sequoia

Reconstruction of multi-generational pedigrees from SNP data

Jisca Huisman (jisca.huisman @ gmail.com)

June 20, 2019

Contents

0.1 Quick-start example 1: Simulated data

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")
> writeSeq(SeqList = SeqOUT, GenoM = Geno, PedComp = chk,
+           folder = "Sequoia-OUT")
```

0.2 Quick-start example 2: Real data

First get a subset of SNPs which are as informative (high MAF, high call rate), reliable (low error rate), and independent (low LD) as possible. Ideally around 500 – 800 for full pedigree reconstruction, but the necessary number also depends on the mating structure (e.g. level of polygamy and inbreeding). In addition you need a dataframe with the sex and birth/hatching year of as many individuals as possible, in arbitrary order.

```
> install.packages("sequoia")  # only required first time
> library(sequoia)             # load the package
> #
> # save the genotype data with 1 row per individual, 1 column per SNP
> # (0/1/2/NA), e.g. in PLINK:
> # plink --file mydata --geno 0.1 --maf 0.3 --indep 50 5 2
> # plink --file mydata --extract plink.prune.in --out
> #
> # read in genotype data
> # if already coded as 0/1/2, with missing=-9:
> Geno <- as.matrix(read.csv("mydata.csv", header=FALSE, row.names=1))
> # for many other  input formats:
> Geno <- GenoConvert(InFile = "mydata.ped", InFormat="ped")
> #
> # read in lifehistory data: order ID-Sex-birthyear, column names ignored
> LH <- read.table("LifeHistoryData.txt", header=T)
> #
> # duplicate check & parentage assignment (takes few minutes)
> # (maximum number of sibship-clustering iterations = 0)
> # if genotyping error rate is unknown, start of high
> ParOUT <- sequoia(GenoM = Geno,  LifeHistData = LH_HSg5,
+                   MaxSibIter = 0, Err=0.01, MaxMismatch=10)
> #
> # inspect duplicates (intentional or accidental)
> ParOUT$DupGenotype
> # (...)
> #
> # check if distr. of age-differences for each relative type is sensible
> PlotAgePrior(ParOUT$AgePriors)
> #
> # compare assigned parents to field pedigree (check column order!)
> FieldPed <- read.table("FieldPed.txt", header=T)
> PC.par <- PedCompare(Ped1 = FieldPed[, c("id", "dam", "sire")],
+             Ped2 = ParOUT$PedigreePar)
> PC.par$Counts["TT",,]
> # (...)
> #
> # calculate Mendelian errors per SNP (works also w field pedigree)
> stats <- SnpStats(Geno, ParOUT$PedigreePar)
```

```
> #
> # polish dataset: remove one indiv. from each duplicate pair
> # (1st one, or one w lowest call rate) & drop high error rate SNPs
> Geno2 <- Geno[!rownames(Geno) %in% ParOUT$DupGenotype$ID2, ]
> Geno2 <- Geno2[, -which(stats[,"ER"]>50)]
> # (check histogram for sensible threshold)
> #
> # iterate the above as necessary
> #
> # run full pedigree reconstruction (may take few hours)
> SeqOUT <- sequoia(GenoM = Geno2,
+                   MaxSibIter = 20,
+                   Err=0.001)
> # inspect no. assigned parents, proportion dummy parents, etc.
> SummarySeq(SeqOUT)
> # (see Example 1 for saving results)
```

0.3 Background

The core of `Sequoia` is to

- Assign genotyped parents to genotyped individuals ('parentage assignment'), even if the sex or birth year of some candidate parents is unknown;
- Cluster genotyped half- and full-siblings for which the parent is not genotyped into sibships, assigning a 'dummy parent' to each sibship
- Find grandparents to each sibship, both among genotyped individuals and among dummy parents to other sibships.

`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 (parent-offspring, full siblings, grandparent-grandoffspring, ...) before making an assignment reduces the number of false positives, compared to parentage assignment methods that rely on the likelihood ratio between parent-offspring versus unrelated only [4]. The heuristic, sequential approach used is considerably quicker than most alternative approaches such as MCMC, and when genetic information is abundant there is little to 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.

A word of caution: 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 not have been assigned.

1 Input

1.1 Life history data

The life history data (`LifeHistData`) should be a dataframe with three columns (column names are ignored, order is important!):

- 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, 3=unknown, 4=hermaphrodites, all other numbers, letters, or NA = unknown
- BirthYear: 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). Negative numbers and NA's are interpreted as unknown.

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

When the genotype data is currently in another format, such as Colony input files or PLINK's .ped or .raw files, `sequoia`'s `GenoConvert()` can be used, as described below.

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 with a decent genotyping call rate (e.g. > 0.9), in low linkage disequilibrium (LD) with each other, and with high minor allele frequencies (e.g. 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 --geno 0.1 --maf 0.3 --indep 50 5 2
```

which on a windows machine is equivalent to running inside R

```
> system("cmd", input = "plink --file mydata --maf 0.3 --indep 50 5 2")
```

This will create a list of SNPs with a missingness below 0.1, a minor allele frequency of at least 0.3, 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 $1/(1 - r^2)$. For further details, see `https://www.cog-genomics.org/plink2/ld#indep`.

It is advised to 'tweak' the parameter values until a set with a few hundred SNPs (300–700) is created. To assist with this, the function `SnpStats` gives for each SNP both the allele frequency and the missingness. In addition, when a pedigree is provided (e.g. an existing one, or from a preliminary parentage-only run), the number of Mendelian errors per SNP is calculated.

