sequoia

Reconstruction of multi-generational pedigrees from SNP data

Jisca Huisman (jisca.huisman @ gmail.com)

May 17, 2020

Contents

1 Quick-start examples

1.1 Example 1: Simulated data

An example pedigree and associated life history data are provided with the package, which can be used to try out some of the functions. 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$PedigreePar')
SeqOUT <- sequoia(GenoM = Geno,
                  SeqList = ParOUT,
                  MaxSibIter = 10)
#
# compare the assigned real and dummy parents to the true pedigree
chk <- PedCompare(Ped1 = Ped_HSg5, Ped2 = SeqOUT$Pedigree)
chk$Counts["TT",,]
#
# save results
save(SeqOUT, file="Sequoia_output_date.RData")
writeSeq(SeqList = SeqOUT, GenoM = Geno, folder = "Sequoia-OUT")
```

1.2 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 400 – 700 for full pedigree reconstruction; fewer are necessary for only parentage assignment, while more may be necessary with high levels of polygamy or inbreeding. In addition you need a dataframe with the sex and birth/hatching year of as many individuals as possible, in arbitrary order.

```
# 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))
CheckGeno(Geno)
# read in many other input formats (not .vcf (yet)):
Geno <- GenoConvert(InFile = "mydata.ped", InFormat="ped")
#
# read in lifehistory data: ID-Sex-birthyear, column names ignored
# optional: minimum & maximum birth year, when not exactly known
LH <- read.table("LifeHistoryData.txt", header=T)
#
# duplicate check & parentage assignment (takes few minutes)
# (maximum number of sibship-clustering iterations = 0)
ParOUT <- sequoia(GenoM = Geno,  LifeHistData = LH_HSg5,
                  MaxSibIter = 0, Err=0.005,
                  quiet = FALSE, Plot = TRUE)
#
# inspect duplicates (intentional or accidental)
ParOUT$DupGenotype
#
# compare assigned parents to field pedigree (check column order!)
FieldPed <- read.table("FieldPed.txt", header=T)
PC.par <- PedCompare(Ped1 = FieldPed[, c("id", "dam", "sire")],
                     Ped2 = ParOUT$PedigreePar)
PC.par$Counts["TT",,]
#
# calculate Mendelian errors per SNP (works also w field pedigree)
stats <- SnpStats(Geno, ParOUT$PedigreePar)
MAF <- ifelse(stats[,"AF"] <= 0.5, stats[,"AF"], 1-stats[,"AF"])
#
# ..........................................................
# polish dataset: remove one indiv. from each duplicate pair
# & drop low call rate samples
# & drop SNPs with high error rate and/or low MAF
Geno2 <- Geno[!rownames(Geno) %in% ParOUT$DupGenotype$ID2, ]
Geno2 <- Geno2[, -which(stats[,"Err.hat"]>0.05 | MAF < 0.1)]
#
Indiv.Mis <- apply(Geno2, 1, function(x) sum(x == -9)) / ncol(Geno2)
Geno2 <- Geno2[Indiv.Mis < 0.2, ]
```

