sequoia
Reconstruction of multi-generational pedigrees from SNP data

Jisca Huisman (jisca.huisman@gmail.com)

July 25, 2017

Background `Sequoia` provides a conservative hill-climbing algorithm to construct a high-likelihood pedigree from data on hundreds of single nucleotide polymorphisms (SNPs), described in [1]. Explicit consideration of the likelihoods of alternative relationships before making an assignment reduces the number of false positives, compared to parentage assignment methods relying on the likelihood ratio parent-offspring versus unrelated only [4]. When genetic information is abundant, the heuristic, sequential approach used is considerably quicker than most alternative approaches, with little or no loss in accuracy. Typical computation times are a few minutes for parentage assignment, and a few hours for full pedigree reconstruction when not all individuals are genotyped.

Contents

Example

An example pedigree and associated life history data are provided with the package, which can be used to try out the steps detailed here. This fictional pedigree consists of 5 generations with interconnected half-sib clusters (Pedigree II in [1]).

```
> install.packages("sequoia")  # only required first time
> library(sequoia)             # load the package
> #
> # get the example pedigree and life history data
> data(Ped_HSg5, LH_HSg5)
> tail(Ped_HSg5)
> #
> # simulate genotype data for 200 SNPs
> Geno <- SimGeno(Ped = Ped_HSg5, nSnp = 200)
> #
> # run sequoia - duplicate check & parentage assignment only
> # (maximum number of sibship-clustering iterations = 0)
> ParOUT <- sequoia(GenoM = Geno,
+                   LifeHistData = LH_HSg5,
+                   MaxSibIter = 0)
> names(ParOUT)
> # [1] "Specs"  "AgePriors"  "LifeHist"  "PedigreePar"  "MaybeParent"
> # "TotLikParents"
> #
> # run sequoia - sibship clustering & grandparent assignment
> # use parents assigned above (in 'ParOUT$PedigreePar')
> SeqOUT <- sequoia(GenoM = Geno,
+                   SeqList = ParOUT,
+                   MaxSibIter = 5)
> #
> # compare the assigned real and dummy parents to the true pedigree
> chk <- PedCompare(Ped1 = Ped_HSg5, Ped2 = SeqOUT$Pedigree)
> chk$Counts
> #
> # save results
> save(SeqOUT, file="Sequoia_output_date.RData")
> write.table(SeqOUT$Pedigree, file="Pedigree.txt",
+             sep="\t", row.names=FALSE, quote=FALSE)
>
```

1 Input

1.1 Life history data

The life history data (`LifeHistData`) should be a dataframe with three columns:

- ID: It is probably safest to stick to R's 'syntactically valid names', defined as "consists of letters, numbers and the dot or underline characters and starts with a letter, or the dot not followed by a number" in `?make.names`.
- Sex: 1 = female, 2 = male, other numbers or NA = unknown (except 4 = hermaphrodites [under development, for now possible in parentage assignment only])
- BY: Year of birth/hatching/germination. In species with more than one generation per year, a finer time scale than year of birth ought to be used (in round numbers!), ensuring that parents are born prior to their putative offspring (e.g. parent's BY=2001 and offspring BY=2005, or BY=1 and BY=5 respectively). Negative numbers and NA's are interpreted as unknown.

The column names are ignored, and therefore the order of the columns is critical. Ideally this basic life history information is provided for all genotyped individuals, but this is not necessary. This dataframe may include many more individuals than the genotype data, or in a different order.

1.2 Genotype data

The SNP data should be provided as a numeric matrix `GenoM` with one line per individual, and one column per SNP, with each SNP is coded as 0, 1, 2 copies of the reference allele, or missing (-9). The rownames should be the individual IDs, and column names are ignored.

```
> GenoM <- as.matrix(read.table("MyGenoData.txt",
+                               row.names=1, header=FALSE))
```

The 0/1/2 format can for example be obtained using PLINK (`https://www.cog-genomics.org/plink2`) [3] in combination with `sequoia`'s `GenoConvert()`, as described below. `GenoConvert` can also convert Colony input files.

1.2.1 Real data - Selection of SNP markers

Using tens of thousands of SNP markers for pedigree reconstruction is unnecessary, will slow down computation, and may even hamper inferences by their non-independence. Rather, a subset of SNPs in low linkage disequilibrium (LD) with each other, and with high minor allele frequencies (MAF > 0.3), ought to be selected first if more than a few hundred SNPs are available. The calculations assume independence of markers, and while low (background) levels of LD are unlikely to interfere with pedigree reconstruction, high levels may give spurious results. Markers with a high MAF provide the most information, as although rare allele provide strong evidence when they are inherited, this does not balance out the rarity of such events.

Creating a subset of SNPs can be done conveniently using PLINK, using for example in command prompt (or linux terminal) the command

```
plink --file mydata --maf 0.4 --indep 50 5 2
```

which on a windows machine is equivalent to running inside R

```
> system("cmd", input = "plink --file mydata --maf 0.4 --indep 50 5 2")
```

This will create a list of SNPs with a minor allele frequency of at least 0.4, and which in a window of 50 SNPs, sliding by 5 SNPs per step, have a VIF of maximum 2. VIF, or variance inflation factor, is $1/(1 - r^2)$. It is advised to 'tweak' the parameter values until a set with a few hundred SNPs (300–700) is created. For further details, see `https://www.cog-genomics.org/plink2/ld#indep`.