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 Exclusion of low call rate samples & SNPs

Samples with a very low genotyping succes rate (call rate) can sometimes wrongly be assigned as parents to unrelated individuals, as `sequoia` does not (yet) deal perfectly with

these cases. In addition, at least in my experience with SNP arrays, a low sample call rate is often indicative of poor sample quality or a poor genotyping run, and associated with a high sample error rate. Therefore, samples with a call rate below 0.5 are automatically excluded; their sample IDs are returned in the list element `ExcludedInd` (see 3 for other list elements). A stricter threshold (e.g. 0.8) is advised, and can most easily be done in PLINK using the option `--mind 0.2`.

In addition, SNPs with a call rate below 0.1 are excluded (listed in `ExcludedSNPs`, if any), as these contribute almost no information. Again, a stricter threshold is advised, and can most easily be done in PLINK (see above).

Checks for individuals and SNPs with low call rate, and monomorphic SNPs, are done automatically when calling `sequoia`, but can be done separately in advance by calling `CheckGeno`.

1.2.3 Family IDs

By default, the 'Family ID' (1st) column in the PLINK file is ignored, and IDs are extracted from the second column only. If the family IDs are essential to distinguish between individuals, use GenoConvert with the flag 'UseFID = TRUE' which will combine individual IDs and family IDs as FID_IID. Ensure the IDs in the lifehistory file are in the same format, for example by using `LHConvert`. The FID and IID can be split again in the resulting pedigree using `PedStripFID`.

1.2.4 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`. Using this requires a Fortran95 compiler, for example gfortran which comes with the linux-emulator 'Cygwin' for windows. 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. These files can be generated using `writeSeq`, for example after running sequoia on a subset of the data. No manual for this has been written yet, please email jisca.huisman@gmail.com if you intend to use this and require help.

1.2.5 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 highest-level function is `EstConf`, which takes a reference pedigree as input (sequoia-inferred or otherwise), and repeatedly simulates genetic data according to this pedigree (by calling `SimGeno()`), with user-specified genotyping error rates, error patterns, and call rates; then infers a pedigree based on this simulated data (by calling `sequoia`); and compares the inferred pedigree to the reference pedigree (by calling `PedCompare`) to count the number of mismatches. Each of these functions can be used separately.

1.3 sequoia 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 (note that this differs from versions prior to 1.2). Other error structures could easily be implemented but are currently not user-settable within the pedigree-reconstruction part, they are however user-settable in `SimGeno`.

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

	X		
x	0	1	2
0	$1-\epsilon-(\epsilon/2)^2$	ϵ	$(\epsilon/2)^2$
1	$\epsilon/2$	$1-\epsilon$	$\epsilon/2$
2	$(\epsilon/2)^2$	ϵ	$1-\epsilon-(\epsilon/2)^2$

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 maximum number of iterations of sibship clustering (i.e., full pedigree reconstruction, also including assignment of grandparents). Sibship clustering is much more time consuming than parentage assignment, and may take several hours for large datasets, and it is often prudent to first run with `MaxSibIter=0` so that only the much faster parentage assignment is performed, and inspect the output. It is also possible to run only the check for duplicates, by default done before parentage assignment, by specifiying `MaxSibIter=-1`.

During sibship clustering, `MaxSibIter` mostly functions as a safety net for the rare cases where the total likelihood does not converge. When 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.

Complex When it is known that the dataset contains only monogamous matings, assignment rate can be improved by using the option `Complex='mono'`, and half-siblings and half-avuncular relationships are not considered as alternatives. SNP panels that have insufficient power to distinguish between full siblings and half-siblings, may have ample power to distinguish between full siblings and third degree relatives (e.g. full cousins). This can especially make a difference when full aunts/uncles and grandparents differ consistently more in age than full siblings, and do not have to be considered as alternatives to full sibling either. Similarly, by setting `Complex='simp'` polygamous matings are still considered, but all inbred and double relationships (e.g. paternal half-siblings and maternal full cousins) are not considered as alternatives (but may still be assigned as side-effect). This again can be useful if the SNP panel has limited power, and the occurance of such complex relationships is very rare.

useAge

args.AP After parentage assignment and before full pedigree reconstruction, the age-difference distribution is estimated by `sequoia` calling `MakeAgePrior`. By default, these age-difference based prior distributions are flattened when number of pairs of assigned relatives with known age difference is limited, and any dips and the tails are smoothed. Sometimes this is undesirable, e.g. when generations do not overlap, or when the assigned relatives are known to represent the entire possible age-difference range for that relative type. Then, in args.AP any non-default arguments can be specified that will be passed on to `MakeAgePrior`.

