Package 'sequoia'

May 18, 2020

Type Package

Title Pedigree Inference from SNPs

Version 2.0.7

Date 2020-05-17

Author Jisca Huisman [aut, cre]

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

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 7.1.0

Suggests xlsx, knitr, rmarkdown, bookdown

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-05-18 18:30:09 UTC

R topics documented:

`CalcMaxMismatch` *Maximum number of mismatches*

Description

Calculate the maximum expected number of mismatches for duplicate samples, parent-offspring pairs, and parent-parent-offspring trios.

Usage

```
CalcMaxMismatch(Err, MAF, ErrFlavour = "version2.0", qntl = 1 - 1e-05)
```

Arguments

`Err`	estimated genotyping error rate, as a single number or 3x3 matrix. If a matrix, this should be the probability of observed genotype (columns) conditional on actual genotype (rows). Each row must therefore sum to 1.
`MAF`	vector with minor allele frequency at each SNP.
`ErrFlavour`	function that takes `Err` as input, and returns a 3x3 matrix of observed (columns) conditional on actual (rows) genotypes, or choose from inbuilt ones as used in sequoia 'version2.0', 'version1.3', or 'version1.1'. Ignored if `Err` is a matrix. See `ErrToM`.

qntl — quantile of binomial distribution to be used as the maximum, of individual-level probability. For a desired dataset-level probability quantile Q, use $qntl = Q^(1/N)$, where N is the number of individuals.

Details

The thresholds for maximum number of mismatches calculated here aim to minimise false negatives, i.e. to minimise the chance that any true duplicates or true parent-offspring pairs are already excluded during the filtering steps where these MaxMismatch values are used. Consequently, there is a high probability of false positives, i.e. it is likely that some sample pairs with fewer mismatches than the MaxMismatch threshold, are in fact not duplicate samples or parent-offspring pairs. Use of these MaxMismatch thresholds is therefore only the first step of pedigree reconstruction by sequoia.

Value

a vector with three integers:

DUP — Maximum number of differences between 2 samples from the same individual

OH — Maximum number of Opposing Homozygous SNPs between a true parent-offspring pair

ME — Maximum number of Mendelian Errors among a true parent-parent- offspring trio

See Also

SnpStats

CalcOHLLR *calculate OH and LLR*

Description

Count opposite homozygous (OH) loci between parent-offspring pairs and Mendelian errors (ME) between parent-parent-offspring trios, and calculate the parental log-likelihood ratios (LLR).

Usage

```
CalcOHLLR(
  Pedigree = NULL,
  GenoM = NULL,
  CalcLLR = TRUE,
  LifeHistData = NULL,
  AgePrior = FALSE,
  Err = 1e-04,
  ErrFlavour = "version2.0",
  Tassign = 0.5,
  Complex = "full",
```

```
  GDX = TRUE,
  quiet = FALSE
)
```

Arguments

Pedigree	dataframe with columns id-dam-sire. May include non-genotyped individuals, which will be treated as dummy individuals.
GenoM	the genotype matrix
CalcLLR	calculate log-likelihood ratios for all assigned parents (genotyped + dummy/non-genotyped; parent vs. otherwise related). If FALSE, only number of mismatching SNPs are counted (OH & ME), and parameters LifeHistData, AgePrior, Err, Tassign, and Complex are **ignored**. Note also that calculating likelihood ratios is much more time consuming than counting OH & ME.
LifeHistData	Dataframe with columns ID - Sex - BirthYear, and optionally columns BY.min and BY.max. If provided, used to delimit possible alternative relationships.
AgePrior	logical (TRUE/FALSE) to estimate the ageprior from Pedigree and LifeHistData, or an agepriors matrix (see MakeAgePrior). Affects which alternative relationships are considered (only those where $P(A\|R)/P(A) > 0$). When TRUE, MakeAgePrior is called using its default values.
Err	estimated genotyping error rate, as a single number or 3x3 matrix. If a matrix, this should be the probability of observed genotype (columns) conditional on actual genotype (rows). Each row must therefore sum to 1.
ErrFlavour	function that takes Err as input, and returns a 3x3 matrix of observed (columns) conditional on actual (rows) genotypes, or choose from inbuilt ones as used in sequoia 'version2.0', 'version1.3', or 'version1.1'. Ignored if Err is a matrix. See ErrToM.
Tassign	used to determine whether or not to consider some more exotic relationships when Complex="full".
Complex	determines which relationships are considered as alternatives. Either "full" (default), "simp" (simplified, ignores inbred relationships), or "mono" (monogamous).
GDX	call getAssignCat to classify individuals as genotyped (G), substitutable by a dummy (D) or neither (X).
quiet	logical, suppress messages

Details

Any individuals in Pedigree that do not occur in GenoM are substituted by dummy individuals; a value of '0' in column 'SNPd.id.dam' in the output means that either the focal individual or the dam was thus substituted, or both were. Use getAssignCat to distinguish between these cases.

The birth years in LifeHistData and the AgePrior are not used in the calculation and do not affect the value of the likelihoods for the various relationships, but they _are_ used during some filtering steps, and may therefore affect the likelihood _ratio_. The default (AgePrior=FALSE) assumes all age-relationship combinations are possible, which may mean that some additional alternatives are considered compared to the sequoia default, resulting in somewhat lower LLR values.

A negative LLR for A's parent B indicates either that B is not truely the parent of A, or that B's parents are incorrect. The latter may cause B's presumed true, unobserved genotype to greatly divert from its observed genotype, with downstream consequences for its offspring. In rare cases it may also be due to 'weird', non-implemented double or triple relationships between A and B.

Value

the `Pedigree` dataframe with additional columns:

`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 homozygotes, but the offspring not being a heterozygote. The offspring being OH with both parents is counted as 2 errors.
`SNPd.id.dam`	Number of SNPs scored (non-missing) for both individual and dam
`SNPd.id.sire`	Number of SNPs scored for both individual and sire
`id.cat`	Character denoting whether the focal individual is genotyped (G), substitutable by a dummy (D), or neither (X).
`dam.cat`	as id.cat, for dams. If id.cat and/or dam.cat is 'X', the dam cannot be assigned.
`sire.cat`	as dam.cat, for sires
`Sexx`	Sex in LifeHistData, or inferred Sex when assigned as part of parent-pair
`BY.est`	mode of birth year probability distribution
`BY.lo`	lower limit of 95% highest density region of birth year probability distribution
`BY.hi`	higher limit

The columns 'LLRdam', 'LLRsire' and 'LLRpair' are only included when `CalcLLR=TRUE`. The columns 'dam.cat' and 'sire.cat' are only included when `GDX=TRUE`. The columns 'Sexx', 'BY.est', 'BY.lo' and 'BY.hi' are only included when `LifeHistData` is provided, and at least one genotyped individual has an unknown birthyear or unknown sex.