The resulting list ('plink.prune.in') can be used to create the genotype file used as input for Sequoia, with SNPs codes as 0, 1, 2, or NA, with the command

```
plink --file mydata --extract plink.prune.in --recodeA --out
 inputfile_for_sequoia
```

This will create a file with the extension .RAW, which can be converted to the required input format using

```
> GenoM <- GenoConvert(InFile = "inputfile_for_sequoia.raw")
```

This function can also convert from files in two-columns-per-SNP format, as used by e.g. Colony.

1.2.2 Very large datasets

When the number of individuals is very large, loading the genotype data into R will take up a lot of memory, and may even exceed R's memory limit and be impossible. A stand-alone version of the algorithm underlying this R package does not suffer from this limitation, and is available as Fortran source code from `https://github.com/JiscaH`. This can be compiled under linux, or under windows using Cygwin; it has not been tested on Mac. The input consists of three text files: the lifehistory data; the genotype data with one column for IDs followed by one column per SNP (0/1/2/-9), and no header row; and the parameter settings, for which an example file is included with the code.

1.2.3 Simulating SNP data

When SNP data is not (yet) available, but an approximate pedigree is, it is possible to test `sequoia` on a simulated dataset. This may be useful to for example explore the number of markers required to reliably infer a particular pedigree structure. Alternatively, this can be used to estimated the pedigree-wide error rate of an inferred pedigree (see section 4.2).

The function `SimGeno()` lets the user specify the average proportion of missing genotypes per individual (`MisHQ`), the genotyping error rate (`ErHQ`), and the fraction of known parents (in the supposed 'true' pedigree) which have not been genotyped (`ParMis`). Moreover, the data can be made to contain a fraction of low-quality samples (`PropLQ`, with associated `MisLQ` and `ErLQ`), to assess whether inclusion of samples which did not pass stringent quality control would improve or hamper pedigree reconstruction.

1.3 Parameters

DummyPrefix The prefixes for dummy individuals (sham parental IDs assigned to sibship clusters) can be altered to avoid confusion with IDs of real individuals. Defaults to 'F' for females ('F0001', 'F0002', . . .) and 'M' for males ('M0001', 'M0002', . . .).

Err The genotyping error rate assumed, typically probably around 1E-4 to 1E-3. The error model is given in Table 1; other error structures could easily be implemented but are currently not user-settable.

Table 1: Default probabilities used of observing genotype X, conditional on actual genotype x.

	X		
x	0	1	2
0	1-ϵ	ϵ	0
1	$\epsilon/2$	1-ϵ	$\epsilon/2$
2	0	ϵ	1-ϵ

MaxMismatch The maximum number of loci at which candidate parent and offspring are allowed to be opposite homozygotes, used to filter out highly unlikely pairs. Note that the actual upper limit used is `MaxOH = MaxMismatch + ceiling(Err * nSnp)`.

MaxSibIter The number of iterations of sibship clustering. As this is by far the most time consuming step, and may take several hours for large datasets, it would be wise to first run with `MaxSibIter=0` so that only the much faster parentage assignment is performed, and inspect the output. If during sibship clustering the total likelihood asymptotes before `MaxSibIter` is reached, the algorithm is terminated and the results returned.

MaxSibshipSize Maximum number of offspring for a single individual. A generous safety margin is advised of at least twice the biologically plausible maximum.

Tassign Threshold log10-likelihood ratio (LLR) required for acceptance of a proposed relationship, relative to next most likely relationship. Must be zero or positive, with higher values resulting in more conservative assignments.

Tfilter Threshold LLR between a proposed relationship versus unrelated, to select candidate relatives. Typically negative, and more negative values may prevent filtering out of true relatives, but will increase computational time.

Complexity When it is known that the dataset contains only monogamous matings, the assignment rate can be improved by using the option `Complexity='mono'`. [under development . . .]

1.3.1 Re-use of previous output

The parameter values used as arguments when calling `sequoia` will be returned in the list element `Specs`. These settings can be re-used in a subsequent run, optionally afer changing them

```
> load("Sequoia_output_date.RData")  # if it was saved to disk
> ParOUT$Specs
> #   NumberIndivGenotyped NumberSnps GenotypingErrorRate MaxMismatch
> # 1                  920        200               1e-04           3
> #   Tfilter Tassign nAgeClasses MaxSibshipSize MaxSibIter
> # 1      -2     0.5           6            100          0
> #   DummyPrefixFemale DummyPrefixMale Complexity FindMaybeRel CalcLLR
> # 1                 F               M       full         TRUE    TRUE
> ParOUT$Specs$DummyPrefixFemale <- "D-FEM"
> ParOUT$Specs$DummyPrefixMale <- "D-MALE"
> SeqOUTX <- sequoia(GenoM = Geno,
+                    SeqList = list(Specs = ParOUT$Specs),
+                    MaxSibIter = 10)
```

When `SeqList` is provided and contains an element named `Specs`, all other (default) parameter values are ignored, *except* `MaxSibIter`. It is also possible to re-use the entire output list,

```
> SeqOUT <- sequoia(SeqList = ParOUT)
```

which will use both `AgePriors` and `PedigreePar` in 'ParOUT', as detailed below.