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(GenoM = Geno,
+                    SeqList = ParOUT)
```

which will use `Specs`, `LifeHist`, `AgePriors`, and `PedigreePar` in 'ParOUT'.

2 Running Sequoia

When running `sequoia`, some or all of the following sub-programs are run:

1. **Data check:** check that the genotype data and lifehistory data are in a valid format
2. **Duplicates:** Check for identical genotypes, and for duplicated IDs in the genotype and life history data
3. **Parentage:** Parentage assignment (assign genotyped parents to genotyped focal individuals)
4. **Agepriors:** Calculation of age-difference based prior probability ratios for each type of relative
5. **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)
6. **Remaining relatives:** Idenfication of likely relatives, not assigned due to e.g. missing or incompatible age or sex information, or LLR just below the assignment threshold.

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

2.1 Data check

A common problem with SNP datasets it that errors slip through, because the data are too large to spot the errors by simply looking at the data in excel. There are many tools available to deal with genotype data and do proper quality control; the function `CheckGeno` merely checks that the data is in the correct format, and does not have any SNPs or individuals with excessively many missing values.

2.2 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 2.6).

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, no assignment is made (but the pair will be returned in `MaybeParent` when `FindMaybeRel=TRUE`, or with function `GetMaybeRel` (subsection 2.6). 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 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 parentage 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).

`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 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

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

(note that column order changed between v0.9 and v0.10, and column FS was added)

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

2.4.1 Changing the AgePriors

For example, for a species with strictly non-overlapping generations, one may wish to alter `AgePriors` to the matrix in Table 2.4, which can be done in various ways.

Completely manual, prior to any parentage assignment:

```
> AP <- matrix(0, nrow=5, ncol=9,
+              dimnames=list(0:4,
+                            c("M","P","MGM","PGF","MGF","FS","MS","PS","UA")))
> AP[1,c("MS", "PS")] <- 1
> AP[2,c("M", "P", "UA")] <- 1
> AP[3,c("MGM", "PGF", "MGF")] <- 1
> SeqOUT <- sequoia(GenoM = Geno,
+                   LifeHistData = LH,
+                   SeqList=list(AgePriors=AP),
+                    MaxSibIter = 0)
```

or using an existing pedigree:

```
> AP <- MakeAgePrior(Ped=OldPed, LifeHistData = LH,
+                    Flatten = FALSE, Smooth = FALSE)
> SeqOUT <- sequoia(GenoM = Geno,
+                   LifeHistData = LH,
+                   SeqList=list(AgePriors=AP),
+                    MaxSibIter = 0)
```

or in-between parentage assignment and sibship clustering (does not work as well if there wrongly assigned parents from a different cohort than the true parents):

```
> SeqOUT <- sequoia(GenoM = Geno,
+                   LifeHistData = LH,
+                   args.AP = list(Flatten = FALSE, Smooth = FALSE),
+                   MaxSibIter = 10)
```

or enforcing the age-difference distribution based on the existing pedigree on the sibship clustering too (useful if e.g. only a small subset of individuals has been SNP genotyped):

```
> AP <- MakeAgePrior(Ped=OldPed, LifeHistData = LH,
+                    Flatten = FALSE, Smooth = FALSE)
> SeqOUT1 <- sequoia(GenoM = Geno,
+                   LifeHistData = LH,
+                   SeqList=list(AgePriors=AP),
+                    MaxSibIter = 0)
> SeqOUT2 <- sequoia(GenoM = Geno,
+                    LifeHistData = LH,
+                    SeqList = list(AgePriors = AP,
+                                   PedigreePar = SeqOUT1$PedigreePar),
+                    MaxSibIter = 10)
```

Note that any genetically identified parent-offspring pairs which are impossible according to the age prior (not exactly 1 year / time unit apart) will be returned in `MaybeParent` (section 2.6).

2.5 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 an already assigned parent, the proportion of individuals with unknown sex or birth year, 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
- Find grandparent – grand-offspring pairs (round 3+)
- Merge existing sibships
- Replace dummy parents by genotyped individuals (round 2+)
- Add lone individuals to sibships (round 2+)

- Assign genotyped parents to genotyped individuals
- Assign grandparents to sibships (round 2+; grandparents may be dummy individuals as well as genotyped individuals)

The total likelihood (section 3.3) typically asymptotes within five to ten iterations, even for complex pedigrees. When an asymptote is reached before `MaxSibIter`, dependency on the age prior is increased (if `UseAge` = 'extra') and the algorithm continues until a new asymptote or `MaxSibIter` is reached. Then, parental likelihood ratios (LLRs) are calculated, and the algorithm is terminated. In this last step, for each individual the likelihoods for all possible relationships with each assigned parent are calculated, and the LLR between it being the parent and the most likely alternative is returned. This step may take considerable time, and can be skipped by specifying `CalcLLR` = FALSE.

2.6 Maybe-relatives

During parentage assignment, occasionally pairs are encountered which genetically are more likely to be parent-offspring than unrelated, but which can not be assigned because the sex or age difference is unknown or incompatable, or because the LLR is just below the assignment threshold (e.g. about equally likely PO and FS). Similarly, during pedigree reconstruction not all pairs of almost-certainly-relatives can be assigned, e.g. because half-siblings, full avuncular and grandparent-grandoffspring cannot always be distinguished. Distinguishing different kinds of second degree relatives 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, the output indicates '2nd' as most likely relationship, and the LLR is between being 2nd degree relatives versus the most likely of PO, FS, HA (3rd degree relative) or U.

These pairs can be identified by running `sequoia` with option `FindMaybeRel=TRUE`, or by running function `GetMaybeRel` with as argument a sequoia output list or (any) pedigree. It will also identify parent-parent-offspring trios which could not be assigned because the sex for both parents is unknown.

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

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 2.6), the total likelihood of the data after each iteration (section 3.3), and the input data and parameters (except the large genotype data) (see Table 4).

For an (graphical) overview of the assignment rate, the proportion of assigned parents that are dummies, sibship sizes, the function `SummarySeq` can be used.

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, forming a sibship with a single offspring. 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).

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). For more details, see FAQ.

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 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

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

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]).

3.4 Save output

There are various ways in which the output can be stored. This includes saving the seqoia list object, and optionally any other object, in an .RData file