See Also

`SummarySeq` for visualisation of OH & LLR distributions; `GenoConvert` to read in various genotype data formats, `CheckGeno`; `PedPolish` to check and 'polish' the pedigree; `getAssignCat` to find which id-parent pairs are both genotyped or can be substituted by dummy individuals; `sequoia` for pedigree reconstruction

Examples

```
## Not run:
# have a quick look for errors in an existing pedigree,
# without running pedigree reconstruction
PedA <- CalcOHLLR(Pedigree = MyOldPedigree, GenoM = MyNewGenotypes,
  CalcLLR=FALSE)

# or run sequoia with CalcLLR=FALSE, and add OH + LLR later
SeqOUT <- sequoia(Genotypes, LifeHist, CalcLLR=FALSE)
PedA <- CalcOHLLR(Pedigree = SeqoUT$Pedigree[, 1:3], GenoM = Genotypes,
  LifeHistData = LIfeHist, AgePrior = TRUE, Complex = "full")

# visualise
SummarySeq(PedA, Panels=c("LLR", "OH"))

## End(Not run)
```

CheckGeno *check GenoM*

Description

Check that the provided genotype matrix is in the correct format, and check for low call rate samples and SNPs

Usage

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

Arguments

GenoM	the genotype matrix
quiet	suppress messages
Plot	display the plots of SnpStats

Value

a list with, if any are found:

ExcludedSNPs	SNPs scored for <10 excluded when running sequoia
ExcludedSnps-mono	monomorphic (fixed) SNPs; automatically excluded when running sequoia. Column numbers are *after* removal of ExcludedSNPs, if any.
ExcludedIndiv	Individuals scored for <5 reliably included during pedigree reconstruction. Individual call rate is calculated after removal of 'Excluded SNPs'
Snps-LowCallRate	SNPs scored for 10 recommended to be filtered out
Indiv-LowCallRate	individuals scored for <50 recommended to be filtered out

Thresholds

Appropriate call rate thresholds for SNPs and individuals depend on the total number of SNPs, distribution of call rates, genotyping errors, and the proportion of candidate parents that are SNPd (sibship clustering is more prone to false positives). Note that filtering first on SNP call rate tends to keep more individuals in.

See Also

SnpStats to calculate SNP call rates; CalcOHLLR to count the number of SNPs scored in both focal individual and parent

Examples

```
## Not run:
GenoM <- SimGeno(Ped_HSg5, nSnp=400, CallRate = runif(400, 0.2, 0.8))
Excl <- CheckGeno(GenoM)
GenoM.orig <- GenoM   # make a 'backup' copy
if ("ExcludedSnps" %in% names(Excl))
  GenoM <- GenoM[, -Excl[["ExcludedSnps"]]]
if ("ExcludedInd" %in% names(Excl))
  GenoM <- GenoM[!rownames(GenoM) %in% Excl[["ExcludedInd"]], ]
if ("ExcludedIndiv" %in% names(Excl))
  GenoM <- GenoM[!rownames(GenoM) %in% Excl[["ExcludedIndiv"]], ]

# warning about  SNPs scored for <50% of individuals ?
SnpCallRate <- apply(GenoM, MARGIN=2,
                     FUN = function(x) sum(x!=-9)) / nrow(GenoM)
hist(SnpCallRate, breaks=50, col="grey")
GenoM <- GenoM[, SnpCallRate > 0.6]

# to be on the safe side, filter out low call rate individuals
IndivCallRate <- apply(GenoM, MARGIN=1,
                       FUN = function(x) sum(x!=-9)) / ncol(GenoM)
hist(IndivCallRate, breaks=50, col="grey")
GoodSamples <- rownames(GenoM)[ IndivCallRate > 0.8]

## End(Not run)
```

ComparePairs *Comparison of all pairwise relationships in 2 pedigrees*

Description

Compare, count and identify different types of relative pairs between two pedigrees. The matrix returned by DyadCompare [Deprecated] is a subset of the matrix returned here using default settings.

Usage

```
ComparePairs(
  Ped1 = NULL,
  Ped2 = NULL,
  Pairs2 = NULL,
  GenBack = 1,
  patmat = FALSE,
  DumPrefix = c("F0", "M0"),
  Return = "Counts"
)
```

Arguments

Ped1	(Original/reference) pedigree, dataframe with 3 columns: id-dam-sire
Ped2	Second (inferred) pedigree
Pairs2	dataframe with relationships categories between pairs of individuals, instead of or in addition to Ped2, e.g. as returned by GetMaybeRel. First three columns: ID1-ID2-relationship, column names and any additional columns are ignored.
GenBack	Number of generations back to consider; 1 returns parent-offspring and sibling relationships, 2 also returns grandparental, avuncular and first cousins. GenBack >2 is not implemented.
patmat	logical, distinguish between paternal versus maternal relative pairs?
DumPrefix	character vector of length 2 with the dummy prefixes in Ped1 and/or Ped2. IDs starting with these prefixes will not be excluded, but individuals with dummy parents are compared. Use GetRelCat on a single pedigree to find relationships with dummies.
Return	Return a matrix with Counts or a Summary of the number of identical relationships and mismatches per relationship, or detailed results as a 2xNxN Array or as a Dataframe. All returns a list with all four.

Details

If Pairs2 is as returned by GetMaybeRel (identified by the additional column names 'LLR' and 'OH'), these relationship categories are appended with an '?' in the output, to distinguish them from those derived from Ped2.

When Pairs2$TopRel contains values other than the ones listed among the return values for the combination of patmat and GenBack, they are prioritised in decreasing order of factor levels, or in decreasing alphabetical order, and before the default (ped2 derived) levels.

Value

a matrix with counts, a 3D array or a 4-column dataframe, depending on Return, with by default (GenBack=1,patmat=FALSE) the following 7 relationships:

S	Self (not in counts)
MP	Parent

O	Offspring (not in counts)
FS	Full sibling
HS	Half sibling
U	Unrelated, or otherwise related
X	Either or both individuals not occurring in both pedigrees

Where in the array and dataframe, 'MP' indicates that the second (column) individual is the parent of the first (row) individual, and 'O' indicates the reverse.

When `GenBack=2,patmat=TRUE`, the following relationships are distinguished:

S	Self (not in counts)
M	Mother
P	Father
O	Offspring (not in counts)
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 (not in counts
FA	Full avuncular; maternal or paternal aunt or uncle
HA	Half avuncular
FN	Full nephew/niece (not in counts
HN	Half nephew/niece (not in counts
FC1	Full first cousin
DFC1	Double full first cousin
U	Unrelated, or otherwise related
X	Either or both individuals not occurring in both pedigrees

Note that for avuncular and cousin relationships no distinction is made between paternal versus maternal, as this may differ between the two individuals and would generate a large number of subclasses. When a pair is related via multiple paths, the first-listed relationship is returned.

When `GenBack=1,patmat=TRUE` the categories are (S)-M-P-(O)-FS-MHS-PHS- U-X. When `GenBack=2,patmat=FALSE`, MGM, MGF, PGM and PGF are combined into GP, with the rest of the categories analogous to the above.

Note that in the dataframe each pair is listed twice, e.g. once as P and once as O, or twice as FS.

When `Return = "Counts"` (the default), a matrix with counts is returned, with the classification in Ped1 on rows and that in Ped2 in columns. Counts for 'symmetrical' pairs ("FS", "HS", "MHS", "PHS", "FC1", "DFC1", "U","X") are divided by two.