2 Running Sequoia

Under the hood, `sequoia` consists of four sub-programs:

1. **Duplicates:** Check for duplicate entries in the genotype and life history data
2. **Agepriors:** Calculation of age-difference based prior probability ratios
3. **Parentage:** Parentage assignment (assign genotyped parents to genotyped focal individuals)
4. **Sibships:** Clustering of half- and full-siblings, grandparent assignment to singletons and sibships, and identification of avuncular relationships between sibships (jointly referred to as 'Sibships' for brevity)

these all return their output to a single list, with the elements listed in Table 4 and detailed in section 3.

2.1 Check for duplicates

The data may contain positive controls, as well as other intentional and unintentional duplicated samples, with or without life-history information. Sequoia searches the data for (near) identical genotypes, allowing for a `MaxMismatch` mismatches between the genotypes, which may or may not have the same individual ID. Note that very inbred individuals may be nearly indistinguishable from their parent(s), especially when the number of SNPs is limited. Additionally, the genotype and life-history files are checked for duplicate IDs.

It will also return a vector of individuals included in the genotype data, but not in the life history data (`NoLH`). This is merely a service to the user; individuals without life history information can often be successfully included in the pedigree (but not always, see section 3.3).

2.2 Age difference based prior

Based on the species' age at first and last reproduction, some age differences between parent and offspring or between siblings are more likely than others, and some downright impossible. The age differences calculated from the birth years provided in `LifeHistData` are used as a secondary source of information, amongst others to help distinguish between half-siblings, grandparent–grand-offspring and full avuncular pairs.

The list element `AgePriors` contains 8 columns, and as many rows as the birth year range detected in the life history data. It initially only indicates whether a given relationship is biologically possible (1) or not (0) for a given age difference between individuals, for any species (e.g. parents and their offspring can never be exactly the same age). The first row is for individuals born in the same year, the second row for individuals born one year apart, etc. The columns are labelled for various relationship categories, with M = mother, P = father, MS = maternal sibling, PS = paternal sibling, MGM = maternal grandmother, PGF = paternal grandfather, MGF = maternal grandfather and paternal grandmother, and AU = avuncular (niece/nephew – aunt/uncle).

For example, the first value in the column 'MS' can be interpreted as 'if I were to pick two individuals born in the same year, and two individuals from my sample at random, how much more likely are the first pair to be maternal siblings, compared to the second pair?' Or to phrase it differently: 'Now that I learned that these individuals are born in the same year, does that make them more likely or less likely to be maternal siblings than before I knew this?' Values below 1 indicate less likely, and values above 1 more likely. For MS, PS and AU absolute age differences are used (with overlapping generations, nephews may be older than their aunts), while parents and grandparents are necessarily older than their (grand-)offspring (categories M, P, MGM, PGF and MGF).

These age-difference based priors are by default automatically updated after parent-age assignment, based on the empirical distribution of age differences between individuals and their assigned fathers and mothers. This update is prevented when `SeqList` is provided and contains an element `AgePriors` (see Table 2).

Table 2: Behaviour when 'AgePriors' and/or 'PedigreePar' are provided in 'SeqList'. –: not provided / not run; age prior categories are 'user'= user-provided, 'basic' = minimal restrictions, 'parents' = based on assigned parents

in SeqList		Age prior used	
AgePriors	PedigreePar	Parentage	Sibships
–	–	basic	parents
user	–	user	parents
–	Y	–	parents
user	Y	–	user

`AgePriors` can be altered to match the biological characteristics of the species, but the number of rows must not be decreased, and the column order kept as it is. If the number of rows is increased, `Specs['nAgeClasses']` should be updated to match the new number of rows.

Table 3: Example age-difference prior, for non-overlapping generations

MS	PS	MGM	PGF	MGF	UA	M	P
1	1	0	0	0	0	0	0
0	0	0	0	0	1	1	1
0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

For example, for a species with strictly non-overlapping generations, one may wish to alter `AgePriors` to the matrix in Table 3, which can be done as follows

```
> AP <- as.matrix(SeqOUT1$AgePriors)
> AP[AP>0] <- 0
> AP[1,c("MS", "PS")] <- 1
> AP[2,c("M", "P", "UA")] <- 1
> AP[3,c("MGM", "PGF", "MGF")] <- 1
> SeqOUT2 <- sequoia(SeqList=list(Specs=SeqOUT1$Specs, AgePriors=AP),
+                     MaxSibIter = 0)
```

Note that any identified parent-offspring pairs which are not exactly 1 year / time unit apart will be returned in `MaybeParent` (section 3.3). It is possible to enforce the same age-difference prior on the sisbhip clustering as well, but only if parentage assignment and sibship clustering are run separately (see Table 2)

2.3 Parentage assignment

Assignment of genotyped parents to genotyped offspring is performed by default, unless earlier-assigned parents are provided in `SeqList$PedigreePar`.

