPd = NULL, Plot = TRUE)
```

Arguments

SeqList	the list returned by sequoia. Only elements 'Pedigree' or 'PedigreePar' and 'AgePriors' are used.
Ped	Dataframe, pedigree with the first three columns being id - dam - sire. Column names are ignored, as are additional columns.
DumPrefix	character vector of length 2 with prefixes for dummy dams (mothers) and sires (fathers). Will be read from SeqList's 'Specs' if provided. Used to distinguish between dummies and non-dummies.
SNPd	character vector with ids of SNP genotyped individuals. Only when Ped is provided instead of SeqList, then used to distinguish between genetically assigned parents and 'observed' parents (e.g. observed in the field, or assigned previously using microsatellites). Will be read from SeqList's 'PedigreePar' if provided.
Plot	Show barplots and histograms of the results, as well as of the parental LLRs, Mendelian errors, and agepriors, if present.

Value

A list with the following elements:

ParentCount	a 2x3x2x4 array with the number of assigned parents, split by D1: genotyped vs dummy individuals; D2: female, male and unknown-sex individuals; D3: dams vs sires; D4: genotyped, dummy, observed vs no parent
GPCount	a 4x4 matrix with for all genotyped individuals the number of assigned grandparents, split by D1: Maternal grandmother, maternal grandfather, paternal grandmother, paternal grandfather; D2: genotyped, dummy, observed vs no grandparent
SibSize	a list with as first element a table of maternal sibship sizes, and as second element a table of paternal sibship sizes. Each table is a matrix with a number of rows equal to the maximum sibship size, and 3 columns, splitting by the type of parent: genotyped, dummy, or observed.

See Also

sequoia

Examples

```
## Not run:
data(SimGeno_example, LH_HSg5, package="sequoia")
SeqOUT <- sequoia(GenoM = SimGeno_example,
                  LifeHistData = LH_HSg5, MaxSibIter = 10)
Ped_example <- SeqOUT$Pedigree
Ped_example$dam[1:20] <- paste0("Mum", 1:20)  # some field mums
SummarySeq(SeqOUT, Ped=Ped_example)

## End(Not run)
```

---

writeColumns *write data to a file column-wise*

Description

write data.frame or matrix to a text file, using white space padding to keep columns aligned as in print

Usage

```
writeColumns(x, file = "", row.names = TRUE, col.names = TRUE)
```

Arguments

x	the object to be written, preferably a matrix or data frame. If not, it is attempted to coerce x to a matrix.
file	a character string naming a file.
row.names	a logical value indicating whether the row names of x are to be written along with x.
col.names	a logical value indicating whether the column names of x are to be written along with x

---

writeSeq *write sequoia output to excel or text files*

Description

The various list elements returned by sequoia are each written to text files in the specified folder, or to separate sheets in a single excel file (requires library **xlsx**).

Usage

```
writeSeq(SeqList, GenoM = NULL, PedComp = NULL, OutFormat = "txt",
  folder = "Sequoia-OUT", file = "Sequoia-OUT.xlsx", quiet = FALSE)
```

Arguments

SeqList	the list returned by sequoia, to be written out.
GenoM	the matrix with genetic data (optional). Ignored if OutFormat='xls', as the resulting file could become too large for excel.
PedComp	a list with results from PedCompare (optional). SeqList$DummyIDs is combined with PedComp$DummyMatch if both are provided.
OutFormat	'xls' or 'txt'.
folder	the directory where the text files will be written; will be created if it does not already exists. Relative to the current working directory, or NULL for current working directory. Ignored if OutFormat='xls'.
file	the name of the excel file to write to, ignored if OutFormat='txt'.
quiet	suppress messages.

Details

The text files can be used as input for the stand-alone Fortran version of #' sequoia, e.g. when the genotype data is too large for R. See vignette('sequoia') for further details.

Examples

```
## Not run:
writeSeq(SeqList, OutFormat="xls", file="MyFile.xlsx")

# add additional sheets to the excel file:
library(xlsx)
write.xlsx(MyData, file = "MyFile.xlsx", sheetName="ExtraData",
      col.names=TRUE, row.names=FALSE, append=TRUE, showNA=FALSE)

## End(Not run)
```

Index