When `Return = 'Summary'`, the counts table is distilled down into a matrix with four columns, which names assuming `Ped1` is the true pedigree:

n	total number of pairs with that relationship in Ped1
OK	Number of pairs with same relationship in Ped2 as in Ped1
lo	Number of pairs with 'lower' relationship in Ped2 as in Ped1 (see ranking above), but not unrelated in Ped2
hi	Number of pairs with 'higher' relationship in Ped2 as in Ped1

When `Return = "Array"`, the first dimension is 1=Ped1, 2=Ped2, the 2nd and 3rd dimension are the two individuals of the pair.

When `Return = "Dataframe"`, the columns are

id.A	First individual of the pair
id.B	Second individual of the pair
RC1	the relationship category in Ped1, as a factor with all considered categories as levels, including those with 0 count
RC2	the relationship category in Ped2

See Also

`PedCompare` for individual-based comparison; `GetRelCat` for pairs of relatives within a single pedigree.

Examples

```
## Not run:
data(Ped_HSg5, SimGeno_example, LH_HSg5, package="sequoia")
SeqOUT <- sequoia(GenoM = SimGeno_example, LifeHistData = LH_HSg5,
                  MaxSibIter = 0)
ComparePairs(Ped1=Ped_HSg5, Ped2=SeqOUT$Pedigree, Return="Counts")
# matrix with counts of pairs
RC.A <- ComparePairs(Ped1=Ped_HSg5, Ped2=SeqOUT$Pedigree, Return="Array")
RC.A[, "a05017", "b05018"] # check specific pairs

RC.DF <- ComparePairs(Ped1=Ped_HSg5, Ped2=SeqOUT$Pedigree,
  Return="Dataframe")
RC.DF[RC.DF$id.A=="a05017" & RC.DF$id.B=="b05018", ] # check specific pairs
table(RC.DF$Ped1, RC.DF$Ped2)
# incl. S,O,GO,FN,HN; duplicated counts for FS,HS,FC1,DFC1,U,X
Mismatches <- RC.DF[RC.DF$Ped1 != RC.DF$Ped2, ]

Maybe <- GetMaybeRel(SimGeno_example, SeqList=SeqOUT, ParSib="sib")
cp <- ComparePairs(Ped1=Ped_HSg5, Ped2=SeqOUT$Pedigree,
                   Pairs2=Maybe$MaybeRel, Return="All")
cp$Counts[, colSums(cp$Counts)>0]
cp$Summary[,"OK"] / cp$Summary[,"n"]  # pairwise assignment rate


## End(Not run)
```

DyadCompare *Compare dyads*

Description

Count the number of half and full sibling pairs correctly and incorrectly assigned. DEPRECATED - SEE ComparePairs

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)
```

`ErrToM` *Generate error matrix*

Description

Generate a matrix with the probabilities of observed genotypes (columns) conditional on actual genotypes (rows), or return a function to generate such matrices (using a single value Err as input to that function)

Usage

```
ErrToM(Err = NA, flavour = "version2.0", Return = "matrix")
```

Arguments

`Err`	estimated genotyping error rate, as a single number or 3x3 or 4x4 matrix. If a single number, an error model is used that aims to deal with scoring errors typical for SNP arrays. If a matrix, this should be the probability of observed genotype (columns) conditional on actual genotype (rows). Each row must therefore sum to 1. If `Return='function'`, this may be `NA`.
`flavour`	matrix-generating function, or one of 'version2.0', 'version1.3' (='SNPchip'), 'version1.1' (='version111'), referring to the sequoia version in which it was used as default. Ignored if `Err` is a matrix and `Return='matrix'` (in which case the matrix will only be checked for validity).
`Return`	output, 'matrix' (always 3x3) or 'function'.

Details

By default (`flavour = "SNPchip"`), `Err` is interpreted as a locus-level error rate (rather than allele-level), and equals the probability that an actual heterozygote is observed as either homozygote (i.e., the probability that it is observed as AA = probability that observed as aa = `Err`/2). The probability that one homozygote is observed as the other is $(\texttt{Err}/2)^2$.

The inbuilt 'flavours' correspond to the presumed and simulated error structures, which have changed with sequoia versions. The most appropriate error structure will depend on the genotyping platform; 'version0.9' and 'version1.1' were inspired by SNP array genotyping while 'version1.3' and 'version2.0' are intended to be more general.

version2.0:

	0	**1**	**2**
0	$(1-E/2)^2$	$E(1-E/2)$	$(E/2)^2$
1	$E/2$	$1-E$	$E/2$
2	$(E/2)^2$	$E(1-E/2)$	$(1-E/2)^2$

version1.3

	0	**1**	**2**
0	$1 - E - (E/2)^2$	E	$(E/2)^2$
1	$E/2$	$1 - E$	$E/2$
2	$(E/2)^2$	E	$1 - E - (E/2)^2$

version1.1

	0	**1**	**2**
0	$1 - E$	$E/2$	$E/2$
1	$E/2$	$1 - E$	$E/2$
2	$E/2$	$E/2$	$1 - E$

version0.9 (not recommended)

	0	**1**	**2**
0	$1 - E$	E	0
1	$E/2$	$1 - E$	$E/2$
2	0	E	$1 - E$

Value

either a 3x3 matrix, or a function generating a 3x3 matrix.

`EstConf` *Estimate confidence probability*

Description

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

Usage

```
EstConf(
  Pedigree = NULL,
  LifeHistData = NULL,
  args.sim = list(nSnp = 400, SnpError = 0.001, ParMis = c(0.4, 0.4)),
  args.seq = list(MaxSibIter = 10, Err = 0.001, Tassign = 0.5),
  nSim = 10,
  quiet = TRUE
)
```

Arguments

Pedigree	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.
quiet	suppress messages. 'very' also suppresses simulation counter, TRUE just runs SimGeno and sequoia quietly.

Details

The confidence probability is taken as the number of correct (matching) assignments, divided by all assignments made in the *observed* (inferred-from-simulated) pedigree. In contrast, the false negative & false positive assignment rates are proportions of the number of parents in the *true* (reference) pedigree. Each rate is calculated separatedly for dams & sires, and separately for each category (**G**enotyped/**D**ummy(fiable)/**X** (none)) of individual, parent and co-parent.

This function does not know which individuals in Pedigree are genotyped, so the confidence probabilities need to be added to the Pedigree by the user as shown in the example at the bottom.

A confidence of '1' assignments on simulated data were correct for that category-combination. It should be interpreted as (and perhaps modified to) $> 1 - 1/N$, where sample size N is given in the last column of the ConfProb and PedErrors dataframes in the output. The same applies for a false negative/positive rate of '0'.

Value

a list, with the main results in dataframe ConfProb and array PedErrors. ConfProb has 7 columns:

id.cat, dam.cat, sire.cat	Category of the focal individual, dam, and sire, in the pedigree inferred based on the simulated data. Coded as G=genotyped, D=dummy, X=none
dam.conf	Probability that the dam is correct, given the categories of the assigned dam and sire (ignoring whether or not the sire is correct). Rounded to nchar(N) significant digits
sire.conf	as dam.conf, for the sire
pair.conf	Probability that both dam and sire are correct, given their categories
N	Number of individuals per category-combination, across all nSim simulations

array PedErrors has three dimensions:

class	• FalseNeg(atives): could have been assigned but was not (individual + parent both genotyped or dummyfiable; P1only in PedCompare).

- FalsePos(itives): no parent in reference pedigree, but one was assigned based on the simulated data (P2only)
- Mismatch: different parents between the pedigrees

`cat` Category of individual + parent, as a two-letter code where the first letter indicates the focal individual and the second the parent; G=Genotyped, D=Dummy, T=Total

`parent` dam or sire

The other list elements are:

`Pedigree.reference` the pedigree from which data was simulated

`Pedigree.inferred` a list with for each iteration the inferred pedigree based on the simulated data

`SimSNPd` a list with for each iteration the IDs of the individuals simulated to have been genotyped

`RunParams` a list with the 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']`.

Assumptions

Because the actual true pedigree is (typically) unknown, the provided reference pedigree is used as a stand-in and assumed to be the true pedigree, with unrelated founders. It is also assumed that the probability to be genotyped is equal for all parents; in each iteration, a new random set of parents (proportion set by `ParMis`) is mimicked to be non-genotyped. In addition, SNPs are assumed to segregate independently.

See Also

`SimGeno`, `sequoia`, `PedCompare`

Examples

