Package 'sequoia'

August 13, 2018

Type Package

Title Pedigree Inference from SNPs

Version 1.1.1

Date 2018-08-13

Description Fast multi-generational pedigree inference from incomplete data on hundreds of SNPs, including parentage assignment and sibship clustering. See citation('sequoia') for more information.

License GPL-2

LazyData TRUE

Imports plyr (>= 1.8.0), stats, utils

RoxygenNote 6.1.0

Suggests xlsx, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Jisca Huisman [aut, cre]

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

Repository CRAN

Date/Publication 2018-08-13 15:20:03 UTC

R topics documented:

DyadCompare 2
EstConf 3
FindFamilies 4
GenoConvert 5
LHConvert 6
LH_HSg5 7
MakeAgeprior 8
MergeFill 9
PedCompare 9
PedStripFID 12

Ped_HSg5 13
sequoia 14
SimGeno 17
SimGeno_example 19
SnpStats 19
writeColumns 20
writeSeq 21

Index **22**

`DyadCompare` *Compare dyads*

Description

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

Usage

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

Arguments

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

Value

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

See Also

`PedCompare`

Examples

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

## End(Not run)
```

EstConf *Estimate confidence probability*

Description

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

Usage

```
EstConf(Ped = NULL, LifeHistData = NULL, Specs = NULL, Full = TRUE,
  nSim = 10, ParMis = 0.4, args.sim = NULL, return.PC = FALSE,
  quiet = TRUE)
```

Arguments

Ped	Reference pedigree from which to simulate, dataframe with columns id-dam-sire. Additional columns are ignored
LifeHistData	Dataframe with id, sex (1=female, 2=male, 3=unknown), and birth year.
Specs	Parameter values for running sequoia, as named vector.
Full	Full pedigree reconstruction (TRUE) or only parentage assignment (FALSE)
nSim	number of simulations to perform.
ParMis	proportion of parents assumed to have a fully missing genotype.
args.sim	list of additional arguments to pass to SimGeno
return.PC	return all PedCompare Counts?
quiet	suppress messages. 'very' also suppresses simulation counter

Details

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

Value

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

When return.PC is TRUE, a list is returned with two arrays: ConfProb contains the average confidence probability across simulations, and SimCounts all counts of matches, mismatches, Pedigree1-only and pedigree2- only per simulation.

Examples

```
## Not run:
data(SimGeno_example, LH_HSg5, package="sequoia")
SeqOUT <- sequoia(GenoM = SimGeno_example,
                  LifeHistData = LH_HSg5, MaxSibIter = 0)
ConfPr <- EstConf(Ped = SeqOUT$PedigreePar, LifeHistData = LH_HSg5,
                  Specs = SeqOUT$Specs, Full = FALSE, nSim = 10)

## 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 file*

Description

Convert a genotype file from PLINK's .raw, or Colony's 2-column-per-marker format, to sequoia's 1-column-per-marker format.

Usage

```
GenoConvert(InFile = NULL, InFormat = "raw", OutFile = NA,
  OutFormat = "seq", UseFID = FALSE, FIDsep = "__", quiet = FALSE)
```

Arguments

InFile	character string with name of genotype file to be converted
InFormat	One of "raw", "col" or "seq", see Details.
OutFile	character string with name of converted file. If NA, return matrix with genotypes in console; if NULL, write to "GenoForSequoia.txt".
OutFormat	as InFormat. Currently raw -> seq, col -> seq and seq -> col are implemented.
UseFID	Use the family ID column in the PLINK file. The resulting ids (rownames of GenoM) will be in the form FID__IID.
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.
quiet	suppress messages

Details

The following formats can be specified by 'InFormat' and 'OutFormat':

- col: No header row, 1 descriptive column, genotypes are coded as numeric values, missing as 0, in 2 columns per marker.
- ped: No header row, 6 descriptive columns, genotypes are coded as A, C, T, G, missing as 0, in 2 columns per marker. NOTE: not yet implented, use PLINK's –recodeA to convert this format to "raw".
- raw: Header row with SNP names, 6 descriptive columns, genotypes are coded as 0, 1, 2, missing as NA, in 1 column per marker.
- seq: No header row, 1 descriptive column genotypes are coded as 0, 1, 2, missing as -9, in 1 column per marker.

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.

Author(s)

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

See Also

LHConvert, PedStripFID

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")

## End(Not run)
```

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(InFile = NULL, UseFID = FALSE, SwapSex = TRUE,
  FIDsep = "__", LHIN = NULL)
```

Arguments

InFile	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.

LHIN — dataframe with additional sex and birth year info. In case of conflicts, LHIN takes priority, with a warning. If UseFID=TRUE, ids are assumed to be 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.

Use with caution, as not extensively tested yet.