The number of pairs to be checked if they are parent and offspring is very large for even moderate numbers of individuals, e.g. 5 000 pairs for 100 individuals, and 2 million for 2 000 individuals. Therefore, three 'sieves' are applied sequentially to find candidate parent-offspring pairs, with decreasing 'mesh size'

- The number of SNPs at which the pair are opposing homozygotes must be less than or equal to the per-SNP genotyping error rate `Err` times the number of SNPs (rounded up to nearest whole number), plus the safety margin `MaxMismatch`,
- The likelihood ratio between being parent and offspring versus unrelated, not conditioning on any already assigned parents, must be equal to or greater than `Tfilter`,
- The likelihood ratio between the pair being parent and offspring versus being otherwise related must be equal to or greater than `Tassign`, to filters out siblings, grandparents and aunts/uncles,

and the older of the pair is assigned as parent of the younger. If it is unclear which is the older, or if it is unclear whether the parent is the mother or the father, the pair is returned in `MaybeParent` (section 3.3). If there are multiple candidate parents of the same sex, or some of unknown sex, the parent pair or single parent resulting in the highest likelihood is assigned. This heuristic sequential filtering approach makes parentage assignment quick, and for example takes less than a minute for an empirical dataset with 2 500 genotyped individuals on a laptop with an intel i7 2.3 GHz CPU and 8GB RAM

2.4 Sibship clustering & the rest

Full pedigree reconstruction, including sibship clustering amongst those individuals which have not been assigned two genotyped parents, is performed when `MaxSibIter`> 0. This may take from a few seconds to several hours, depending on the number of individuals without a parent, the number of sibships that is being clustered, and their degree of interconnection. During this phase, all first and second degree links between individuals are attempted to be assigned, using the following steps in each iteration

- Find pairs of full- and half-siblings
- Cluster sibling pairs into sibships
- Merge existing sibships
- Replace dummy parents by genotyped individuals (round 2+)
- Add lone individuals to sibships (round 2+)
- Assign genotyped parents to genotyped individuals
- Find grandparent – grandoffspring pairs (round 3+)
- Assign grandparents to sibships (round 2+; grandparents may be dummy individuals as well as genotyped individuals)

The total likelihood (section 3.4) typically asymptotes within five to ten iterations, even for complex pedigrees. When an asymptote is reached before `MaxSibIter`, a final iteration with stronger dependence on the age prior is ran, parental likelihoods are calculated, a check is done for non-assigned potential relatives, and the algorithm is terminated. These last steps may take considerable time, and either or both can be skipped by specifying `CalcLLR` = FALSE and/or `FindMaybeRel` = FALSE.

3 Output

Beside the inferred pedigree (section 3.1), `sequoia` also returns summary information of the dummy parents (section 3.2), any pairs of individuals which are likely to be relatives but could not be assigned as such (section 3.3), the total likelihood of the data after each iteration (section 3.4), and the input data and parameters (except the large genotype data) (full overview in Table 4).

3.1 PedigreePar & Pedigree

`PedigreePar` is the scaffold pedigree returned after assigning genotyped parents to genotyped offspring. `Pedigree` additionally includes dummy individuals, assigned to infered groups of half-siblings for which the shared parent is not genotyped. Note that dummy individuals are also assigned as the 'in-between' individual of identified grandparent – grand-offspring pairs. Dummy individuals are appended at the bottom of the pedigree with their assigned parents, i.e. the sibship's assigned grandparents, and by default have IDs 'F0001', 'F0002', . . . for dams and 'M0001', 'M0002', . . . for sires (sections 1.3 and 3.2).

Table 4: Output from Sequoia, returned within a named list.

Output	Description
AgePriors	Age-difference based prior probabilities
DummyIDs	Details per half-sib cluster
DupGenoID	Duplicated IDs in genotype data
DupGenotype	(near) Duplicated genotypes
DupLifeHistID	Duplicated IDs in life history data
LifeHist	sex and birth year data
MaybeParent	Non-assigned likely PO pairs
MaybeRel	Non-assigned likely relatives
NoLH	IDs in genotype data not present in life history data
Pedigree	Pedigree
PedigreePar	Scaffold pedigree
Specs	Parameter values
TotLikParents	Total likelihood during parentage
TotLikSib	Total likelihood during sib clustering

The pedigrees columns are

- IDs of the individual, its assigned dam (mother) and sire (father),
- The log10 likelihood ratio (LLR) of the dam, sire and the parent pair; this is the ratio between the likelihood of the assigned parent being the parent, versus the most likely alternative type of being related to the focal individual (see Table 5),
- The number of loci at which the offspring and the assigned dam or sire are opposite homozygotes (`PedigreePar` only).

The parental LLRs are calculated at the very end, and are conditional on all other links in the reconstructed pedigree. The parent-pair LLR is relative to the most likely assignment of a single parent (or no parent). Note that this LLR differs from for example Cervus [2], which returns the natural log of the ratio between the probability that the assigned parent is the parent, versus that the next most likely candidate is the parent.

Some parents may have a very small or even negative single-parent LLR, but the LLR of the parent pair should ideally always be positive. For full sibling pairs and dummy-parents of dummy-individuals this is not always the case, due to some approximations used when calculating the parental LLR (which are not used during the assignment steps). It is however probably worthwhile to be cautious about assignments with low or negative LLRs, and for example compare with a previous pedigree (section 4.1) or the genomic relatedness (section 4.3).

If some of the LLRs are very large negative or positive numbers, please send a bug report to `jisca.huisman@gmail.com` with a short description of your dataset — something probably went wrong.

Table 5: Pairwise relationships considered.