```
## Not run:
data(Ped_HSg5, LH_HSg5, package="sequoia")

## Example A: parentage assignment only
conf.A <- EstConf(Pedigree = 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-3, Tassign=0.5),
   nSim = 2)

# parent-pair confidence, per category:
conf.A$ConfProb

# calculate (correct) assignment rates (ignores co-parent)
1 - apply(conf.A$PedErrors, c(1,3), sum, na.rm=TRUE)
```

```
## Example B: with sibship clustering, based on sequoia inferred pedigree
RealGenotypes <- SimGeno(Ped = Ped_HSg5, nSnp = 100,
                         ParMis=c(0.19,0.53), SnpError = 6e-3)
SeqOUT <- sequoia(GenoM = RealGenotypes,
                  LifeHistData = LH_HSg5,
                  Err=5e-3, MaxSibIter=10)

conf.B <- EstConf(Pedigree = SeqOUT$Pedigree,
              LifeHistData = LH_HSg5,
               args.sim = list(nSnp = 100, SnpError = 5e-3,
                               ParMis=c(0.2, 0.5)),
              args.seq = list(Err=5e-3, MaxSibIter = 10),
              nSim = 3)
Ped.withConf <- getAssignCat(Pedigree = SeqOUT$Pedigree,
                             Genotyped = rownames(RealGenotypes))
Ped.withConf <- merge(Ped.withConf, conf.B$ConfProb, all.x=TRUE)
Ped.withConf <- Ped.withConf[, c("id","dam","sire", "dam.conf", "sire.conf",
                                 "id.cat", "dam.cat", "sire.cat")]

## 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). This joining can be reversed using PedStripFID.
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 occasional 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

CheckGeno,SnpStats,LHConvert

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)
```

---

getAssignCat *Assignability of reference pedigree*

Description

Identify which individuals are genotyped, and which can potentially be substituted by a dummy individual. 'Dummifiable' are those non-genotyped individuals with at least 2 genotyped offspring, or at least 1 genotyped offspring and 1 genotyped parent.

Usage

```
getAssignCat(Pedigree, Genotyped)
```

Arguments

Pedigree	dataframe with columns id-dam-sire. Reference pedigree.
Genotyped	character vector with ids of genotyped individuals.

Details

It is assumed that all individuals in Genotyped have been genotyped for a sufficient number of SNPs. To identify samples with a too-low call rate, use CheckGeno. To calculate the call rate for all samples, see the examples below.

Some parents indicated here as assignable may never be assigned by sequoia, for example parent-offspring pairs where it cannot be determined which is the older of the two, or grandparents that are indistinguishable from full avuncular (i.e. genetics inconclusive because the candidate has no parent assigned, and ageprior inconclusive).

Value

the Pedigree dataframe with 2 additional columns, dam.cat and sire.cat, with coding similar to that used by PedCompare:

GG	Genotyped individual, genotyped parent
GD	Genotyped individual, Dummy parent; i.e. 'id' has at least 1 genotyped sibling or a genotyped grandparent
DG	Dummy individual, Genotyped parent; i.e. 'id' has at least 1 genotyped offspring, and parent is assignable as grandparent of the dummy-substituted-individual's offspring
DD	Dummy individual, Dummy parent
X	Either or both id and parent is/are not genotyped, and has/have no genotyped offspring, and therefore the parent- offspring link cannot be assigned.
NA	No parent in Pedigree

Examples

```
data(Ped_HSg5, SimGeno_example, package="sequoia")
PedA <- getAssignCat(Ped_HSg5, rownames(SimGeno_example))
table(PedA$dam.cat, PedA$sire.cat, useNA="ifany")

# calculate call rate
## Not run:
CallRates <- apply(MyGenotypes, MARGIN=1,
                   FUN = function(x) sum(x!=-9)) / ncol(MyGenotypes)
hist(CallRates, breaks=50, col="grey")
GoodSamples <- rownames(MyGenotypes)[ CallRates > 0.8]
threshold depends on total number of SNPs, genotyping errors, proportion of
candidate parents that are SNPd (sibship clustering is more prone to false
positives).
PedA <- getAssignCat(MyOldPedigree, rownames(GoodSamples))

## 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,
  Pedigree = NULL,
  LifeHistData = NULL,
  ParSib = "par",
  Complex = "full",
  Err = 1e-04,
  ErrFlavour = "version2.0",
  MaxMismatch = NA,
  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.
Pedigree	dataframe 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, as a single number or 3x3 matrix. If a matrix, this should be the probability of observed genotype (columns) conditional on actual genotype (rows). Each row must therefore sum to 1.

ErrFlavour	function that takes Err as input, and returns a 3x3 matrix of observed (columns) conditional on actual (rows) genotypes, or choose from inbuilt ones as used in sequoia 'version2.0', 'version1.3', or 'version1.1'. Ignored if Err is a matrix. See ErrToM.
MaxMismatch	DEPRECATED AND IGNORED. Now calculated using CalcMaxMismatch.
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 Pedigree.
quiet	suppress messages

Value

A list with

MaybeParent or MaybeRel
: A dataframe with non-assigned likely relatives, with columns ID1 - ID2 - TopRel - LLR - OH - BirthYear1 - BirthYear2 - AgeDif - Sex1 - Sex2 - SNPdBoth

MaybeTrio
: A dataframe with non-assigned parent-parent-offspring trios, with columns id - parent1 - parent2 - LLRparent1 - LLRparent2 - LLRpair - OHparent1 - OHparent2 - MEpair - SNPd.id.parent1 - SNPd.id.parent2

The following categories are used in column 'TopRel', indicating the most likely relationship category:

PO	Parent-Offspring
FS	Full Siblings
HS	Half Siblings
GP	GrandParent - grand-offspring
FA	Full Avuncular (aunt/uncle)
2nd	2nd degree relatives, not enough information to distinguish between HS,GP and FA
Q	Unclear, but probably 1st, 2nd or 3rd degree relatives

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)

Maybe <- GetMaybeRel(GenoM = SimGeno_example,
                       Pedigree = SeqOUT$PedigreePar, ParSib="sib")
```

```
## 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, Pedigree, GenBack = 2, patmat = TRUE)
```

Arguments

`x`	The focal individual, either its rownumber in the pedigree or ID.
`Pedigree`	dataframe 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.
`patmat`	logical, distinguish between paternal versus maternal relative pairs?

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)

See Also

ComparePairs to compare pairwise relationships between 2 pedigrees.

Examples