Value

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

See Also

GenoConvert, PedStripFID to reverse UseFID

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 BY (birth year, here cohort)

Author(s)

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

References

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

See Also

Ped_HSg5 sequoia

MakeAgeprior *Age priors*

Description

Calculate age-difference based prior probability ratios for various categories of pairwise relatives.

Usage

```
MakeAgeprior(Parents = NULL, LifeHistData = NULL, UseParents = TRUE,
  nAgeClasses = 0)
```

Arguments

`Parents`	dataframe with scaffold pedigree of assigned parents; columns id - dam - sire.
`LifeHistData`	dataframe with 3 columns: • ID: max. 30 characters long, • Sex: 1 = females, 2 = males, other numbers = unkown, • Birth Year: (or hatching year) Negative numbers (and NA's) are interpreted as missing.
`UseParents`	use the age distribution of assigned parents. Otherwise, equal probabilities across all age differences are assumed.
`nAgeClasses`	number of age classes; age prior matrix will have nAgeClasses + 1 rows.

Details

if UseParents = TRUE, Retrieve age distributions of maternal & paternal parents, siblings and grandparents from assigned parents, to use as input for sibship clustering and grandparent assignment. If the lifehistory file indicates a single age class, $MS = PS = 1$ and $MGM = PGF = MGF = UA = 0$.

Value

A matrix with the probability ratio of the (absolute) age difference between two individuals conditional on them being a certain type of relative versus being a random draw from the sample.

Using Bayes' theorem,

$$P(relationship|agedifference) = P(agedifference|relationship)/P(agedifference)*P(relationship)$$

and the values here are multiplied by the age-independent genetic-only $P(relationship)$ inside `sequoia`.

One row per age difference (0 - nAgeClasses), and one column for each relationship type, with abbreviations:

`M`	Mothers
`P`	Fathers

MGM	Maternal grandmother
PGF	Paternal grandfather
MGF	Maternal grandfathers and paternal grandmothers
FS	Full siblings
MS	Maternal siblings
PS	Paternal siblings
UA	Avuncular

For siblings and avuncular relationships absolute age differences are used, as when generations overlap, nephews can be older than their aunts.

MergeFill *special merge*

Description

As regular merge, but combine data from columns with the same name

Usage

```
MergeFill(df1, df2, by, overwrite = FALSE, ...)
```

Arguments

df1	first dataframe (lowest priority if overwrite=TRUE)
df2	second dataframe (highest priority if overwrite=TRUE)
by	columns used for merging, required.
overwrite	If FALSE (the default), NA's in df1 are replaced by values from df2. If TRUE, all values in df1 are overwritten by values from df2, except where df2 has NA.
...	additional arguments to merge, such as all.

PedCompare *Compare two Pedigrees*

Description

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

Usage

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

Arguments

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

Details

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

Value

A list with

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

The first dimension of Counts denotes the following categories:

GG	Genotyped individual, assigned a genotyped parent in either pedigree
GD	Genotyped individual, assigned a dummy parent, or at least 1 genotyped sibling or a genotyped grandparent in Pedigree 1)
GT	Genotyped individual, total
DG	Dummy individual, assigned a genotyped parent (i.e., grandparent of the sibship in Pedigree 2)
DD	Dummy individual, assigned a dummy parent (i.e., avuncular relationship between sibships in Pedigree 2)
DT	Dummy total

TT	Total total, includes all genotyped individuals, plus non-genotyped individuals in Pedigree 1, plus non-replaced dummy individuals (see below) in Pedigree 2

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

The second dimension of Counts gives the outcomes:

Total	The total number of individuals with a parent assigned in either or both pedigrees
Match	The same parent is assigned in both pedigrees (non-missing). For dummy parents, it is considered a match if the inferred sibship which contains the most offspring of a non-genotyped parent, consists for more than half of this individual's offspring.
Mismatch	Different parents assigned in the two pedigrees. When a sibship according to Pedigree 1 is split over two sibships in Pedigree 2, the smaller fraction is included in the count here.
P1only	Parent in Pedigree 1 but not 2; includes non-assignable parents (e.g. not genotyped and no genotyped offspring).
P2only	Parent in Pedigree 2 but not 1.

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

'MergedPed' provides the following columns:

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

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

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

Author(s)

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

See Also

DyadCompare, sequoia.