```
> save(SeqList, LHdata, Geno, file="Sequoia_output_date.RData")
```

which can be read back into R at a later point

```
> load("Sequoia_output_date.RData")
> # 'SeqList' and 'LHdata' will appear in R environment
```

The advantage is that all data is stored and can easily be manipulated when recalled. The disadvantage is that the file is not human-readable, and (to my knowledge) can only be opened by R.

Alternatively, the various dataframes and list elements can each be written to a text file in a designated folder. This can be done using `write.table` or `write.csv`, or (since v0.10) using `writeSeq`:

```
> writeSeq(SeqList, GenoM = Geno, folder=paste("Sequoia_OUT", Sys.Date()))
```

which also creates a README file, to remind one that this was created by sequoia and the date. This can be used for any notes or comments, and any R scripts could be saved in the same folder.

The same function can also write the dataframes and list elements to an excel file (.xls or .xlsx), each to a separate sheet, using library `xlsx`:

```
> writeSeq(SeqList, OutFormat="xls", file="Sequoia_OUT.xlsx")
```

Note that 'GenoM' is ignored, as a very large genotype matrix may result in a file that is too large for excel to open. If you have a genotype matrix of modest size, you can add it to the same excel file:

```
> library(xlsx)
> write.xlsx(Geno, file = "Sequoia_OUT.xlsx", sheetName="Genotypes",
+       col.names=FALSE, row.names=TRUE, append=TRUE, showNA=FALSE)
```

The option `append=TRUE` ensures that the sheet is appended to the file, rather than the file overwritten.

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 2, to non-genotyped individuals in pedigree 1.

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 inferred 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 pretend 20% of birth years and genders are unknown. The specific numbers will differ between simulated datasets, but the output structure will be the same.

```
> data(LH_HSg5, Ped_HSg5)
> GM <- SimGeno(Ped = Ped_HSg5, nSnp = 200, ErHQ = 1e-3)
> #
> LH <- LH_HSg5
> LH$BY[sample.int(nrow(LH), round(nrow(LH)*0.2))] <- NA
> LH$Sex[sample.int(nrow(LH), round(nrow(LH)*0.2))] <- NA
> #
> # run sequoia, with max 5 iterations of full pedigree reconstruction
> SeqX <- sequoia(GenoM = GM,  LifeHistData = LH, MaxSibIter = 5)
> #
> #check the number of mismatches in the full pedigree
> comp <- PedCompare(Ped1 = Ped_HSg5, Ped2 = SeqX$Pedigree)
> comp$Counts
> # , , dam
> #
> #    Total Match Mismatch P1only P2only
> # GG   529   522        4      3      0
> # GD   367   363        4      0      0
> # GT   892   885        4      3      0
> # DG    39    39        0      0      0
> # DD    29    29        0      0      0
> # DT    68    68        0      0      0
> # TT   961   953        4      3      1
> #
> # , , sire
> #
> #    Total Match Mismatch P1only P2only
> # GG   550   549        1      0      0
> # GD   343   337        5      1      0
> # GT   892   886        5      1      0
> # DG    38    38        0      0      0
> # DD    30    30        0      0      0
> # DT    68    68        0      0      0
> # TT   960   954        5      1      0
```

The errors are `Mismatch + P2only`, while `P1only` are the non-assigned parents

```
> # error rate:
> (4+1+5+0)/(2*960)
> #[1] 0.005208333
>
> # correct assignment rate
> (953+954)/(2*960)
> #[1] 0.9932292
```

We can investigate the mismatches further (in Rstudio, you can also use `View(comp$Mismatch)`):

```
> comp$Mismatch
>  #     id  dam.1 sire.1 dam.2 sire.2   id.r  dam.r  sire.r Cat Parent
>  # b05019 a04004 b04002 F0003  M0004 b05019 a04001  b04002  GG    dam
>  # b05018 a04004 b04002 F0003  M0004 b05018 a04001  b04002  GG    dam
>  # a05017 a04004 b04002 F0003  M0004 a05017 a04001  b04002  GG    dam
>  # b05020 a04004 b04002 F0003  M0004 b05020 a04001  b04002  GG    dam
>  # b05164 a04053 b04048 F0047  M0031 b05164 a04053 nomatch  GG   sire
>  # a05090 a04053 b04164 F0047  M0031 a05090 a04053 nomatch  GD   sire
>  # b05092 a04053 b04164 F0047  M0031 b05092 a04053 nomatch  GD   sire
>  # a05091 a04053 b04164 F0047  M0031 a05091 a04053 nomatch  GD   sire
>  # a04004 a03173 b03044 F0031  M0009 a04004 a03173  b03093  GD   sire
```

and split the mismatches by the three errors

dam a04004 vs F0003 The offspring of dam a04004 and sire b04002 in pedigree 1 are assigned the correct sire in pedigree 2, but apparently the wrong dam (F0003). We can gather some information about this dummy dam

```
> SeqX$DummyIDs[SeqX$DummyIDs$id=="F0003", ]
> #      id   dam  sire LLRdam LLRsire LLRpair sex BY.est BY.min BY.max NumOff     O1
> # 3 F0003 F0031 M0007   9.34   10.79    4.11   1      5      5      5     12 b05019
> #       O2     O3     O4     O5     O6     O7     O8     O9    O10    O11    O12
> # 3 a05017 b05173 a05174 b05175 a05176 b05037 b05038 b05040 a05039 b05020 b05018
```

based on its offspring (b05019, a05017, ...), `PedCompare` judges that this dummy female most likely is the non-genotyped individual a04001 (column dam.r in `comp$Mismatch`). A closer look at the true pedigree shows that this female is a full sibling of the true dam a04004