```
# check histograms for sensible thresholds, iterate if necessary
#
# run full pedigree reconstruction (may take up to a few hours)
# including re-run of parentage assignment
SeqOUT <- sequoia(GenoM = Geno2,
                  LifeHistData = LH_HSg5,
                  MaxSibIter = 20,
                  Err = 0.001)
#
# inspect assigned parents, proportion dummy parents, etc.
SummarySeq(SeqOUT)
# (see Example 1 for saving results)
```

2 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–grand-offspring, ...) 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.

3 Key points

3.1 Pedigree reconstruction – overview

When running `sequoia`, one or more of these sub-programs are run (each can also be run separately, using the function in brackets):

1. **Data check:** check that the genotype data and life history data are in a valid format (`CheckGeno`)
2. **Duplicates:** Check for identical genotypes, and for duplicated IDs in the genotype and life history data (`sequoia(..., MaxSibIter=-1)`)
3. **Parentage:** Parentage assignment (assign genotyped parents to genotyped focal individuals) (`sequoia(..., MaxSibIter=0)`)
4. **Agepriors:** Calculation of age-difference based prior probability ratios for each type of relative (`MakeAgePrior`)
5. **Full pedigree reconstruction:** Clustering of half- and full-siblings, grandparent assignment to singletons and sibships, and identification of avuncular relationships between sibships (`sequoia(..., MaxSibIter >0)`)
6. **Remaining relatives:** Identification of likely relatives, prior to pedigree reconstruction or not assigned due to e.g. missing or incompatible age or sex information, or LLR just below the assignment threshold. (`GetMaybeRel`)

3.2 Package function overview

3.2.1 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. *Does not support vcf yet.*

LHConvert Extract sex and birth year from PLINK file; optionally 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 the provided genotype matrix is in the correct format, and check for low call rate samples and SNPs

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

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

3.2.2 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.

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.

3.2.3 Age 'prior'

MakeAgePrior For various categories of pairwise relatives (R), calculate age-difference (A) based probability ratios $P(A|R)/P(A) = P(R|A)/P(R)$, or how much likelier a relationship is given the age difference. It applies corrections when the skeleton-pedigree contains few/no pairs with known age difference for some relationships.

PlotAgePrior Visualise the age-difference based prior probability ratios as a heatmap.

3.2.4 Pedigree reconstruction

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

GetMaybeRel Identify pairs of individuals likely to be related, but not assigned as such in the provided pedigree. Either search only for potential parent-offspring pairs, or for all 1st and 2nd degree relatives.

3.2.5 Pedigree check & comparisons

These functions can be applied to any pedigree, not just pedigrees reconstructed by sequoia.

SummarySeq (1 pedigree) Graphical overview of the assignment rate, the proportion dummy parents, sibship sizes, parental LLR distributions, and Mendelian errors, + tables with pedigree summary statistics.

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

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

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

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

ComparePairs (2 pedigrees) Compare, count and identify different types of relative pairs between two pedigrees. The matrix returned by `DyadCompare` [Deprecated] is a subset of the matrix returned here using default settings.

3.2.6 Miscellaneous

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

ErrToM Generate a matrix with the probabilities of observed genotypes (columns) conditional on actual genotypes (rows), or return a function to generate such matrices. The error matrix can be used as input for `sequoia` and `CalcOHLLR`, the error function as input for `SimGeno`

writeSeq Write the list with sequoia output in human-readable format, either as a folder with .txt files, or as a many-tabbed excel file. The latter uses R package `xlsx`, which requires java and can (therefore) be cumbersome to install.

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

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

3.3 Opposite homozygosity (OH) & Mendelian errors (ME)

Parent and offspring will never be opposing homozygotes (one aa, other AA) at a locus, except due to genotyping errors. This metric is easy and quick to calculate even in very large datasets, and provides a good way to to subset candidate parents (done by `sequoia` internally), or to check if an existing pedigree is consistent with new genetic data (function `CalcOHLLR(, CalcLLR=FALSE)`).

The count of opposing homozygous (OH) SNPs is not always a reliable indicator to distinguish parents from not-parents. When the genotyping error rate is high, true parent–offspring pairs may have a high OH count . On the other side, by chance some full siblings may have a very low OH count, and in small inbred populations this may be true for other types of relatives as well. Therefore, `sequoia` uses OH only as a filter to reduce the number of candidate parents for each individual, and uses likelihood ratios for actual assignments.

The number of Mendelian errors in an offspring-mother-father trio includes the OH counts between offspring and each parent, as well as cases where the offspring should have been heterozygous, but isn't (mother *aa*, father *AA*, offspring *aa* or *AA*), and cases where it is heterozygous, but shouldn't (when mother and father are identical homozyotes). This metric is used to subset 'compatible' parent-pairs when there are candidate parents from both sexes.

3.3.1 MaxMismatch

The parameter 'MaxMismatch' in previous versions proved a common source of confusion and false-negatives. It has therefore been deprecated as input parameter from version 2.0, and the value is calculated internally instead (by `CalcMaxMismatch`). It now also differentiates stringently between

- `MaxMismatchDUP`: The maximum number of differences in observed genotypes between potential duplicates
- `MaxMismatchOH`: The maximum number of opposing homozygous SNPs between potential parent and offspring
- `MaxMismatchME`: The maximum number of Mendelian errors among a potential offspring-dam-sire trio

3.4 Genotyping errors

3.4.1 Effect on pedigree inference

Unsurprisingly, a higher rate of genotyping errors leads to more false negatives and more false positives during pedigree reconstruction, as illustrated in Figure 1. The scale of the effect can be unpredictable: One wrong assignment due to a few genotyping errors in a key individual may have numerous knock-on effects, while dozens of genotyping errors in a 'dead end' individual may have no consequences at all. Accordingly, even with only 5 replicates per simulated/assumed error rate combination, there is considerable scatter in the performance metrics (Figure 1).

Runtime increases considerably with assumed genotyping error rate (3rd panel in Figure 1) because the time-saving filtering steps become less effective. At the same time are those steps may filter out some true relatives, contributing to the increase in false negatives.