```
data(Ped_griffin)
# find all relatives of a specific individual
Rel42 <- GetRelCat("i042_2003_F", Ped_griffin)
Rel42[Rel42 != "U"]

# make NxN matrix with relationship categories:
Ped_griffin_sub <- Ped_griffin[Ped_griffin$birthyear<2003,]  # quicker
RCM <- sapply(seq_along(Ped_griffin_sub$id), GetRelCat, Ped_griffin_sub)
table(RCM)
#   M  MHS    O    P    S    U
#  10    6   16    6   40 1522
# note that sibling & cousin pairs are counted twice!
# Parent-offspring pairs are counted directionally:
# once as offspring (O), once as mother (M) or father (P)

# for large pedigrees, table(factor()) is much faster:
table(factor(RCM, levels=c("M","P","FS","MHS","PHS","U")))

# list the maternal half-siblings:
these <- which(RCM=="MHS", arr.ind=TRUE)
data.frame(id1 = Ped_griffin_sub$id[these[,1]],
           id2 = Ped_griffin_sub$id[these[,2]])


# Get Colony-style lists of full sibs & half sibs dyads:
## Not run:
RCM <- sapply(seq_along(MyPedigree$id), GetRelCat, Pedigree = MyPedigree,
              GenBack = 1, patmat = FALSE)
# rownumbers of pairs of FS & HS
FullSibDyads <- which(RCM == "FS", arr.ind=TRUE)
HalfSibDyads <- which(RCM == "HS", arr.ind=TRUE)

# each pair is listed 2x - fix:
FullSibDyads <- FullSibDyads[FullSibDyads[,1] < FullSibDyads[,2], ]
HalfSibDyads <- HalfSibDyads[HalfSibDyads[,1] < HalfSibDyads[,2], ]

# translate rownumbers into IDs
MyPedigree$id <- as.character(MyPedigree$id)
FullSibDyads <- cbind(MyPedigree$id[FullSibDyads[,1]],
```

```
                        MyPedigree$id[FullSibDyads[,2]])
HalfSibDyads <- cbind(MyPedigree$id[HalfSibDyads[,1]],
                        MyPedigree$id[HalfSibDyads[,2]])

## 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

For various categories of pairwise relatives (R), calculate age-difference (A) based probability ratios $P(A|R)/P(A)$.

Usage

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

Arguments

Pedigree	dataframe with id - dam - sire in columns 1-3, and optional column with birth years. Other columns are ignored.
LifeHistData	dataframe with 3 or 5 columns: id - sex (not used) - birth year (- BY.min - BY.max), with unknown birth years coded as negative numbers 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. It may include individuals not in the pedigree, and not all individuals in the pedigree need to be in LifeHistData.
MaxAgeParent	maximum age of a parent, a single number (max across dams and sires) or a vector of length two (dams, sires). If NULL, it will be estimated from the data. If there are fewer than 20 parents of either sex assigned, MaxAgeParent is set to the maximum age difference in the birth year column of Pedigree or LifeHistData.
Discrete	Discrete generations? By default (NULL), discrete generations are assumed if all parent-offspring pairs have an age difference of 1, and all siblings an age difference of 0, and there are at least 20 pairs of each category (mother, father, maternal sibling, paternal sibling). Otherwise, overlapping generations are presumed. When Discrete=TRUE (explicitly or deduced), Smooth and Flatten are always automatically set to FALSE. Use Discrete=FALSE to enforce (potential for) overlapping generations.

`Flatten` To deal with small sample sizes for some or all relationships, calculate weighed average between the observed age difference distribution among relatives and a flat (0/1) distribution. When `Flatten=NULL` (the default) automatically set to TRUE when there are fewer than 20 parents with known age of either sex assigned, or fewer than 20 maternal or paternal siblings with known age difference. Also advisable if the sampled relative pairs with known age difference are non-typical of the pedigree as a whole.

`lambdaNW` Control weighing factors when `Flatten=TRUE`. Weights 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. Large values (>0.2) put strong emphasis on the pedigree, small values (<0.0001) cause the pedigree to be ignored. Default results in $W = 0.5$ for $N = 100$.

`Smooth` Smooth the tails of and any dips in the distribution? Sets dips (<10% of average of neighbouring ages) to the average of the neighbouring 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. Peaks are not smoothed out, as these are less likely to cause problems than dips, and are more likely to be genuine characteristics of the species. Is set to `FALSE` when generations do not overlap (`Discrete=TRUE`).

`Plot` plot a heatmap of the results? Only when `Pedigree` is provided

`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

The ratio $P(A|R)/P(A)$ is the ratio between the observed counts of pairs with age difference A and relationship R ($N_{A,R}$), and the expected counts if age and relationship were independent ($N_{.,.} * p_A * p_R$).

During pedigree reconstruction, the ratios $P(A|R)/P(A)$ calculated here are multiplied by the age-independent genetic-only $P(R|G)$ to obtain a probability that the pair are relatives of type R conditional on both their age difference and their genotypes (i.e. using Bayes' theorem, $P(R|A,G) = P(A|R)/P(A) * P(R|G)$).

The age-difference prior is used for pairs of genotyped individuals, as well as for dummy individuals. This assumes that the propensity for a pair with a given age difference to both be sampled does not depend on their relationship, so that the ratio $P(A|R)/P(A)$ does not differ between sampled and unsampled pairs.

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 five columns, one for each relationship type, with abbreviations:

`M` Mothers

P	Fathers
FS	Full siblings
MS	Maternal half-siblings
PS	Paternal half-siblings

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, 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 Pedigree.
Weights	vector length 4, the weights used to flatten the distributions
LR.RU.A.unweighed	matrix with nAgeClasses+1 rows and 5 columns; LR.RU.A prior to flattening and smoothing
Specs.AP	the names of the input Pedigree and LifeHistData (or NULL), the 'effective' settings of Discrete, Smooth, and Flatten, and the value of lambdaNW

CAUTION

The small sample correction with Smooth and/or Flatten prevents errors in one dataset, but may introduce errors in another; a single solution that fits to the wide variety of life histories and datasets is impossible. Please do inspect the matrix, e.g. with PlotAgePrior.

Single cohort

When no birth year information is given, or all individuals have the same birth year, it is assumed that a single cohort has been analysed and a matrix with 0's and 1's is returned. When Discrete=FALSE, avuncular pairs are assumed potentially present, while when Discrete=TRUE avuncular is not considered as a relationship possibility.

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, but always before their putative offspring (e.g. parent's BirthYear= 1 (or 2001) and offspring BirthYear=5 (or 2005)). Negative numbers and NA's are interpreted as unknown, and fractional numbers are not allowed.

Maximum parental age

The number of rows in the output ageprior matrix equals the maximum parental age +1 (the first row is for age difference 0). The maximum parental age equals:

- the maximum age of parents if a pedigree is provided, or
- the (largest) value of MaxAgeParent, or

- 1, if generations are discrete, or
- the maximum range of birth years in LifeHistData (including BY.min and BY.max, when provided)

Exception is when `MaxAgeParent` is larger than the maximum age of parents in the provided skeleton pedigree, then `MaxAgeParent` is used. Thus, `MaxAgeParent` can be used when the birth year range in LifeHistData and/or the age distribution of assigned parents does not capture the absolutely maximum age of parents. Not adjusting this may hinder subsequent assignment of both dummy parents and grandparents.