```
> Ped_HSg5[Ped_HSg5$id %in% c("a04001", "a04004", "b04002"), ]
> #         id    dam   sire
> # 617 a04001 a03173 b03044
> # 618 b04002 a03173 b03044
> # 620 a04004 a03173 b03044
```

Moreover, b05019 and its siblings are the result of a full-sib mating, further complicating the assignment.

sire b04165 vs M0031 We can have a look at the offspring assigned to dummy male M0031:

```
> PedM <- comp$MergedPed  # just to save typing
> #
```

```
> PedM[which(PedM$sire.2=="M0031"), ]
> #         id  dam.1 sire.1 dam.2 sire.2 id.r  dam.r  sire.r
> # 877 a05090 a04053 b04164 F0047  M0031 <NA> a04053 nomatch
> # 878 b05164 a04053 b04048 F0047  M0031 <NA> a04053 nomatch
> # 879 b05092 a04053 b04164 F0047  M0031 <NA> a04053 nomatch
> # 880 a05091 a04053 b04164 F0047  M0031 <NA> a04053 nomatch
```

and see that all but one (a05164, second row) share the same true sire b04164.

We can have a look if b040164 has more true offspring

```
> PedM[which(PedM$sire.1=="b04164"), ]
> #         id  dam.1 sire.1  dam.2 sire.2 id.r  dam.r  sire.r
> # 846 b05175 a04001 b04164  F0003  M0028 <NA> a04001  b04164
> # 847 a05166 a04122 b04164 a04122  M0028 <NA>   <NA>  b04164
> # 848 a05176 a04001 b04164  F0003  M0028 <NA> a04001  b04164
> # 849 a05089 a04053 b04164  F0047  M0028 <NA> a04053  b04164
> # 850 a05167 a04122 b04164 a04122  M0028 <NA>   <NA>  b04164
> # 851 a05174 a04001 b04164  F0003  M0028 <NA> a04001  b04164
> # 852 b05173 a04001 b04164  F0003  M0028 <NA> a04001  b04164
> # 853 b05165 a04122 b04164 a04122  M0028 <NA>   <NA>  b04164
> # 877 a05090 a04053 b04164  F0047  M0031 <NA> a04053 nomatch
> # 879 b05092 a04053 b04164  F0047  M0031 <NA> a04053 nomatch
> # 880 a05091 a04053 b04164  F0047  M0031 <NA> a04053 nomatch
> # 897 a05168 a04122 b04164   <NA>   <NA> <NA>   <NA>    <NA>
```

and see that his offspring are split across two sibships, M0028 and M0031, resulting in an Mismatch count equal to the size of the smaller of the two halves (here 3). One offspring (a05169) is not assigned a dam or sire in pedigree 2, contributing to the 'P1only' count.

Both the split and the non-assignment are most likely side effects of the mis-assignment of b04164 as full sibling rather than maternal half-sibling of a05090, b05092 and a05091, resulting in a mis-estimation of the most likely genotype of the non-genotyped shared father.

a04004 This individual was assigned M0009 as father (sire.2), which corresponds to non-genotyped male b03093 (sire.r), while its true father (sire.1) is b03044.

```
> PedM[which(PedM$dam.1=="a03173"), ]
> #         id  dam.1 sire.1 dam.2 sire.2   id.r  dam.r sire.r
> # 619 a04003 a03173 b03044 F0031  M0007   <NA> a03173 b03044
> # 636 b04080 a03173 b03093 F0031  M0009   <NA> a03173 b03093
> # 639 b04079 a03173 b03093 F0031  M0009   <NA> a03173 b03093
> # 640 a04004 a03173 b03044 F0031  M0009   <NA> a03173 b03093 <--
> # 643 a04078 a03173 b03093 F0031  M0009   <NA> a03173 b03093
> # 645 a04077 a03173 b03093 F0031  M0009   <NA> a03173 b03093
> # 968  M0004 a03173 b03044 F0031  M0007 b04002 a03173 b03044
> # 970  F0003 a03173 b03044 F0031  M0007 a04001 a03173 b03044
```

Thus, a04004's mother mated with both b03044 and b03093, and a04004 got clustered with the wrong full sibling group (but the correct maternal half-siblings).

4.1.1 Dyads

If you only care if pairs of individuals are 'full sibs', 'half sibs' or 'other', you can use `dyadcompare`

```
> DyadCompare(Ped_HSg5, SeqX$PedigreePar)
> #     RC.2
> # RC.1     FS     HS      U
> #   FS    561    325    500
> #   HS      0   2131   2579
> #   U       0      0 416644
```

which here shows that no unrelated individuals (row U) are wrongly assigned as full (column FS) or half (HS) siblings, while many full sib pairs were left unassigned.

4.1.2 Colony

To compare Colony output with an existing pedigree, use:

```
> BestConfig <- read.table("Colony/file/file.BestConfig",
+                          header=T, sep="", comment.char="")
> PedCompare(Ped1 = ExistingPedigree,
+            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.