An inaccurate assumed genotyping error rate `Err` can negatively affect assignment success too: among others, when `Err` is much lower than the actual genotyping error rate, some true parent-offspring pairs will be assigned as full siblings, while when `Err` is much too high some true full siblings will be classified as parent-offspring. The consequences of being off by a factor 2 are limited (Figure 1), but being off by a factor 10 or even 100 will affect assignment considerably.

Therefore, **do consider carefully which value to use for `Err`**, and explore if and how results change when you vary this parameter. The default assumed genotyping error rate of 0.01% is based on data from a SNP array after stringent quality control; in many cases the error rate will be (much) higher (but changing the default value is likely to cause problems for people re-using old code).

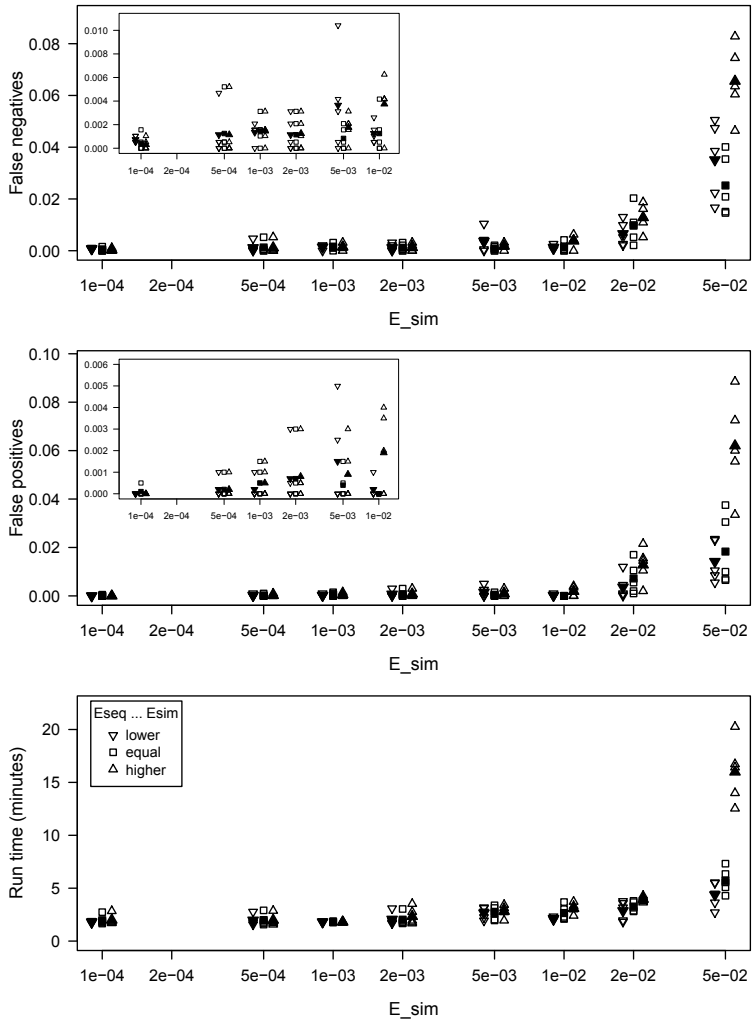


Figure 1: A higher rate of simulated genotyping error rate (x-axes) increases false negatives (top), false positives (middle) and run time (bottom). To mimic imprecise knowledge of true genotyping rates, sequoia was also run with the 'neighbouring' values of E_sim. Results from pedigree 'Ped_HSg5' with 200 SNPs, call rate of 99%, and 40% of parents non-genotyped. Filled symbols show average across the 5 replicates.

When exploring different values, possibly in combination with non-default values for the thresholds `Tfilter` and `Tassign`, do keep in mind the consequences of potential false-positives versus false-negatives during later use of the pedigree.

3.4.2 ErrFlavour

Genotyping errors are unavoidable, and different genotyping methods are likely to have different error structures. Therefore, from version 2.0 sequoia allows fine-scale control over how potential genotyping errors are accounted for. An error matrix with the probability to observe genotype G (columns), conditional on actual genotype g (rows), can now be specified to several functions via parameter `ErrFlavour`. The current default ('version2.0') is given in Table 1, and previous defaults can be given in the help file of `ErrToM`. The slight differences between versions only have any consequences when the error rate is high ($> 1\%$).

Table 1: Probability to observe genotype G (columns) conditional on actual genotype g (rows) and per-locus error rate E

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

During pedigree inference, the error rate is presumed equal across all SNPs. It is also assumed that none of the errors (or missingness) are due to heritable mutations.

3.4.3 Estimation

Function `SnpStats` can estimate the genotyping error rate per SNP, conditional on a provided pedigree and error structure (`ErrFlavour`). These estimated error rates will not be as accurate as from duplicate samples, since a single error in an individual with many offspring will be counted many times, while errors in individuals without parents or offspring will not be counted at all.

3.5 Birth years & Age difference based 'prior'

For details, including worked examples, mathematical framework, and full range of options, please see the separate vignette on this topic.

For every species, some age differences between parent and offspring or between siblings are more likely than others, and some are downright impossible. Parents are always older than their offspring, and this reduces the number of candidate parents to consider. Similarly, a species' minimum and maximum age of reproduction limits the age-difference between siblings. When these differ between the sexes, it can help distinguish between maternal and paternal half-siblings.

In addition, for parent–offspring pairs it is impossible to determine genetically which one is the parent, and which one is the offspring — age data is required to orientate the pair the right way around in the pedigree. Exception is when a 'complementary' set of parents can be identified, which may both be of unknown age (but for at least one the sex must be known to make an assignment).

3.5.1 Definition & interpretation

The 'agepriors' as used by `sequoia` s not an official term, nor a proper prior, but a set of age-difference based probability ratios which can be interpreted as

> *If I were to pick two individuals with an age difference A, and two individuals at random, how much more likely are the first pair to have relationship R, compared to the second pair?*

And the value may vary from 0 (impossible), to 1 (just as likely), and typically not far beyond 10 (10x as likely). This quantification allows age-differences to be used jointly with genetic information, which is mostly (except when `UseAge='extra'`) done in a conservative fashion: only when the focal relationship is the most likely both when

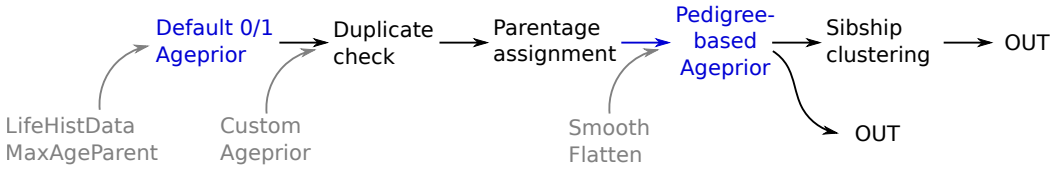
considering only genetic data, and when considering genetic + age data, is the assignment made. Note that 'most likely' applies to 'from the set of possible relationships', i.e. those with an age prior value > 0.

3.5.2 Implementation

Estimation of the ageprior is done from the age differences between relatives according to a provided pedigree by function `MakeAgeprior`. Since often not all biologically possible age-differences may be present in the provided pedigree, by default it applies a correction to allow for relatives to differ 1-2 years more in age than observed in the pedigree, and to smooth out any dips or gaps in the distributions.

The probability ratios are included `sequoia`'s output in the list element `AgePriors` as a matrix with 5 columns (M = mother, P = father, FS = full sibling, MS = maternal sibling, PS = paternal sibling) and as many rows as the birth year range detected in the life history data, or as specified by `AgePriors`'s argument `MaxAgeParent`.

By default, during parentage assignment the ageprior only specifies that an age difference of 0 (first row) for parent-offspring pairs is impossible (0) and that all other age-relationship combinations are possible (1). It is updated after parentage assignment, based on the age differences in the just-assigned scaffold pedigree, and this update is used during full pedigree reconstruction (Figure 3.5.2).



3.5.3 Grandparents, aunts & uncles

The ageprior distributions for grandparents and avuncular pairs (aunts/ uncles) are calculated from the distributions for parent-offspring and siblings just before full pedigree reconstruction, and included in the output as `ParOUT$AgePriorExtra`. The distribution for avuncular pairs is not strictly symmetrical around zero (see row names of this matrix); aunts/uncles are often older than their nieces/nephews, but not necessarily so. Note that up to sequoia v1.3 it was assumed that the ageprior for maternal and paternal full- and half- aunts and uncles was approximately similar, and approximately symmetrical around zero – both these assumptions have been dropped from version 2.0.

These ageprior distributions are an important tool to distinguish between half-siblings, grandparent–grand-offspring and full avuncular pairs, which are genetically indistinguishable unless both already have at least one parent assigned. Especially when there are only one or two age classes in the data, or none (all birth years unknown), it is worth contemplating whether or not it is desirable to consider full- and half-avuncular relationships as alternatives, and checking whether the extended ageprior (`ParOUT$AgePriorExtra`) matches the desired behaviour.

3.5.4 Customising the ageprior

When running `sequoia`, you can pass arguments to `AgePriors` via `args.AP`. For example, when you are sure generations do not overlap in your sample, you can specify this via

`AgePriors`'s argument `Discrete`:

```
SeqOUT <- sequoia(GenoM = Geno, LifeHistData = LH,
                  args.AP = list(Discrete = TRUE))
```

Alternatively, if you have a field-pedigree or microsatellite-based pedigree that contains many more individuals than have been SNP-genotyped, an ageprior estimated from that old pedigree will be much more informative than the one estimated from a limited number of SNP-genotyped parent-offspring pairs. This can be done as follows:

```
APfromOld <- MakeAgePrior(Pedigree = MyOldPedigree,
                          LifeHistData = LH,
                          Smooth = TRUE)
SeqOUT <- sequoia(GenoM = Geno,
                  SeqList = list(AgePriors = APfromOld),
                  MaxSibIter = 10)
```

When argument `SeqList` contains an element `AgePriors`, that one is used during both parentage assignment and sibship clustering. Thus, if you wish to use different agepriors during those different phases, you need to run them separately (see Section 4.3.1).

When you have no pedigree, or no birth year information, but know the maximum age of parents, you can specify just that:

```
APdraft <- MakeAgePrior(MaxAgeParent = c(4, 5))  # dams, sires
```

You optionally could edit this matrix manually before using it as input in `sequoia`.

3.5.5 Age-based excluded true relatives

Any genetically identified parent-offspring pairs, or other types of relatives, which are impossible according to the specified age prior can be found by `GetMaybeRel`. When you include the inferred pedigree as input, any relative pairs in the pedigree will be excluded from the output. I suggest to re-run pedigree reconstruction after updating the ageprior to allow for their age difference, and/or after correcting their birth years.

3.6 Parent log-likelihood ratios (LLR)

When parentage assignment or pedigree reconstruction is completed (when the total likelihood has asymptoted) for each assigned parent-offspring pair a log-likelihood ratio (LLR) is calculated. This is the ratio between the likelihood of being parent and offspring, versus the next-most-likely relationship between them, conditional on all other relationships in the pedigree. Note that this differs from for example Cervus [2], which returns the ratio between the likelihood that the assigned parent is the parent, versus that the next most likely candidate is the parent.

The calculation of these parental LLR's can be rather time-consuming, and therefore `sequoia` can be run with `CalcLLR=FALSE`. When one wants to calculate the parental LLR's later, or wants to calculate them for any existing pedigree, use `CalcOHLLR`.

Very small or negative LLR's The parent LLR's are calculated based on genetic data only, without any consideration of age. Consequently, they are often close to zero for dummy – dummy parent-offspring pairs, as genetically the parent may just as well be the offspring. Such pairs will not have been assigned unless there is a compatible co-parent (`LLRpair` should always be positive) or if there is overwhelming age-based evidence to decide which one is the parent.

When a parental LLR is negative without an accompanying positive `LLRpair`, or when `LLRpair`, this is reason to cautious about the assignment and for example compare it with a previous pedigree (section 6.2) or inspect the genomic relatedness (section 6.3). In contrast to MCMC type approaches, `sequoia`'s algorithm has very limited scope to undo previously made assignments, and may get stuck at suboptimal solutions.

Extremely negative LLRs (< -100) are typically caused by one or more of the alternative relationships considered becoming very convoluted when conditioning on the rest of the pedigree, and the particular case neither being caught as 'highly improbable and not implemented' nor being implemented properly (see Table 7.2 on page 29 for double relationships that are implemented). In the unlikely event that the parent-offspring relationship is not implemented properly (the parent-offspring link was a 'side-effect' of other assignments), you will see an error value of 444 or 777.

3.7 Estimate confidence levels

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 (`SimGeno`);
- reconstructing a pedigree from these simulated data (`sequoia`);
- counting the number of mismatches between the 'true' pedigree, used as input for the simulated data, and the pedigree reconstructed from the simulated data (`PedCompare`).

When repeated at least 10–20 times, the proportion of assignments that is correct provides an estimate of 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.

This process is conveniently wrapped in the function `EstConf`, which calculates separate confidence levels for dams and sires (which may have very different sampling proportions), and for the different combinations of genotyped/dummy(/none) individual, parent and co-parent.

4 Input

4.1 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))
```

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.