See Also

`sequoia` (and its argument `args.AP`), `PlotAgePrior` for visualisation. The age vignette gives further details, mathematical justification, and some examples.

Examples

```
data(LH_HSg5, Ped_HSg5, package="sequoia")

# no pedigree available:
MakeAgePrior(LifeHistData = LH_HSg5)
MakeAgePrior(LifeHistData = LH_HSg5, Discrete=TRUE)
MakeAgePrior(LifeHistData = LH_HSg5, MaxAgeParent = c(2,3))
## Not run:
# with pedigree:
MakeAgePrior(Pedigree=Ped_HSg5[1:100,], LifeHistData = LH_HSg5)
MakeAgePrior(Ped_HSg5[1:100,], LH_HSg5, Discrete=FALSE)
# With 'Flatten', the value depens on the no. pairs per relationship:
MakeAgePrior(Ped_HSg5[1:100,], LH_HSg5, Flatten=TRUE)
AP.all <- MakeAgePrior(Ped_HSg5[1:200,], LH_HSg5, Flatten=TRUE)
AP.all$tblA.R

## End(Not run)
```

`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,
  DumPrefix = c("F0", "M0"),
  SNPd = NULL,
  Symmetrical = TRUE
)
```

Arguments

Ped1	original pedigree, dataframe with columns id-dam-sire; only the first 3 columns will be used.
Ped2	inferred pedigree, e.g. SeqOUT$Pedigree or SeqOUT$PedigreePar, with columns id-dam-sire.
DumPrefix	character vector of length 2 with the dummy prefixes in Pedigree 2; all IDs not starting with the Dummy prefix are taken as genotyped if SNPd=NULL.
SNPd	character vector with IDs of genotyped individuals.
Symmetrical	When determining the category of individuals (Genotyped/Dummy/X), use the 'highest' category across the two pedigrees (TRUE, default) or only consider Ped1 (Symmetrical = FALSE).

Details

The comparison is divided into different classes of 'assignable' parents (getAssignCat). 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, see below

`Counts.detail` a large numeric array with number of matches and mismatches, with more detail for all possible combination of categories

`MergedPed` A dataframe with 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.2), and the best-matching individual in Pedigree 1 (id.1)

`Mismatch` A subset of MergedPed with mismatches between Ped1 and Ped2, as defined below

`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

'MergedPed', 'Mismatch', 'Ped1only' and 'Ped2only' provide the following columns:

`id` All ids in both Pedigree 1 and 2. For dummy individuals, this is the id *in pedigree 2*

`dam.1, sire.1` parents in Pedigree 1

`dam.2, sire.2` parents in Pedigree 2

`id.r, dam.r, sire.r`
The *real* id of dummy individuals or parents in Pedigree 2, i.e. the best-matching non-genotyped individual in 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"

`id.dam.cat, id.sire.cat`
the category of the individual (first letter) and *highest category* of the dam (sire) in Pedigree 1 or 2: G=Genotyped, D=(potential) dummy, X=none. Individual, one-letter categories are generated by `getAssignCat`. Using the 'best' category from both pedigrees makes comparison between two inferred pedigrees symmetrical and more intuitive.

`dam.class, sire.class`
classification of dam and sire: Match, Mismatch, P1only, P2only, or '_' when no parent is assigned in either pedigree

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 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 'DT' is the assignment of fathers to both maternal and paternal sibships (i.e., to dummies of both sexes).

In 'ConsensusPed', the priority used is parent.r (if not "nomatch") > parent.2 > parent.1. The columns 'id.cat', dam.cat' and 'sire.cat' have two additional levels compared to 'MergedPed':

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, Unconfirmed (parent in Pedigree 1, with no dummy match in Pedigree 2)
X	No parent in either pedigree

Assignable

Note that 'assignable' may be overly optimistic. Some parents from Ped1 indicated as assignable may never be assigned by sequoia, for example parent-offspring pairs where it cannot be determined which is the older of the two, or grandparents that are indistinguishable from full avuncular (i.e. genetics inconclusive because the candidate has no parent assigned, and ageprior inconclusive).

Dummifiable

Considered as potential dummy individuals are 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.

Genotyped 'mystery samples'

If Pedigree 2 includes samples for which the ID is unknown, the behaviour of PedCompare depends on whether the temporary IDs for these samples are included in SNPd. If they are included, matching (actual) IDs in Pedigree 1 will be flagged as mismatches (because the IDs differ). If they are not included in SNPd, or SNPd is not explicitly provided, matches are accepted, as the situation is indistinguishable from comparing dummy parents across pedigrees.

This is of course all conditional on relatives of the mystery sample being assigned in Pedigree 2.

Author(s)

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

See Also

ComparePairs for comparison of all pairwise relationships in 2 pedigrees, EstConf for repeated simulate-reconstruct-compare, sequoia for the main pedigree reconstruction function, getAssignCat for all parents in the reference pedigree that could have been assigned.

Examples

```
## Not run:
data(Ped_HSg5, SimGeno_example, LH_HSg5, package="sequoia")
SeqOUT <- sequoia(GenoM = SimGeno_example, LifeHistData = LH_HSg5, Err=0.001)
compare <- PedCompare(Ped1=Ped_HSg5, Ped2=SeqOUT$Pedigree)
compare$Counts   # 2 non-assigned, due to simulated genotyping errors
compare$Counts["TT",,]  # totals only
compare$Counts[,,"dam"]  # dams only

# inspect 'assignable but non-assigned in Ped2'
compare$P1only[compare$P1only$Cat=="GG", ]
# further inspection:
head(compare$MergedPed)
compare$MergedPed[which(compare$MergedPed$dam.1=="a00001"), ]

# get an overview of all non-genotyped -- dummy matches
BestMatch <- compare$MergedPed[!is.na(compare$MergedPed$id.r),
                               c("id", "id.r")]

# success of paternity assignment, if genotyped mother correctly assigned
dimnames(compare$Counts.detail)
compare$Counts.detail["G","G",,"Match",]

## End(Not run)
```

PedPolish *Pedigree fix*

Description

Ensure all parents & all genotyped individuals are included, remove duplicates, rename columns, and replace 0 by NA or v.v.

Usage

```
PedPolish(
  Ped,
  GenoNames = NULL,
  ZeroToNA = TRUE,
```

```
  NAToZero = FALSE,
  DropNonSNPd = TRUE,
  FillParents = FALSE
)
```

Arguments

Ped	dataframe where the first 3 columns are id, dam, sire
GenoNames	character vector with ids of genotyped individuals (rownames of genotype matrix)
ZeroToNA	logical, replace 0's for missing values by NA's (defaults to TRUE)
NAToZero	logical, replace NA's for missing values by 0's. If TRUE, ZeroToNA is automatically set to FALSE
DropNonSNPd	logical, remove any non-genotyped individuals (but keep non-genotyped parents), & sort pedigree in order of GenoNames
FillParents	logical, for individuals with only 1 parent assigned, set the other parent to a dummy (without assigning siblings or grandparents). Makes the pedigree compatible with R packages and software that requires individuals to have either 2 or 0 parents, such as kinship.

Details

recognized column names are any that contain:

dam "dam", "mother", "mot", "mom", "mum", "mat"

sire "sire", "father", "fat", "dad", "pat"

sequoia requires the column order id - dam - sire; columns 2 and 3 are swapped if necessary.

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_griffin` *Example pedigree: griffins*

Description

Example Pedigree used in the ageprior vignette, with overlapping generations.

Usage