Since version 0.10, this process is conveniently wrapped in the function `EstConf`.

```
> data(SimGeno_example, LH_HSg5, package="sequoia")
> SeqOUT <- sequoia(GenoM = SimGeno_example[, 1:100],
+                   LifeHistData = LH_HSg5, MaxSibIter = 5)
> ConfPr <- EstConf(Ped = SeqOUT$PedigreePar,
+                   LifeHistData = LH_HSg5,
+                   Specs = SeqOUT$Specs, Full = TRUE,
+                   nSim = 3, ParMis = 0.4)
> # , , mean
> #      GG    GD    GT  DG  DD  DT    TT
> # dam   1 0.950 0.980 NaN NaN NaN 0.980
> # sire  1 0.986 0.995 NaN NaN NaN 0.995
> #
> # , , min
> #      GG    GD    GT  DG  DD  DT    TT
> # dam   1 0.896 0.957 NaN NaN NaN 0.957
> # sire  1 0.957 0.986 NaN NaN NaN 0.986
>
```

The second set of confidence probabilities ('min') is calculated using the maximum number of errors in a simulation, rather than the average number.

To add confidence probability to the pedigree based on real data, assuming that replacement of dummies by IDs of non-genotyped individuals is free from error,

```
> PedC <- PedCompare(Ped1 = Ped_HSg5,
+                    Ped2 = SeqOUT$Pedigree)$ConsensusPed
> ConfProb <- cbind(ConfPr[,,"mean"],
+                   "U" = NA,  # Ungenotyped, parent taken from Ped1
+                   "X" = NA)  # no parent in either pedigree
> PedC$dam.cat2 <- PedC$dam.cat
> PedC$dam.cat2[PedC$dam.cat == "GR"] <- "GD"
> PedC$dam.cat2[PedC$dam.cat == "RG"] <- "DG"
> PedC$dam.cat2[PedC$dam.cat %in% c("DD", "DR", "RD", "RR")] <- "DD"
> PedC$dam.prob <- ConfProb["dam", as.character(PedC$dam.cat2)]
>
> # and analogously for sires.
```

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/#MakingaGRM`, 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.

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

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	Y[2]				Y
GP	Y	–	Y	[1]						Y
FA		–				Y				Y
HA		Y	Y[2]							Y
F1C										Y
GGG		–					[3]			Y

5.2 Hermaphrodites

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`, and parent-parent-offspring trios in `MaybeTrio`.

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, the one that matches the prior information is chosen. Parent–offspring pairs in the pedigree prior that are not genetically a match will never be assigned.

```
> cand.dams <- read.table("Candidate_dams.txt", header=TRUE,
+                         stringsAsFactors=FALSE)
> # cdam has columns 'id' and 'dam', and does not need entries for all ids
> cand.par <- cbind(cand.dams, sire=NA)
> par.herm <- sequoia(GenoM = Geno,
+                     LifeHistData = LH,
+                     SeqList = list(PedigreePar = cand.par),
+                     MaxSibIter=0)
> # In combination with maxSibIter=0, PedigreePar is a pedigree prior
>
> # re-use all settings (including the newly assigned parents):
> seq.herm <- sequoia(GenoM = Geno,
+                     LifeHistData = LH,
+                     SeqList = par.herm,
+                     MaxSibIter = 10)
```

Note that the functionallity for hermaphrodites has not been as extensively tested as the rest of the program, and especially the option to cluster sibships is still a beta version - use with caution.

5.3 Cluster families

Certain analyses, such as the Mendelian error check in PLINK, are done on a family-by-family basis. The function `FindFamilies` takes a pedigree as input and clusters the individuals in as few families as possible, by repeatedly searching all ancestors and all descendants of each individual and ensuring those all have the same family ID.

This function does not take separate FID and IID columns in the input pedigree, rather these need to be joined together before running `FindFamilies`, and then split afterwards using `PedStripFID`.

5.4 Pedigree stats & plots

A table with basic pedigree summary statistics can for example be generated using `pedigreeStats` from library pedantics. This package can also draw the pedigree, using `drawPedigree`.

There is a range of software available to plot pedigrees in various styles, such as for example PedigreeViewer. Be aware that many use a different column order (id - sire - dam, instead of id - dam - sire used here), and often use '0' to denote missing parents, rather than NA.

References

[1] J Huisman. Pedigree reconstruction using snp data: parentage assignment, sibship clustering, and beyond. *Molecular Ecology Resources*, 17(5):1009-1024.

[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.

6 Function overview

6.1 Input check

GenoConvert Read in genotype data from PLINK file, Colony file, or many user-specified formats, and return a matrix in sequoia's (or Colony) format.

LHConvert Extract sex and birthyear from PLINK file; can recode sex to 1=female, 2=male, check consistency with other LifeHistData, or combine family ID and individual ID into FID_IID.

CheckGeno Check that genotype matrix is in valid format for sequoia, and find individuals and SNPs with very low call rate, to be (automatically) excluded.

SnpStats Calculate per-SNP allele frequency, missingness, and, if any pedigree provided, number of Mendelian errors.

6.2 Simulate genotype data

SimGeno Simulate genotype data for independent SNPs. Specify pedigree, founder MAF, call rate, proportion of non-genotyped parents, genotyping error & error model, inheritance mode.

MkGenoErrors Add genotyping errors and missingness to genotype data; more fine-scale control than with SimGeno.

EstConf Estimate assignment error rate (false positives & false negatives). Using a reference pedigree, repeatedly simulate genotype data, run sequoia, and compare inferred to reference pedigree.

6.3 Age and birth years

MakeAgePrior Estimate age-difference distribution (ADD) for each type of relatives, correcting for ADD in sample and for few/no pairs of some relationships yet.