Examples

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

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

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

## End(Not run)
```

PedStripFID *backtransform IDs*

Description

Reverse the joining of FID and IID in GenoConvert and LHConvert

Usage

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

Arguments

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

Details

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

Value

a pedigree with 6 columns

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

Ped_HSg5 *Example pedigree*

Description

This is Pedigree II in the paper.

Usage

```
data(Ped_HSg5)
```

Format

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

Author(s)

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

References

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

See Also

LH_HSg5 SimGeno_example sequoia

`sequoia` *Pedigree Reconstruction*

Description

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

Usage

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

Arguments

`GenoM`	numeric matrix with genotype data: One row per individual, and one column per SNP, coded as 0, 1, 2 or -9 (missing). Use `GenoConvert` to convert genotype files created in PLINK using –recodeA or in Colony's 2-column format to this format.
`LifeHistData`	Dataframe with 3 columns: • ID: max. 30 characters long, • Sex: 1 = females, 2 = males, other = unkown, except 4 = hermaphrodite, • BY: (birth or hatching year) Integer, negative numbers are interpreted as missing values. If the species has multiple generations per year, use an integer coding such that the candidate parents' 'Birth year' is at least one smaller than their putative offspring's.
`SeqList`	list with output from a previous run, containing the elements 'Specs', 'AgePriors' and/or 'PedigreePar', as described below, to be used in the current run. If `SeqList$Specs` is provided, all other input parameter values except `MaxSibIter` are ignored.
`MaxSibIter`	number of iterations of sibship clustering, including assignment of grandparents to sibships and avuncular relationships between sibships. Set to 0 to not (yet) perform this step, which is by far the most time consuming and may take several hours for large datasets. Clustering continues until convergence or until MaxSibIter is reached.
`Err`	estimated genotyping error rate. The error model aims to deal with scoring errors typical for SNP arrays.
`MaxMismatch`	maximum number of loci at which candidate parent and offspring are allowed to be opposite homozygotes. Setting a more liberal threshold can improve performance if the error rate is high, at the cost of decreased speed.

Tfilter	threshold log10-likelihood ratio (LLR) between a proposed relationship versus unrelated, to select candidate relatives. Typically a negative value, related to the fact that unconditional likelihoods are calculated during the filtering steps. More negative values may decrease non-assignment, but will increase computational time.
Tassign	minimum LLR required for acceptance of proposed relationship, relative to next most likely relationship. Higher values result in more conservative assignments. Must be zero or positive.
MaxSibshipSize	maximum number of offspring for a single individual (a generous safety margin is advised).
DummyPrefix	character vector of length 2 with prefixes for dummy dams (mothers) and sires (fathers); maximum 20 characters each.
Complex	either "full" (default), "simp" (simplified, no explicit consideration of inbred relationships; not fully implemented yet), "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.
FindMaybeRel	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 (is parent vs. is otherwise related). Time-consuming in large datasets.
quiet	suppress messages.

Details

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

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

Value

A list with some or all of the following components:

AgePriors	Matrix with age-difference based prior probability ratios, used for full pedigree reconstruction.
DummyIDs	Dataframe with pedigree for dummy individuals, as well as their sex, estimated birth year (point estimate, upper and lower bound of 95% confidence interval), number of offspring, and offspring IDs (genotyped offspring only).
DupGenotype	Dataframe, duplicated genotypes (with different IDs, duplicate IDs are not allowed). The specified number of maximum mismatches is used here too. Note that this dataframe may include pairs of closely related individuals, and monozygotic twins.

DupLifeHistID	Dataframe, rownumbers of duplicated IDs in life history dataframe. For convenience only, but may signal a problem. The first entry is used.
ExcludedInd	Individuals in GenoM which were excluded because of a too low genotyping success rate (<50%).
ExcludedSNPs	Column numbers of SNPs in GenoM which were excluded because of a too low genotyping success rate (<10%).
LifeHist	Provided dataframe with sex and birth year data.
MaybeParent	Dataframe with pairs of individuals who are more likely parent-offspring than unrelated, but which could not be phased due to unknown age difference or sex, or for whom LLR did not pass Tassign.
MaybeRel	Dataframe with pairs of individuals who are more likely to be first or second degree relatives than unrelated, but which could not be assigned.
MaybeTrio	Dataframe with non-assigned parent-parent-offspring trios (both parents are of unknown sex), with similar columns as the pedigree
NoLH	Vector, IDs in genotype data for which no life history data is provided.
Pedigree	Dataframe with assigned genotyped and dummy parents from Sibship step; entries for dummy individuals are added at the bottom.
PedigreePar	Dataframe with assigned parents from Parentage step.
Specs	Named vector with parameter values.
TotLikParents	Numeric vector, Total likelihood of the genotype data at initiation and after each iteration during Parentage.
TotLikSib	Numeric vector, Total likelihood of the genotype data at initiation and after each iteration during Sibship clustering.

List elements PedigreePar and Pedigree both have the following columns:

id	Individual ID
dam	Assigned mother, or NA
sire	Assigned father, or NA
LLRdam	Log10-Likelihood Ratio (LLR) of this female being the mother, versus the next most likely relationship between the focal individual and this female (see Details for relationships considered)
LLRsire	idem, for male parent
LLRpair	LLR for the parental pair, versus the next most likely configuration between the three individuals (with one or neither parent assigned)

In addition, PedigreePar has the columns

OHdam	Number of loci at which the offspring and mother are opposite homozygotes
OHsire	idem, for father

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, EstConf, writeSeq, vignette("sequoia")

Examples