```
data(Ped_griffin)
```

Format

A data frame with 200 rows and 4 variables (id, dam, sire, birthyear)

Author(s)

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

See Also

`SeqOUT_griffin` for a sequoia run on simulated genotype data based on this pedigree; `Ped_HSg5` for another pedigree, `sequoia`

Ped_HSg5 *Example pedigree*

Description

This is Pedigree II in the paper, with discrete generations and considerable inbreeding

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 as a heatmap

Usage

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

Arguments

AP	matrix with age priors (P(A|R)/P(A)) with age differences in rows and relationships in columns; by default M: maternal parent (mother), P: paternal parent (father), FS: full siblings, MS: maternal siblings (full + half), PS: paternal siblings.
legend	if TRUE, a new plotting window is started and layout is used to plot a legend next to the main plot. Set to FALSE if you want to add it as panel to an existing plot (e.g. with par(mfcol=c(2,2))).

Value

a heatmap

See Also

MakeAgePrior, SummarySeq

Examples

```
## Not run:
PlotAgePrior(SeqOUT$AgePriors)

## End(Not run)
```

SeqOUT_griffin *Example sequoia output (griffins)*

Description

Example output of a sequoia run including sibship clustering, based on the griffin pedigree.

Usage

```
data(SeqOUT_griffin)
```

Format

a list, see sequoia

Author(s)

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

See Also

Ped_griffin,sequoia

Examples

```
## Not run:
GenoS <- SimGeno(Ped.griffin, nSnp=400, ParMis=0.4)
griffin.sex <- sapply(Ped.griffin$ID,
                      function(x) substr(x, start=nchar(x), stop=nchar(x)))
LH_griffin <- data.frame(ID = Ped_griffin$ID,
                         Sex = ifelse(griffin.sex=="F", 1, 2),
                         BirthYear = Ped_griffin$BY)
SeqOUT_griffin <- sequoia(GenoS, LH_griffin,
                  MaxSibIter = 10,
```

```
                  args.AP = list(Smooth = FALSE))
## End(Not run)
```

---

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,
  ErrFlavour = "version2.0",
  MaxMismatch = NA,
  Tfilter = -2,
  Tassign = 0.5,
  MaxSibshipSize = 100,
  DummyPrefix = c("F", "M"),
  Complex = "full",
  UseAge = "yes",
  args.AP = list(Flatten = NULL, Smooth = TRUE),
  FindMaybeRel = FALSE,
  CalcLLR = TRUE,
  quiet = FALSE,
  Plot = NULL
)
```

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 (optionally 5): **ID** max. 30 characters long, **Sex** 1 = females, 2 = males, other = unknown, except 4 = hermaphrodite, **BirthYear** birth or hatching year, integer, with missing values as NA or any negative value. **BY.min** minimum birth year, only used if BirthYear is missing

BY.max maximum birth year, only used if BirthYear is missing

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 (- BY.min - BY.max).

`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, as a single number or 3x3 matrix. If a matrix, this should be the probability of observed genotype (columns) conditional on actual genotype (rows). Each row must therefore sum to 1.

`ErrFlavour` function that takes `Err` as input, and returns a 3x3 matrix of observed (columns) conditional on actual (rows) genotypes, or choose from inbuilt ones as used in sequoia 'version2.0', 'version1.3', or 'version1.1'. Ignored if `Err` is a matrix. See `ErrToM`.

`MaxMismatch` DEPRECATED AND IGNORED. Now calculated using `CalcMaxMismatch`.

`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` DEPRECATED, advised to run `GetMaybeRel` separately. TRUE/FALSE to identify pairs of non-assigned likely relatives after pedigree reconstruction. Can be time-consuming in large datasets.

CalcLLR	calculate log-likelihood ratios for all assigned parents (genotyped + dummy; parent vs. otherwise related). Time-consuming in large datasets. Can be done separately with CalcOHLLR.
quiet	suppress messages: TRUE/FALSE/"verbose".
Plot	display plots from SnpStats, MakeAgePrior, and SummarySeq. Defaults (NULL) to TRUE when quiet=FALSE or "verbose", and FALSE when quiet=TRUE. If you get errors an error 'figure margins too large', enlarge the plotting area (drag with mouse). 'invalid graphics state' error can be dealt with by clearing the plotting area with dev.off().

Details

Dummy parents of sibships are denoted by F0001, F0002, ... (mothers) and M0001, M0002, ... (fathers), are appended to the bottom of the pedigree, and may have been assigned real or dummy parents themselves (i.e. sibship-grandparents). A dummy parent is not assigned to singletons.

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 probability ratios for each relationship, used for full pedigree reconstruction; see MakeAgePrior for details. When running only parentage assignment (MaxSibIter=0) the returned AgePriors has been updated to incorporate the information of the assigned parents, and is ready for use during full pedigree reconstruction.
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, row numbers of duplicated IDs in life history dataframe. For convenience only, but may signal a problem. The first entry is used.
ErrM	Error matrix; probability of observed genotype (columns) conditional on actual genotype (rows)
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.
LifeHistPar	LifeHist with additional columns 'Sexx' (inferred Sex when assigned as part of parent-pair), 'BY.est' (mode of birth year probability distribution), 'BY.lo' (lower limit of 95% highest density region), 'BY.hi' (higher limit), inferred after parentage assignment. 'BY.est' is NA when the probability distribution is flat between 'BY.lo' and 'BY.hi'.
LifeHistSib	as LifeHistPar, but estimated after full pedigree reconstruction
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.
AgePriorExtra	As AgePriors, but including columns for grandparents and avuncular pairs. NOT updated after parentage assignment, but returned as used during the run.

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 homozygotes, 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 to read in various data formats, CheckGeno, SnpStats to calculate missingness and allele frequencies, MakeAgePrior to estimate effect of age on relationships, GetMaybeRel to find pairs of potential relatives, SummarySeq and PlotAgePrior to visualise results, GetRelCat to turn a pedigree into pairwise relationships, CalcOHLLR to calculate OH and LLR, PedCompare and ComparePairs to compare to a previous pedigree, EstConf and SimGeno to estimate assignment errors, writeSeq to save results, vignette("sequoia") for further details & FAQ.