PlotAgePrior Plot 'AgePriors' in sequoia output list, or output from MakeAgePrior incl. intermediate results.

6.4 Pedigree reconstruction

sequoia Main function to run parentage assignment and full pedigree reconstruction, calls many of the other functions.

SummarySeq Graphical overview of the assignment rate, the proportion dummy parents, sibship sizes, parental LLR distributions, and Mendelian errors.

GetMaybeRel Find pairs of (remaining) putative relatives in the data; either only parent-offspring, or all 1st and 2nd degree relatives.

writeSeq Write the list with sequoia output in human-readable format, either as a folder with .txt files or a many-tabbed excel file.

6.5 Pedigree check

GetRelCat For each pair of individuals, determine their relationship according to the pedigree.

PedCompare Compare 2 pedigrees, e.g. field and genetically inferred, or reference and inferred-from-simulated-data. Matches dummy parents to non-genotyped parents.

DyadCompare Compare the pairwise relationships between 2 pedigrees.

6.6 Families within the pedigree

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

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

7 FAQ

7.1 Error messages when reading in data

Hopefully with the updated 'GenoConvert' and new 'CheckGeno' function this will be less of an issue. For smaller datasets (say up to 1000 individuals) it may be useful to read in the data with read.table (using header=TRUE or FALSE, as appropriate, and typically row.names=1), and inspect the data with table(as.matrix(mydata)) for odd entries. For very large datasets, specialist tools may be required to get the data in a standardised format.

7.2 Is age information really needed?

Parentage assignment Purely genetically, it is impossible to tell who is the parent and who the offspring in a parent–offspring pair. Therefore, a parent can only be assigned if birth years are provided for both individuals. Exception is when a parent-pair has been sampled, as a complementary set of parents can be distinguished from a pair of offspring. Any remaining individuals with which the offspring (A) of such a parent-pair (B + C) forms a parent-offspring pair (say D, E, F) must be an offspring of A (with B + C as grandparents). Thus, a high proportion of sampled parents may somewhat compensate for unknown birth years.

Providing guestimated birth years can substantially increase assignment rate, but at the risk that parent–offspring pairs are flipped the wrong way around in the pedigree, which in turn may lead to other wrong assignments. This may especially be a problem in long-lived species which start breeding at an early age.

If the number of individuals with unknown birth years is not excessively large, and some additional information is available to orient parent–offspring pairs the right way around, this risk can be minimised by following an iterative approach: First, run parentage assignment (MaxSibIter = 0) using only birth years which are known with high accuracy. This will return amongst others a dataframe with all parent–offspring pairs for which it could not be determined who is the parent and who the offspring (if any). Then, guestimated birth years can be entered for the simpler cases: individuals in only a single PO-pair, with a individual of known age, where one individual is clearly older than the other based on size or other phenotypic characteristics; or individuals in a range of PO pairs, with a substantial age gap between a set of younger individuals (probably offspring) and one older individual (probably parent). Then, parentage assignment can be run again, newly assigned parents double checked if needed, and additional guestimated birth years added to the lifehistory file. This process can be repeated as often as necessary, and different variations tried out relatively quickly as the parentage assignment takes a few minutes at most even for datasets with thousands of individuals (set CalcLLR = FALSE to speed up this process even further).

Sibship clustering The three different types of second degree relatives (half siblings, grandparent – grand-offspring and full avuncular (aunt/uncle – niece/nephew) are genetically indistinguishable, unless both individuals of the pair already have a parent assigned. In most species, there is limited overlap between the age-difference of half siblings versus grandparents, and in some species avuncular relationships can be distinguished based on age too. In many cases the age-difference distributions of these types strongly overlap, but can nonetheless be informative near the tails.

7.3 How to assign parents for a new cohort

Unlike many other pedigree reconstruction programs, sequoia does not work on a cohort-by-cohort base. Therefore, when a new cohort of offspring has been genotyped and is to be added to the pedigree, it is best to re-run pedigree for all genotyped individuals. This ensures that older siblings of the new offspring are identified, as well as any grandparents. Exception is when all candidate parents have been genotyped, although even then inclusion of their parents may correct for any genotyping errors.

7.4 Why is the assignment rate so low?

In many cases, assignment rate can be boosted without increasing the number of SNPs: by assuming a higher genotyping error rate, providing sex or birth year information on more individuals, or fine-tuning the age-difference based prior. When polygamy is rare and not of particular interest, assuming a monogamous mating system typically increases assignment rate, as does ignoring complex relationships (paternal half-sibling + maternal half-aunt etc.) when these are rare. These mating system choices will risk erroneous assignments when polygamy and complex relationships, respectively, do occur.

Not enough SNPs As opposed to most other pedigree assignment programs, Sequoia does not rely on MCMC to explore many different pedigree possibilities, but instead sequentially assigns highly likely relationships, and expands the pedigree step by step. For a relationship to be highly likely, a substantial number of SNPs is necessary, larger than for MCMC methods: at least 100-200 for parentage assignment, or full sibling clustering in a monogamous population, and at least 400-500 otherwise.

Genotyping errors The default settings for the assumed genotyping error rate and the maximum number of opposing homozygous loci between parent and offspring (MaxMismatch) are based on data from a SNP array after stringent quality control. MaxMismatch is merely a first filtering step, and increasing its value is unlikely to increase the false positive rate — but it may increase computational time considerably. A suitable value can be found by first running parentage assignment (MaxSibIter = 0) with a very high value (say a quarter of the number of SNPs), inspect the range of OH in the resulting pedigree, and set MaxMismatch accordingly (with an appropriate safety margin). A suitable assumed genotyping error rate can be set similarly, or can be based on for example duplicated samples.