PO	Parent-offspring
FS	Full siblings
HS	Half siblings
GP	Grandparent – grand-offspring
FA	Full aunt/uncle – niece/nephew
HA	Half aunt/uncle – niece/nephew, or other 3rd degree relative
U	Unrelated

3.2 DummyIDs

To each cluster of half-siblings a 'dummy' parent is assigned, denoted by increasing numbers, by default with prefix 'F' for females and 'M' for males (sections 1.3). `DummyIDs` is a dataframe with for each dummy individual

- the assigned dam and sire (the sibship's grandparent) and their associated LLRs, which can also be found in `Pedigree`
- its sex
- the estimated birth year, as a point estimate ('BY.est') and lower and upper bound of 95% probability interval ('BY.min' and 'BY.max'). These are based on the birthyears of the individuals in the sibship and of the sibship-grandparents, if any, in combination with `AgePriors`. This may help
- 'NumOff', the number of individuals in the sibship (= the dummy individuals number of offspring)
- the IDs of the individuals in the sibship, with column names 'O1', 'O2', . . .

This information is intended to make it easier to associate dummy IDs to real IDs of observed but non-genotyped individuals (see also section 4.1).

3.3 MaybeParent & MaybeRel

`MaybeParent` countains probable or definite parent-offspring pairs which could not be assigned in `PedigreePar`, with columns

- ID1, ID2: identities of the pair
- Sex1, Sex2: sex of the individuals; 1=female, 2=male
- AgeDif: Age difference, positive numbers indicate that ID2 is older
- TopRel: Relationship with the highest likelihood, may be any of the abbreviations in Table 5, or '2nd' (undetermined type of second degree relative, see text). 'XX' indicates unclear, but more likely to be first or second degree relatives than unrelated.

- LLR: Log10 likelihood ratio (LLR) between the pair being related according to the most likely relationship (column 'TopRel') versus the next most likely relationship.
- OH: The number of loci at which the individuals are opposite homozygotes.

This dataframe includes cases where the pair is more likely to be parent-offspring than unrelated, but where it cannot be excluded that they are otherwise related ('LLR' between most likely and next most likely < `Tassign`), or were an alternative relationship is even more likely ('TopRel' not PO). Additionally, `MaybeParent` may include pairs which are most likely to be parent and offspring, but where lack of birth year information made it impossible to tell which of the two was the parent and which was the offspring ('AgeDif' = NA), or where lack of sex information of the older one made it impossible to tell whether this candidate parent is the mother or the father ('Sex2' = 3).

`MaybeRel` includes pairs which are more likely to be first or second degree relatives than unrelated, but which could not be assigned in `Pedigree`. This includes for example half siblings where it is unclear whether they share a mother or a father. Distinguishing half siblings from grandparent–grand-offspring and full avuncular pairs is not straight forward either, and relies on either both individuals already having at least one parent assigned, or very strong support based on the age diference of the pair. When neither is the case, 'TopRel' indicates '2nd', and LLR is between being 2nd degree relatives versus the most likely of PO, FS, HA or U.

Note that the *most likely* relationship is not necessarily the *true* relationship between a pair, due to the random nature of Mendelian segregation, and possible genotyping errors. In addition, the most likely relationship for a pair will not necessarily result in the highest global likelihood (and may therefore have not been assigned).

3.4 TotLikParents & TotLikSib

These are vectors with the log10 of the approximate total likelihood of the pedigree, which is the probability of observing the genotype data, given the reconstructed pedigree, the allele frequencies of the SNPs, and the presumed genotyping error rate. The value at initiation (the first value in `TotLikParents`) is calculated assuming Hardy-Weinberg equilibrium in the sample. The subsequent value are at the end of each iteration of parentage assignment (`TotLikParents`) or sibshib clustering (`TotLikSib`, should be increasing across iterations, and asymptoting. If there is a large change in value between the second-last and last likelihood, consider running the algorithm for more iterations (increase `MaxSibIter`). One can do a visual check as follows:

```
> TLL <- c(SeqOUT$TotLikParents, SeqOUT$TotLikSib)
> xv <- c(paste("p", 1:length(SeqOUT$TotLikParents)-1),
+         paste("s", 1:length(SeqOUT$TotLikSib)-1))
> plot(TLL, type="b", xaxt="n", xlab="Round")
> axis(1, at=1:length(TLL), labels=xv)
```

The total likelihood is calculated assuming independent SNPs as

$$\mathcal{L} = \prod_{A=1}^{N} \prod_{l}^{L} \sum_{y} \sum_{z} P(A_l = X | DA_l = y, SA_l = z, \epsilon) P(DA_l = y) P(SA_l = z) \tag{1}$$

or the probability of observing individual A's genotype X at SNP l, given the true genotypes y and z of it assigned parents DA and SA, multiplied over all individuals and all SNPS. For example, if X is a heterozygote, the probability of this genotype is 1/2 if y is heterozygous and z a homozygote, 1 if y and z are opposite homozygotes, and 0 (or $\epsilon/2$ when allowing genotyping errors, Table 1) if y and z are identical homozygotes. This is summed over all possible parental genotypes, weighed by the probabilities that the parent have true genotype y and z. These probabilities are determined by the parent's observed genotypes and the genotyping error rate for genotyped parents, or according to Hardy-Weinberg proportions for non-assigned parents. For dummy parents, the probability depends on A's siblings and grandparents (see [1]).

4 Output check

4.1 Comparison with previous pedigree

Often times, a (part) pedigree is already available to which one wants to compare the results, for example consisting of maternal links, deduced from observations in the field. The function `PedCompare()` performs such comparisons, and takes as arguments the 'true' pedigree as `Ped1`, and the newly inferred pedigree as `Ped2`:

```
> compareOUT <- PedCompare(Ped1 = Ped_HSg5, Ped2 = SeqOUT$Pedigree)
```

Where the output list consists of `Counts`, a summary of the number of matches and mismatches between the two pedigrees, as well as `MergedPed`, a side-by-side comparison, and `ConsensusPed`, an amalgamation of the two. `PedCompare()` does its best to align any dummy parents in the inferred pedigree, to non-genotyped individuals in the other pedigree.