Examples

```
# ===  EXAMPLE 1: simulate data  ===
data(SimGeno_example, LH_HSg5, package="sequoia")
head(SimGeno_example[,1:10])
head(LH_HSg5)
SeqOUT <- sequoia(GenoM = SimGeno_example, Err = 0.005,
                  LifeHistData = LH_HSg5, MaxSibIter = 0)
names(SeqOUT)
SeqOUT$PedigreePar[34:42, ]

# compare to true (or old) pedigree:
PC <- PedCompare(Ped_HSg5, SeqOUT$PedigreePar)
PC$Counts["GG",,]

## Not run:
SeqOUT2 <- sequoia(GenoM = SimGeno_example, Err = 0.005,
                  LifeHistData = LH_HSg5, MaxSibIter = 10)
SeqOUT2$Pedigree[34:42, ]

PC2 <- PedCompare(Ped_HSg5, SeqOUT2$Pedigree)
PC2$Counts["GT",,]

# important to run with (approx.) correct genotyping error rate:
SeqOUT2.b <- sequoia(GenoM = SimGeno_example, #  Err = 1e-4 by default
                  LifeHistData = LH_HSg5, MaxSibIter = 10)
PC2.b <- PedCompare(Ped_HSg5, SeqOUT2.b$Pedigree)
PC2.b$Counts["GT",,]
```

```
# ===  EXAMPLE 2: real data  ===
# 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",
                     InFormat = "raw")  # can also do Colony format
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)
SummarySeq(SeqOUT)

writeSeq(SeqOUT, folder="sequoia_output")  # several text files

# runtime:
SeqOUT$Specs$TimeEnd - SeqOUT$Specs$TimeStart

## End(Not run)
```

SimGeno *Simulated genotypes*

Description

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

Usage

```
SimGeno(
  Pedigree,
  nSnp = 400,
  ParMis = 0.4,
  MAF = 0.3,
  CallRate = 0.99,
  SnpError = 5e-04,
  ErrorFM = "version2.0",
  ReturnStats = FALSE,
  OutFile = NA,
  Inherit = "autosomal",
  InheritFile = NA,
  quiet = FALSE,
  PropLQ,
  MisHQ,
```

```
  MisLQ,
  ErHQ,
  ErLQ
)
```

Arguments

Pedigree	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 frequencies 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). Inbuilt ones are as used in sequoia 'version2.0', 'version1.3', or 'version1.1'. See details.
ReturnStats	in addition to the genotype matrix, return the input parameters and mean & quantiles of MAF, error rate and call rates.
OutFile	file name 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	file name 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-distribution 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 PedPolish. For founders, i.e. individuals with no known parents, genotypes are drawn according to the provided MAF and assuming Hardy-Weinberg equilibrium. Offspring genotypes are generated following Mendelian inheritance, assuming all loci are completely independent. Individuals with one known parent are allowed: at each locus, one allele is inherited from the known parent, and the other drawn from the genepool according to the provided MAF.

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 = "version2.0"), 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 $(\text{SnpError}/2)^2$.

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 three elements: list 'ParamsIN', matrix 'SGeno', and list 'StatsOUT':

AF	Frequency in 'observed' genotypes of '1' allele
AF.act	Allele frequency in 'actual' (without genotyping errors & missingness)
SnpError	Error rate per SNP (actual /= observed AND observed /= missing)
SnpCallRate	Non-missing per SNP
IndivError	Error rate per individual
IndivCallRate	Non-missing per individual

Disclaimer

This simulation is highly simplistic and assumes that all SNPs segregate completely independently, that the SNPs are in Hardy-Weinberg equilibrium in the pedigree founders. It assumes that genotyping errors are not due to heritable mutations of the SNPs, and that missingness is random and not e.g. due to heritable mutations of SNP flanking regions. 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

the wrapper EstConf for repeated simulation and pedigree reconstruction; MkGenoErrors for fine control over the distribution of genotyping errors in simulated data.

Examples

```
data(Ped_HSg5)
GenoM <- SimGeno(Pedigree = 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(Pedigree = 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 genotype matrix with 214 rows (ids) and 200 columns (SNPs). Each SNP is coded as 0/1/2 copies of the reference allele, with -9 for missing values. Ids are stored as rownames.

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, Pedigree = NULL, ErrFlavour = "version2.0", 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.
Pedigree	a dataframe with 3 columns: ID - parent1 - parent2. Additional columns and non-genotyped individuals are ignored. Used to estimate the error rate.
ErrFlavour	function that takes the genotyping error rate Err as input, and returns a 3x3 matrix of observed (columns) conditional on actual (rows) genotypes, or choose from inbuilt ones as used in sequoia 'version2.0', 'version1.3', or 'version1.1'. See ErrToM.
Plot	show histograms of the results?

Details

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

Value

a matrix with a number of rows equal to the number of SNPs (=number of columns of GenoM), and when no Pedigree is provided 2 columns:

`AF`	Allele frequency of the 'second allele' (the one for which the homozygote is coded 2)
`Mis`	Proportion of missing calls

When a Pedigree is provided, there are 7 additional columns:

`n.dam, n.sire, n.pair`	Number of dams, sires, parent-pairs succesfully genotyped for the SNP
`OHdam, OHsire`	Count of number of opposing homozygous cases
`MEpair`	Count of Mendelian errors, includes opposing homozygous cases
`Err.hat`	Error rate, as estimated from the joined offspring-parent (-parent) genotypes and the presumed error structure (`ErrFlavour`)

Estimated genotyping error

The error rate is estimated from the number of opposing homozygous cases (OH, parent is AA and offspring is aa) Mendelian errors (ME, e.g. parents AA and aa, but offspring not Aa) in parent-parent-offspring trios, and OH cases for offspring with a single genotyped parent.

The estimated error rates will not be as accurate as from duplicate samples, since a single error in an individual with many offspring will be counted many times, while errors in individuals without parents or offspring will not be counted at all. Moreover, a high error rate may interfere with pedigree reconstruction, and succesful assignment will be biased towards parents with lower error count. Nonetheless, it will provide a ballpark estimate for the average error rate, which will be useful for subsequent (rerun of) pedigree reconstruction.

See Also

`GenoConvert` to convert from various data formats; `CheckGeno` to check the data is in valid format for sequoia and exclude monomorphic SNPs etc., `CalcOHLLR` to calculate OH & ME per individual

---

`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,
  Pedigree = NULL,
  DumPrefix = c("F0", "M0"),
  SNPd = NULL,
  Plot = TRUE,
  Panels = "all"
)
```

Arguments

SeqList	the list returned by sequoia. Only elements 'Pedigree' or 'PedigreePar' and 'AgePriors' are used.
Pedigree	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 Pedigree 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.
Panels	character vector with panel(s) to plot. Choose from 'all', 'G.parents' (parents of genotyped individuals), 'D.parents' (parents of dummy individuals), 'sibships' (distribution of sibship sizes), 'LLR' (log10-likelihood ratio parent/otherwise related), 'OH' (count of opposite homozygote SNPs).

Value

A list with the following elements:

PedSummary	a 1-column dataframe with basic summary statistics, as used to be returned by Pedantics' pedStatSummary (now archived on CRAN)
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 for pedigree reconstruction; CalcOHLLR to (re-)calculate opposite homozygosity & parental LLR; PlotAgePrior to visualise just the ageprior.

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, Pedigree=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,
  MaybeRel = NULL,
  PedComp = NULL,
  OutFormat = "txt",
  folder = "Sequoia-OUT",
  file = "Sequoia-OUT.xlsx",
  ForVersion = 2,
  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.
MaybeRel	a list with results from GetMaybeRel (optional).
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'.
ForVersion	choose '1' for back-compatibility with stand-alone sequoia versions 1.x
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.

See Also

writeColumns to write to a text file, using white space padding to keep columns aligned

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