4.1.1 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.

Since the full dataset may be too large to easily load into R, creating a subset of SNPs can for example be done using PLINK:

```
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 and used to estimate the genotyping error rate.

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", InFormat="raw")
```

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

4.1.2 Exclusion of low call rate samples & SNPs

Samples with a very low genotyping success 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. Samples with a call rate below 5% are automatically excluded, but it is strongly advised to create a subset where all individuals are genotyped for at least 50% of SNPs, and preferably for at least 80%.

In addition, SNPs with a call rate below 10% are excluded, as these contribute almost no information. Again, a stricter threshold is advised of at least 50%.

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

4.1.3 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 life history 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`.

4.1.4 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. The input consists of three text files: the genotype data with one column for IDs followed by one column per SNP (0/1/2/-9), and no header row; the life history data; 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.

4.2 Life history data

The life history data (`LifeHistData`) should be a dataframe with three or five 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".
- 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

Table 2: Example LifeHistData with the optional columns for minimum + maximum birth year.

ID	Sex	BirthYear	BYmin	BYmax
A	2	NA	NA	2010
B	2	2012	NA	NA
C	1	2011	NA	NA
D	3	NA	2000	2012
E	3	NA	NA	NA

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.

- BY.min: (optional) Earliest year in which individual may have been born, if exact year is unknown.
- BY.max: (optional) Latest year in which individual may have been born

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.

4.2.1 BY.min & BY.max

Especially in wild populations the year of birth is regularly unknown, but often some rough estimate can be made to determine 'BY.min' and 'BY.max'. It is possible to provide only one of the pair, e.g. 'BY.max' is at the latest the first year in which an individual was observed.

Even very wide ranges may help in pedigree reconstruction: for example, if individuals A and B are genetically parent and offspring, A was first seen last year (say 2011) as an adult, and B was known to be born this year (2012), than A is certainly the parent of B. Before version 2.0 there was no system to inform sequoia that A could not have been born after year t, and thereby could have been an offspring of A. It was possible to provide 'guestimated' birth years, but this can be time consuming when there are many individuals with unknown birth years, and is potentially prone to errors.

To minimise pedigree errors where parent and offspring are flipped the wrong way around, and the 'secondary' errors derived from these cases, it is recommended to set the possible birth year range as wide as possible, at least during an initial (preliminary) round of parentage assignment. To see the pairs that are genetically parent and offspring, but for which it cannot be determined which of the two is the parent, use function `GetMaybeRel`.

4.3 sequoia parameters explained

MaxSibIter The maximum number of iterations of sibship clustering (i.e., full pedigree reconstruction, also including assignment of grandparents). Non-positive values have a special meaning:

- -9: Only input check and create `$Specs` list element
- -1: check for duplicates

- 0: check for duplicates + parentage assignment
- $x > 0$: as above + at most x iterations of full pedigree reconstruction

Sibship clustering is much more time consuming than parentage assignment and may take several hours for large datasets. It is therefore often prudent to first run only the much faster parentage assignment, and inspect the output.

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.

Err The genotyping error rate assumed, equal across all SNPs.

ErrFlavour The error model, see 3.4

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

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. Counter-intuitively a high `Tassign` can sometimes *increase* the number of wrong assignments, as a correct low-threshold assignment may prevent wrong assignments among close relatives (a.o. by revealing likely genotyping errors).

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

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', ...).

Complex The complexity of the mating system considered. One of:

- mono: only consider monogamous matings. This can be especially useful for small SNP panels that have insufficient power to distinguish reliably between full siblings and half-siblings, but may have ample power to distinguish between full siblings and third degree relatives (e.g. full cousins). Works well 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.
- simp: consider polygamous matings, but ignore all inbred and double relationships (e.g. a pair being both paternal half-siblings and maternal full cousins). Note that such relationships 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.

- full: the default, consider all kinds of 'weird' relationship combinations; see section 7.2. E.g. a pair may be both paternal half-siblings and maternal aunt-nephew, when their mothers where mother and daughter and mated with the same male.
- herm: hermaphrodites, automatically selected when any individuals in the life history data have sex=4. Otherwise like 'full'.

UseAge During parentage assignment age information is always used in the most minimalistic sense that parents must be born before their offspring. When setting `UseAge='no'`, no further use of the birth year information is made during full pedigree reconstruction either. When `UseAge='yes'`, the age distribution of dams, sires, siblings, etc. in the scaffold parentage-only pedigree is estimated (by `MakeAgePrior()`) and used to distinguish between half-siblings, grandparents and full aunts/uncles, amongst others.

This is by default done rather conservatively, but the reliance on age information can be increased by choosing `UseAge='extra'`. To prevent cascading effects of errors, the initial iterations are identical to `UseAge='yes'`, but when the total likelihood plateaus some stringent criteria are relaxed and the program continues for several more iterations.

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`.

For details on `MakeAgePrior`, see the separate vignette on that topic.