Counts An array printed as two 7x5 matrices, one for dams and one for sires. When checking the results from parentage assignment only, only the rows 'GG' (Genotyped focal - Genotyped parent) are relevant:

```
> compareOUT2 <- PedCompare(Ped1 = Ped_HSg5, Ped2 = ParOUT$Pedigree)
> compareOUT2$Counts["GG",,]
> #          dam sire
> # Total    130  170
> # Match    128  166
> # Mismatch   0    0
> # P1only     2    4
> # P2only     0    0
```

Further details, amongst others on what counts as a 'Match' versus 'Mismatch' in the case of dummy parents is provided in the help file (`?PedCompare`).

MergedPed This side-by-side comparison of the two pedigrees allows one to inspect any mismatches and discrepancies between the two pedigrees. In addition to the parents in Ped1 ('dam.1' and 'sire.1') and Ped2 ('dam.2' and 'sire.2'), it includes three columns ('id.r', 'dam.r', and 'sire.r') where dummy IDs in Pedigree 2 are replaced by the most likely non-genotyped individual from Pedigree 1. The value 'nomatch' in these columns indicates that there is no no-genotyped individual for which more than half of its offspring according to Ped1 has been assigned this dummy in Ped2. Note that this does include cases where a true sibship of say five individuals was split into one of three and one of two; the one of three is considered a match, and the smaller a mismatch — even though it can be argued the pedigree does not contain any incorrect links.

ConcensusPed Here the merged pedigree is collapsed, with Pedigree 2 (here `Sequoia` assignments) taking priority over Pedigree 1, and dummy parents being replaced where known (using 'id.r', 'dam.r', and 'sire.r'). The columns 'dam.cat' and 'sire.cat' indicate with a 2-letter code whether the focal individual and the assigned parent were genotyped (G), a dummy individual in Pedigree 2 (D), a dummy individual replaced by a best-match non-genotyped individual from Pedigree 1 (R) or ungenotyped (U, and thus taken from Pedigree 1 only).

Example To increase the chance of mismatches, we simulate a genotype dataset with few SNPs and a high genotyping error rate. The specific numbers will differ between simulated datasets, but the general pattern will be the same

```
> data(LH_HSg5, Ped_HSg5)
> GM <- SimGeno(Ped = Ped_HSg5, nSnp = 100, ErHQ = 1e-3)
> SeqX <- sequoia(GenoM = GM,  LifeHistData = LH_HSg5, MaxSibIter = 5)
> comp <- PedCompare(Ped1 = Ped_HSg5, Ped2 = SeqX$Pedigree)
> comp$Counts
> # , , dam
> #     Total Match Mismatch P1only P2only
> # GG    561   550        6      5      0
> # GD    334   323        5      3      0
> # GT    890   873        9      8      0
> # DG     42    39        1      2      0
> # DD     28    28        0      0      0
> # DT     70    67        1      2      0
> # TT    964   940       10     10      4
> #
> # , , sire
> #     Total Match Mismatch P1only P2only
> # GG    502   498        3      1      0
> # GD    391   382        4      2      0
> # GT    890   880        7      3      0
> # DG     41    39        0      2      0
> # DD     29    28        0      1      0
> # DT     70    67        0      3      0
> # TT    965   947        7      6      5
```

We can investigate the mismatches further:

```
> with(comp$MergedPed,
+      comp$MergedPed[which(dam.1!=dam.2 & dam.1!=dam.r), ])
> #         id  dam.1 sire.1 dam.2 sire.2 id.r   dam.r sire.r
> # 205 b02032 a01166 b01133 F0045 b01165 <NA> nomatch   <NA>
> # 206 a02031 a01166 b01133 F0045 b01165 <NA> nomatch   <NA>
> # 720 b03011 a02106 b02181 F0047  M0020 <NA> nomatch b02181
> # 835 b04097 a03098 b03079 F0048  M0032 <NA> nomatch b03079
> # 897 a02030 a01166 b01133 F0045   <NA> <NA> nomatch   <NA>
```

First, let's have alook at dummy mother F0045, mentioned three times – is this one sibship according to Pedigree1 split into two, or two sibships wrongly merged into one?

```
> comp$MergedPed[which(comp$MergedPed$dam.2=="F0045"), ]
> # -> only a02030, a02031 and b02032 are in this half-sibship
> # -> not wrongly merged
> comp$MergedPed[which(comp$MergedPed$dam.1=="a01166"), ]
> # -> true offspring of a01166 split over dummy mothers
> # F0027 and F0045
```