A few SNPs with a high (apparent) error rate may throw off pedigree reconstruction, resulting in differences in assignment rate and sometimes even which parents are assigned when the assumed genotyping error rate is varied. The function SnpStats() returns the number of mendelian errors per SNP,t when both the genotype matrix and a pedigree are provided (newly inferred or e.g. existing field pedigree), and it may be worth exploring excluding a few SNPs with the highest error rates, when there are clear outliers.

Age prior When genetic data is limited, or to distinguish between different types of second degree relatives, informative age-difference based prior distributions ('age priors') may increase assignment rate. Since agepriors are updated only in-between parentage assignment and sibship clustering, parentage

assignment rate can be improved when re-running using the agepriors estimated after the first time, optionally after some manual tweaking.

If the number of genotyped individuals is limited, or includes very few pairs of close relatives with known age difference, the agepriors estimated from them will not be very informative. If a large pedigree is available from the same or a similar population (e.g. based on observations and microsatellite paternity assignments), it can be useful to estimate the agepriors from that pedigree, to use for both parentage assignment and full pedigree reconstruction of the current sample.

Lacking sex information Currently Sequoia cannot handle sex-linked markers, and therefore cannot distinguish between maternal versus paternal relatives. Sometimes the sex of an individual can be inferred, if it forms a parent-pair with an individual of known sex. This is also the manner in which the sex of dummy parents is determined; half-sibships sharing a parent of unknown sex are not currently implemented. Pairs of relatives for which it is unclear whether they are maternal or paternal relatives, are returned in MaybePar and MaybeRel.

Mating system When the population has a complex mating system, with overlapping generations and many double relatives, a large number of SNPs is needed to distinguish between various plausible alternatives. When the power of the SNP panel is insuffcient to make the distinction, no assignment will be made. However, when it is known in advance that such complex relationships are very rare, assignment rate can be boosted by ignoring them as possible alternatives (choosing Complex='simple').

7.5 Why does it not use year of death?

The year of death forms an upper limit to when an individual could have reproduced, and is used by e.g. FRANZ during parentage assignment. It is not used by sequoia, because it is mainly focussed on populations of wild animals, where it is often impossible to tell whether an individual has emigrated or died, and identification of dead individuals is not always reliable. Emigrants are still candidate parents, as they may reside just outside the study area boundaries, or return briefly and unseen during the breeding season.

7.6 How do I know if the assigned parents are correct?

Field pedigree If the genetically assigned mother matches the mother caring for the individual, or the plant from which the seed was collected, there is little reason to doubt the assignment. Similarly, when the genetically assigned father matches (one of) the observed mates of the mother, the assignment is most likely correct.

In rare other cases, the genetically assigned parent is impossible, for example because the assigned parent was not alive at the time of birth (for mothers) or conception (for fathers). The blame for such an erroneous assignment may be the pedigree reconstruction software (due to genotyping errors, or a bug in the code), but may also be due to sample mislabelling in the lab, or a case of mistaken identity in the field.

Genomic relatedness When many thousands of SNPs are typed, it is possible to calculate the genomic relatedness (R_{grm}) between all pairs of individuals (see XX). Due to the random nature of Mendelian inheritance there is always considerable scatter of genomic relatedness around pedigree relatedness, but when the pedigree relatedness is considerably higher (say $R_{ped} > 0.2$, versus R_{grm} around 0), this is often indicative of a pedigree error. Note however that most estimators of R_{grm} assume

a large, panmictic, non-inbred population, and deviations from these assumptions may contribute to differences between R_{ped} and R_{grm}.

Data simulation Simulations as performed by 'EstConf' do not tell which assignments may be incorrect, but give an estimate of the overall number of incorrect assignments. The simulations are done presuming the inferred pedigree (or an existing pedigree) is the true pedigree, i.e. for a pedigree that is (hopefully) very close to the actual true pedigree.

7.7 Why are some parent LLR's negative or zero?

Age LR not included Especially for dummy parents of dummy individuals, the log-likelihood ratio between them being parent and offspring versus any other type of relatives (parent LLR), is regularly zero or slightly negative. The reason for this is that during pedigree reconstruction, the difference in (estimated) age is considered, and the sum of the genetic-based LLR and age-based LLR is used to make assignments. In the final step, only the genetic-based parent LLR is returned, which is identical between A being the parent of B, and B being the parent A (see also FAQ X).

Incorrect assignment In rare cases, individual A is assigned to a sibship cluster, but after addition of further siblings and perhaps grandparents to this cluster, this assignment of A is no longer the most likely. In MCMC type approaches this would not be a problem, but Sequoia implements only very limited undoing of previously made assignments. Consequently, the assigned dummy parent (B) being the parent of A is no longer the most likely among the options of AB being parent-offspring, full siblings, half siblings, grandparent-grandoffspring, avuncular, or unrelated.

Unaccounted relatedness between parents While all care is taken to avoid this, the calculation of some more exotic relationships are not (correctly) implemented for all possible combinations of dummy individuals and real individuals in the relatives' sub-pedigree. These parent-offspring pairs may be (correctly or incorrectly) assigned as a side effect during pedigree reconstruction, but return an error value (often 444 or 777) for the likelihood when attempted to calculate explicitly during the calculation of parental LLRs, resulting in a very large negative number for the LLR.