4.3.1 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 editing them

```
load("Sequoia_output_date.RData")  # if it was saved to disk
ParOUT$Specs$DummyPrefixFemale <- "D-FEM"
ParOUT$Specs$DummyPrefixMale <- "D-MALE"
SeqOUTX <- sequoia(GenoM = Geno,
                   SeqList = list(Specs = ParOUT$Specs,
                                  PedigreePar = ParOUT$PedigreePar),
                   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, which will use `Specs`, `LifeHist`, `AgePriors`, and `PedigreePar` in 'ParOUT'. If `PedigreePar` is provided and `maxSibIter`> 0, parentage assignment will not be re-run but taken from the provided pedigree.

5 Running Sequoia

5.1 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 are in the correct format, and that there are no SNPs or individuals with excessively many missing values.

5.2 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 `MaxMismatchDUP` 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.

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 7.1).

5.3 Parentage assignment

Assignment of genotyped parents to genotyped offspring is performed 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 `MaxMismatchOH`;
- 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 filter 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` (section 7.1). 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 usually takes only a few minutes, especially when setting `CalcLLR=FALSE`.

5.4 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 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, and the number of SNPs and their error rate. 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 likely full- and half-siblings
- Cluster sibling pairs into sibships
- Find and assign grandparent – grand-offspring pairs (round 3+)
- Merge existing sibships, where possible
- Replace dummy parents by genotyped individuals, where possible (round 2+)
- For individuals without parent(s), find candidate real and dummy parents, and assign parent(-pairs) where possible
- For sibships without grandparent(s), find candidate real and dummy grandparents, and assign grandparent(-pairs) where possible (round 2+)

5.4.1 Dummy Individuals

The 'dummy' parent assigned to each cluster of half-siblings is denoted by increasing numbers, by default with prefix 'F' for females and 'M' for males. Dummy individuals are appended at the bottom of the pedigree with their assigned parents, i.e. the sibship's grandparents, and by default have IDs 'F0001', 'F0002', ... for dams and 'M0001', 'M0002', ... for sires.

In addition, all information for each dummy individual is given in dataframe `DummyIDs`, including its parents (again), its offspring, and the estimated birth year, as a point estimate ('BY.est') and lower and upper bound of the 95% probability interval. This information is intended to make it easier to match dummy IDs to real IDs of observed but non-genotyped individuals (see also section 6.2).

5.4.2 Total likelihood

The total likelihood provides a measure of how well the observed genotype data is explained by the currently inferred pedigree, the allele frequencies of the SNPs, the presumed genotyping error rate, and if founder genotypes were drawn from a genepool in Hardy-Weinberg equilibrium. It is a negative number, and closer to zero indicates a better fit. The total likelihood typically asymptotes within five to ten iterations, even for complex pedigrees. When an asymptote is reached before `MaxSibIter`, either the algorithm is concluded (the default) or dependency on the age prior is increased (if `UseAge` = 'extra') and the algorithm continues until a new asymptote or `MaxSibIter` is reached. If there is a large change in value between the second-last and last likelihood of output element `TotLikSib`, consider running the algorithm for more iterations (increase `MaxSibIter`).

5.5 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.

6 Output check

6.1 Pedigree stats & plots

Various summary statistics can be calculated and displayed using function `SummarySeq`, such as the number and kind of assigned parents, family sizes (Figures 2 and 3), and the distributions of Mendelian errors. Its output also includes a table analogous to `pedigreeStats` by R package `pedantics` (which was archived on CRAN in December 2019).

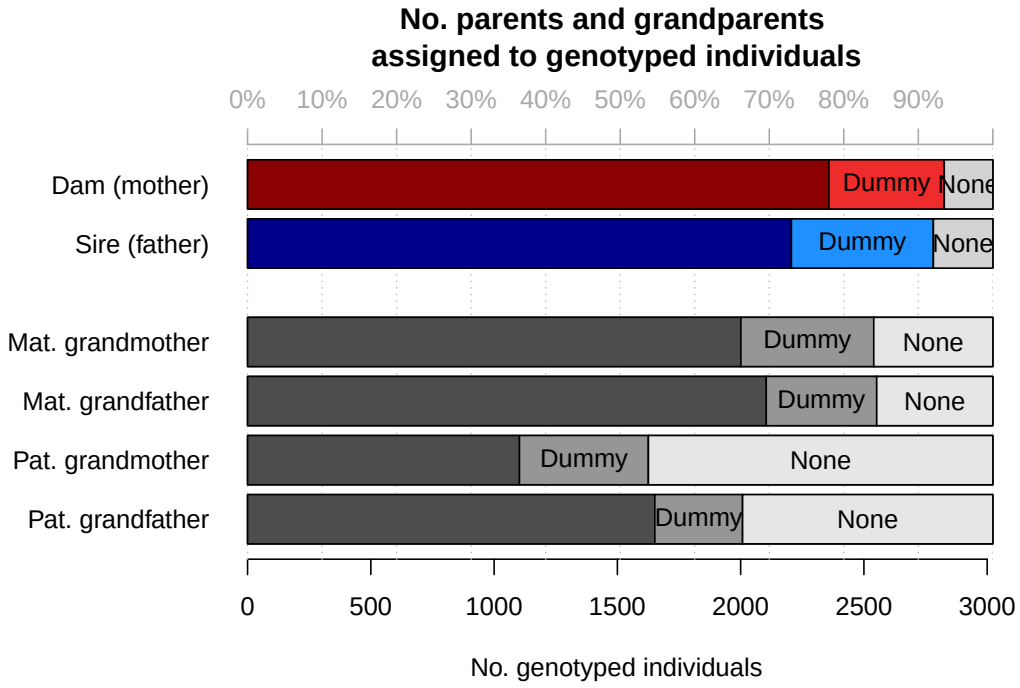


Figure 2: Number of parents assigned to genotyped individuals, split by category

There is a range of software available to plot pedigrees in various styles, such as for example R package `kinship2` or the program PedigreeViewer. Be aware that many use a different column order (id - sire - dam, instead of id - dam - sire used here), use '0' to denote missing parents rather than NA, and/or do not allow individuals to have one parent (they must be founders or have both parents). The last two issues can be resolved with `Sequoia`'s function `PedPolish`.

6.2 Comparison with previous pedigree

Often a (part) pedigree is already available to which one wants to compare the results, for example with field-observed mothers. Function `PedCompare()` performs such comparisons between pedigrees. The terminology assumes that `Ped1` is the 'true' pedigree (the one from which data is simulated) and `Ped2` is the newly inferred pedigree (from the simulated data), but the function works on any two pedigrees, and both may have genotyped as well as non-genotyped individuals.

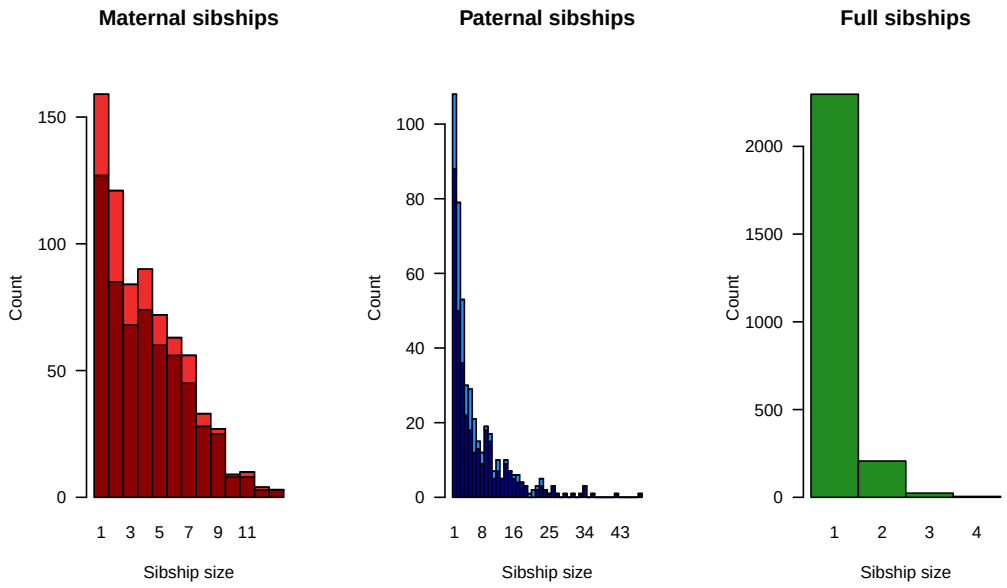


Figure 3: Family sizes, split by genotyped (dark) or dummy (light) parent

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.

6.2.1 Dummy matching

`PedCompare()` does its best to align any dummy parents in the inferred pedigree 2, to non-genotyped individuals in pedigree 1. It is considered a 'match' if the inferred sibship (in Pedigree 2) which contains the most offspring of a non-genotyped parent (in Pedigree 1), consists for more than half of this individual's offspring. When a cluster of 5 siblings in Pedigree 1 is split into one of three and one of two in Pedigree 2, the larger sibship of three is considered a match, and the smaller a mismatch (thus 2 mismatches) — even though it can be argued the inferred pedigree 2 does not contain any incorrect links.

6.2.2 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 = 100, SnpError = 0.01)
#
LH <- LH_HSg5
LH$BirthYear[sample.int(nrow(LH), round(nrow(LH)*0.2))] <- NA
LH$Sex[sample.int(nrow(LH), round(nrow(LH)*0.2))] <- NA
#
# run sequoia, parentage assignment only
SeqX <- sequoia(GenoM = GM,  LifeHistData = LH, MaxSibIter = 0,
```