Second, let's investigate dummy mother F0048 further:

```
> comp$MergedPed[which(comp$MergedPed$dam.2=="F0048"), ]
> # -> b04097 is sole individual in this sibship
> SeqX$MaybeParent[SeqX$MaybeParent$ID1=="b04097", ]
> #        ID1    ID2 Sex1 Sex2 AgeDif TopRel  LLR OH
> # 229 b04097 a03098    2    1      1     GP 0.29  1
> # 310 b04097 b04098    2    2      0     FA 1.17  1
> # -> dam in pedigree1 (a03098) more likely to be PO than U
> # (otherwise not in MaybeParent), but even more likely GP
> # Genotyping errors resulted in pair being OH at 1 SNP.
>
> comp$MergedPed[which(comp$MergedPed$id=="F0048"), ]
> # -> parents of F0048 = grandparents of b04097: a02106 and b02158
> comp$MergedPed[which(comp$MergedPed$id=="a03098"), ]
> #         id  dam.1 sire.1  dam.2 sire.2 id.r dam.r sire.r
> # 288 a03098 a02106 b02158 a02106 b02158 <NA>  <NA>   <NA>
> # -> correct grandparents assigned, even if mother not assigned.
>
> SeqX$MaybeRel[SeqX$MaybeRel$ID1=="b04097", ]
> #        ID1    ID2 Sex1 Sex2 AgeDif TopRel  LLR
> # 124 b04097 a03098    2    1      1     PO 1.31
```

It seems that when taking into account b04097's assigned grandparents in `Pedigree`, the relationship with true mother a03098 is now most likely to be PO, rather than GP. When running another two iteration of sibship clustering, dummy F0048 is indeed replaced by a03098, and the total number of mismatches between the pedigrees drops from 10+7 to 9+5.

Colony To compare Colony output with an existing pedigree, use:

```
> BestConfig <- read.table("Colony/file/file.BestConfig",
+                          header=T, sep="", comment.char="")
> PedCompare(PedFile1 = "ExistingPedigree.txt",
+            Ped2 = BestConfig)
```

4.2 Estimating confidence probabilities

The provided likelihood ratio between the assigned parent being the parent versus otherwise related to the focal individual, does not necessarily indicate how likely it is that the assignment is correct. Pedigree-wide confidence probabilities can, amongst others, be estimated by

- simulating genotype data according to the reconstructed (or an existing) pedigree, imposing realistic levels of missingness and genotyping errors;
- reconstructing a pedigree from these simulated data;
- counting the number of mismatches between the 'true' pedigree, used as input for the simulated data, and the pedigree reconstructed from the simulated data.

When repeated at least 10–20 times, the mean error count divided by the total number of pedigree links provides an estimate of one minus the the confidence probability. Note that this can be rather time consuming, and will give an anti-conservative estimate as the current simulations assume all SNPs are independent.

An example script:

```
> data(LH_HSg5, Ped_HSg5)
> ARER <- array(dim=c(10,7,2))
> for (x in 1:10) {
+   cat(x, "\t", format(Sys.time(), "%H:%M:%S"), "\n")
+   GM <- SimGeno(Ped = Ped_HSg5, nSnp = 100)
+   SimOUT <- sequoia(GenoM = GM,  LifeHistData = LH_HSg5, quiet=TRUE)
+   CountX <- PedCompare(Ped1 = Ped_HSg5, Ped2 = SimOUT$Pedigree)$Counts
+   ARER[x,,1] <- rowMeans(CountX[, "Match", ] / CountX[, "Total", ])
+   ARER[x,,2] <- rowMeans((CountX[, "Mismatch", ] +
+                           CountX[, "P2only", ]) / CountX[, "Total", ])
+ }
> dimnames(ARER) <- list(c(1:10), dimnames(CountX)[[1]], c("AR", "ER"))
> #
> # average assignment rate (AR) & error rate (ER) per category:
> round(apply(ARER, c(2:3), mean), 4)
```

To add confidence probability to the pedigree based on real data

```
> ConfProb <- 1 - round(colMeans(ARER[, , "ER"]),3)
> #
> PedC <- PedCompare(Ped1 = Ped_HSg5,
+                    Ped2 = SeqOUT$Pedigree)$ConsensusPed
> PedC$dam.prob <- ConfProb[as.character(PedC$dam.cat)]
> PedC$sire.prob <- ConfProb[as.character(PedC$sire.cat)]
> #
> # or, when assuming that replacement of dummies by IDs of
> # non-genotyped individuals is free from error,
> PedC$dam.prob <- with(PedC,
+                       ifelse(dam.cat=="GG", ConfProb["GG"],
+                         ifelse(dam.cat %in% c("GD", "GR"), ConfProb["GD"],
+                           ifelse(dam.cat %in% c("DG", "RG"), ConfProb["DG"],
+                             ifelse(dam.cat %in% c("DD", "DR", "RD", "RR"),
+                                    ConfProb["DD"], NA)))))
```

4.3 Comparison pedigree-based and genomic relatedness

In absence of a previous pedigree, or when it is not obvious whether the previous or newly inferred pedigree is correct, one can compare the pairwise relatedness estimated from the pedigrees to a measure of genomic relatedness, estimated directly from the complete SNP data – which may be many more SNPs than used for pedigree reconstruction. Genomic relatedness can be estimated for example using GCTA, `http://cnsgenomics.com/software/gcta/estimate_grm.html`, while pedigree relatedness can be calculated for example using the R package `pedantics`. Genomic relatedness will vary around the pedigree-based relatedness even for a perfect pedigree due to Mendelian variance, but outliers suggest pedigree errors.