```
data(SimGeno_example, LH_HSg5, package="sequoia")
head(SimGeno_example[,1:10])
head(LH_HSg5)
SeqOUT <- sequoia(GenoM = SimGeno_example,
                  LifeHistData = LH_HSg5, MaxSibIter = 0)
names(SeqOUT)
SeqOUT$PedigreePar[34:42, ]
## Not run:
SeqOUT2 <- sequoia(GenoM = SimGeno_example,
                  LifeHistData = LH_HSg5, MaxSibIter = 10)
SeqOUT2$Pedigree[34:42, ]

# reading in data from text files:
GenoM <- as.matrix(read.table("MyGenoData.txt", row.names=1, header=FALSE))
LH <- read.table("MyLifeHistData.txt", header=TRUE)
MySeqOUT <- sequoia(GenoM = GenoM, LifeHistData = LH)

## End(Not run)
```

SimGeno *Simulated genotypes*

Description

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

Usage

```
SimGeno(Ped = NULL, nSnp = 400, ParMis = 0.4, MAF = NULL,
  OutFile = NA, nGen = 20, PropLQ = 0, MisHQ = 0.005,
  MisLQ = 0.3, ErHQ = 5e-04, ErLQ = 0.005, quiet = FALSE)
```

Arguments

Ped	Dataframe, pedigree with columns ID - dam - sire; additional columns are ignored.
nSnp	number of SNPs to simulate.
ParMis	proportion of parents with fully missing genotype.
MAF	(optional) vector with minor allele frequency at each locus. If none specified, allele frequencies will be sampled from a uniform distribution between 0.3 and 0.5.
OutFile	filename for simulated genotypes. If NA (default), return matrix with genotypes within R.
nGen	maximum number of generations to consider (pedigree depth).
PropLQ	proportion of low-quality samples.
MisHQ	average missingness for high-quality samples, assuming a beta-disstribution with alpha = 1.
MisLQ	average missingness in low-quality samples.
ErHQ	error rate in high quality samples (defaults to 0.005).
ErLQ	error rate in low quality samples.
quiet	suppress messages.

Details

Provide either a pedigree dataframe, or the name of a text file containing the pedigree. Please ensure the pedigree is a valid pedigree, for example by first running fixPedigree() from library Pedantics.

Errors are generated by replacing randomly chosen genotypes with random genotypes, with equal probabilities. As this will not result in a change in genotype in around 1/3rd of cases, the number of replaced genotypes is nSnp X n individuals X error rate X 3/2

Value

A matrix with genotype data in sequoia's input format, encoded as 0/1/2/-9.

Disclaimer

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

Author(s)

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

See Also

EstConf

Examples

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

SimGeno_example *Example genotype file*

Description

Simulated genotype data for cohorts 1+2 in Pedigree Ped_HSg5

Usage

```
data(SimGeno_example)
```

Format

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

Author(s)

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

See Also

Ped_HSg5, SimGeno

SnpStats *SNP summary statistics*

Description

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

Usage

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

Arguments

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

Details

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

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

Value

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

`AF`	Allele frequency of the 'second allele' (the one for which the homozygote is coded 2)
`Mis`	Proportion of missing calls
`ER`	(only when Ped provided) number of Mendelian errors in parent- offspring pairs and parent-parent-offspring trios, e.g.parent is AA and offspring is aa.

See Also

`GenoConvert`

---

writeColumns *write data to a file column-wise*

Description

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

Usage

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

Arguments

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

writeSeq *write sequoia output to excel or text files*

Description

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

Usage

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

Arguments

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

Details

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

Examples

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

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

## End(Not run)
```

Index

∗Topic **datasets,**
LH_HSg5, 7
Ped_HSg5, 13
SimGeno_example, 19
∗Topic **sequoia**
LH_HSg5, 7
Ped_HSg5, 13
SimGeno_example, 19

DyadCompare, 2, *12*

EstConf, 3, *17, 18*

FindFamilies, 4, *12*

GenoConvert, 5, *7*, *12*, *14*, *17*, *20*

LH_HSg5, 7, *13*
LHConvert, *6*, 6, *12*

MakeAgeprior, 8
MergeFill, 9

Ped_HSg5, *7*, 13, *19*
PedCompare, *2, 3*, 9, *21*
PedStripFID, *6, 7*, 12

sequoia, *3, 4*, *7, 8*, *12, 13*, 14, *21*
SimGeno, *3*, 17, *19*
SimGeno_example, *13*, 19
SnpStats, 19

writeColumns, 20
writeSeq, *17*, 21