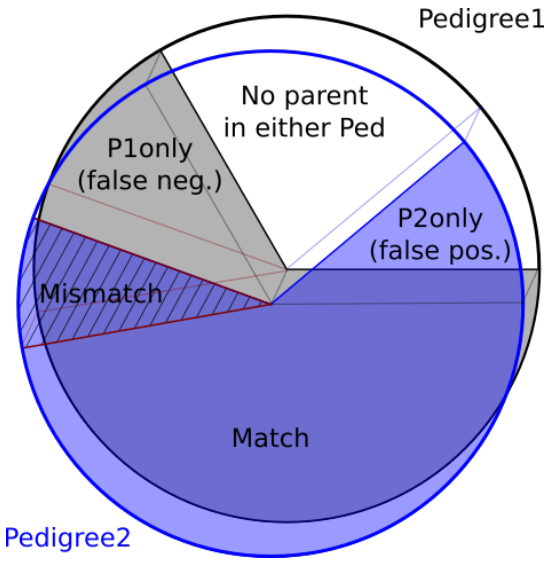


Figure 4: Various categories when comparing two pedigrees, e.g. with Pedigree1 = true pedigree used to simulate genotype data, and Pedigree2 = inferred pedigree.

```
                Err = 0.005)  # assumed genotyping error rate
#
# check if there were any assignment errors
compPar <-  PedCompare(Ped1 = Ped_HSg5, Ped2 = SeqX$PedigreePar)
compPar$Counts["GG",, ]  # GG = Genotyped offspring, Genotyped parent
#          dam sire
# Total    550  553
# Match    494  478
# Mismatch  11   12
# P1only    45   63
# P2only     0    0
```

When Pedigree 1 is the true pedigree, the assignment error rate is `Mismatch + P2only / 2N`, thus here $(11+12+0+0)/(2*1000) = 0.0115$. `P1only` are the false-negatives, i.e. cases where both offspring and its parent in Pedigree 1 were genotyped, but no assignment was made in Pedigree 2.