As the number of pairs p becomes very large even for moderate numbers of individuals n ($p = n \times (n-1)/2$), additional packages are required to assist with merging (`data.table`) and plotting (`hexbinplot`). For example:

```
> Rel.snp <- read.table("GT.grm.gz")
> Rel.id <- read.table("GT.grm.id", stringsAsFactors=FALSE)
> Rel.snp[,1] <- as.character(factor(Rel.snp[,1], labels=Rel.id[,2]))
> Rel.snp[,2] <- as.character(factor(Rel.snp[,2], labels=Rel.id[,2]))
> names(Rel.snp) <- c("IID2", "IID1", "SNPS", "R.SNP")
> Rel.snp <- Rel.snp[Rel.snp$IID1 != Rel.snp$IID2,]
> #
> library(pedantics)
> PedStats <- pedigreeStats(SeqOUT$Pedigree[,1:3], graphicalReport=FALSE,
+                           includeA=TRUE)
> Rel.ped <- as.data.frame.table(PedStats$Amatrix)
> names(Rel.ped) <- c("IID1", "IID2", "R.seq")
> #
> library(data.table)
> Rel.snp <- data.table(Rel.snp, key=c("IID1", "IID2"))
> Rel.ped <- data.table(Rel.ped, key=c("IID1", "IID2"))
> Rel.gt <- merge(Rel.snp[,c(1,2,4)], Rel.ped, all.x=TRUE)
> Rel.gt <- as.data.frame(Rel.gt)
> rm(PedStats, Rel.snp, Rel.ped)
> #
> round(cor(Rel.gt[, 3:4], use="pairwise.complete"),4)
> #
> library(hexbin)
> ColF <- function(n) rev(rainbow(n, start=0, end=4/6,
+                              s=seq(.9,.6,length.out=n),v=.8))
> hexbinplot(Rel.gt$R.SNP~Rel.gt$R.ped, xbins=100, aspect=1, maxcnt=10^6.5,
+           trans=log10,inv=function(x) 10^x, colorcut=seq(0,1,length=14),
+           xlab="Pedigree relatedness", ylab="Genomic relatedness",
+           xlim=c(-.1,.9), ylim=c(-.1, .9), colramp=ColF, colorkey = TRUE)
```

5 Other

5.1 Unusual relationships

Pedigree inference is often applied in small, (semi-)closed populations, and regularly to test for inbreeding. In such cases, pairs of individuals may be related via more than one route. For example, maternal half-siblings may also be niece and aunt via the paternal side, and be mistaken for full-siblings. A range of such double relationships is considered explicitly (Table 6) to minimise such mistakes. If such a type is common in your population but not yet considered by `sequoia`, and seems to be causing problems, please send an email to `jisca.huisman@gmail.com` as adding additional relationships is relatively straightforward.

Table 6: Double relationships between pairs of individuals; – = impossible, Y = explicitly considered, empty = not (yet) explicitly considered (but possible to be inferred in two steps). Abbreviations as before, and GGG=great-grandparent, F1C=full first cousins, H1C=half first cousins (parents are HS).

	PO	FS	HS	GP	FA	HA	GGG	F1C	H1C	U
PO	–	–	Y	Y						Y
FS	–	–	–	–	–	Y		–	Y	Y
HS	Y	–	(FS)	Y		Y[2]				Y
GP	Y	–	Y	[1]						Y
FA		–								Y
HA		Y	Y[2]							Y
F1C										Y
GGG		–					[3]			Y

1: Can not be considered explicitly, as likelihood identical to PO
2: Including the special case were one is inbred
3: Can not be considered explicitly, as likelihood identical to GP

5.2 Hermaphrodites

For *Parentage assignment only*, hermaphrodites can be specified in `LifeHistData` with sex '4', which then 'under the hood' are threated as two individuals with opposite sex, and identical genotypes. When running as usual, and all candidate parents are hermaphrodites, no parents will be assigned at all, as the configuration where A is the dam and B the sire is as likely as B being the dam and A the sire. Likely parent-offspring pairs can be found in 'MaybeParent', but note that this is truncated at 7 times the number of genotyped individuals.

To chose between A being the dam or the sire, additional information is required on the probable dam. This can e.g. the plant from which the seed was collected. This 'prior' information is not (yet) fully integrated, but rather, only when two configurations are equally likely, is the one that matches the pedigree prior chosen. Parent–offspring pairs in the pedigree prior that are not genetically a match will never be assigned.

```
> cdam <- read.table("Candidate_dams.txt", header=TRUE, stringsAsFactors=FALSE)
> # cdam has columns 'id' and 'dam', and does not need entries for all ids
> seq.euc2 <- sequoia(GenoM = Geno, LifeHistData = LH_X, MaxSibIter=0,
+                     SeqList = list(PedigreePar = cbind(cdam, sire=NA)))
```

References

[1] J Huisman. Pedigree reconstruction using snp data: parentage assignment, sibship clustering, and beyond. *Molecular Ecology Resources*, accepted. DOI: 10.1111/1755-0998.12665

[2] T C Marshall, J B K E Slate, L E B Kruuk, and J M Pemberton. Statistical confidence for likelihood-based paternity inference in natural populations. *Molecular ecology*, 7(5):639–655, 1998.

[3] S Purcell, B Neale, K Todd-Brown, L Thomas, M A R Ferreira, D Bender, J Maller, P Sklar, PIW De Bakker, MJ Daly, et al. PLINK: a tool set for whole-genome association and population-based linkage analyses. *The American Journal of Human Genetics*, 81(3):559–575, 2007.

[4] E A Thompson and T R Meagher. Parental and sib likelihoods in genealogy reconstruction. *Biometrics*, pages 585–600, 1987.