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

```
print(head(compPar$Mismatch[, c(1:5, 9:12)]), row.names=F)  # subset columns for easier viewing
 #    id  dam.1 sire.1  dam.2 sire.2 id.dam.cat id.sire.cat dam.class sire.class
 # a02015 a01002 b01151 a01001 b01151         GG          GG  Mismatch      Match
 # a02082 a01070 b01069 a02084 b01072         GG          GG  Mismatch   Mismatch
 # a02084 a01070 b01069 a01071 b01072         GG          GG  Mismatch   Mismatch
 # a03052 a02112 b02158 a02110   <NA>         GG          GD  Mismatch     P1only
 # a03112 a02107 b02187 a03111 b02187         GG          GG  Mismatch      Match
 # a03157 a02051 b02061 a04122   <NA>         GG          GD  Mismatch     P1only
```

and knowing that full siblings in this pedigree have consecutive IDs, we see that for the first individual, the assigned dam was a full sibling of the true dam; for the second individual, its own full sibling was assigned as dam; etc.

For the fourth individual, there is no sire assigned in Pedigree 2, but there is a sire in Pedigree 1, thus `sire.Class = P1only`. The column 'id.sire.cat' is 'GD', which indicates that the individual is Genotyped, while the sire is a potential Dummy.

Continuing with sibship clustering (this may take a while when the genotyping error rate

is high):

```
SeqXF <- sequoia(GenoM = GM,  LifeHistData = LH, MaxSibIter = 10,
                 Err = 0.005)
comp <-  PedCompare(Ped1 = Ped_HSg5, Ped2 = SeqXF$Pedigree)
comp$Counts["GG",, ]   # let's look at `GG' again first
#           dam sire
# Total    549  550
# Match    530  526
# Mismatch  10    8
# P1only     9   16
# P2only     0    0
```

We can see that in the final pedigree, there are slightly fewer mismatches and far fewer false-negatives among parent-offspring pairs that have both been genotyped. This is because the clustered sibships provide additional power to distinguish between parents, full-siblings and aunts/uncles.

```
comp$Counts["TT",, ]   # Totals
#           dam sire
# Total    962  962
# Match    885  880
# Mismatch  20   21
# P1only    55   59
# P2only     2    2
```

The overall assignment error, including dummy individuals, is $(20+21+2+2)/(2*1000) = 0.0225$, thus higher than for the parentage assignment (as is typical). We can see more easily how it differs between categories by averaging across dams and sires, and dividing by the total within each category:

```
tmp <- apply(comp$Counts, 1:2, mean)
round(tmp / tmp[,"Total"], 3)
#    Total Match Mismatch P1only P2only
# GG     1 0.961    0.016  0.023  0.000
# GD     1 0.850    0.027  0.123  0.000
# GT     1 0.918    0.021  0.062  0.000
# DG     1 0.939    0.045  0.015  0.000
# DD     1 0.946    0.018  0.036  0.000
# DT     1 0.943    0.033  0.025  0.000
# TT     1 0.917    0.021  0.059  0.002
```

The 'P2only' cases only show up in the 'TT' grand-totals category, because they involve individuals in Pedigree 2 that shouldn't exist:

```
print(comp$P2only[, 1:6], row.names=F)
 #    id dam.1 sire.1  dam.2 sire.2    id.r
 # M0033  <NA>   <NA> a02116 b02009 nomatch
 # M0034  <NA>   <NA>  F0022 b03005 nomatch
```

We can look a bit further into dummy male M0033:

```
print(comp$MergedPed[which(comp$MergedPed$sire.2 == "M0033"), 1:8], row.names=F)
 #     id  dam.1 sire.1  dam.2 sire.2   id.r  dam.r  sire.r
 # a04033 a03177 b03092 a03177  M0033 a04033   <NA> nomatch
```

```
# a04045 a03083 b03092  F0045  M0033 a04045 a03083 nomatch
# b04081 a03165 b03092 a03165  M0033 b04081   <NA> nomatch
# b04084 a03165 b03092 a03165  M0033 b04084   <NA> nomatch
```

From this, it M0033 looks like a perfectly good match with male b03092 in Pedigree 1. However, that male is already matched to dummy M0027, in which sibship more of its offspring are included:

```
print(comp$MergedPed[which(comp$MergedPed$sire.1 == "b03092"),
                     c("id", "sire.1", "sire.2", "sire.r")], row.names=F)
#    id sire.1 sire.2  sire.r
# a04033 b03092  M0033 nomatch
# a04034 b03092  M0027  b03092
# a04035 b03092  M0027  b03092
# a04036 b03092  M0027  b03092
# a04045 b03092  M0033 nomatch
# a04046 b03092  M0027  b03092
# a04082 b03092  M0027  b03092
# b04048 b03092  M0027  b03092
# b04081 b03092  M0033 nomatch
# b04083 b03092  M0027  b03092
# b04084 b03092  M0033 nomatch
#  M0010 b03092  M0027  b03092
```

Because of such cases, it is often useful to inspect the discrepancies between the pedigrees in some detail, to figure out exactly what is going on (and if the situation is really as dire as the `Counts` suggests).

6.2.3 Comparison of pairwise relationships

An alternative way to compare two pedigrees is by counting the number of pairs of full sibs in Pedigree 1, that are full sibs, half sibs, or unrelated in Pedigree 2. This is implemented in function `ComparePairs`:

```
comp.p  <- ComparePairs(Ped1 = Ped_HSg5, Ped2 = SeqXF$Pedigree,
                        GenBack = 1,
                        patmat = TRUE)  # distinguish between maternal & paternal relatives
comp.p
#      Ped2
# Ped1       M      P      O     FS    MHS    PHS      U      X
#   M      530      0      2      0      0      0     11    417
#   P        0    526      1      0      0      0     16    417
#   FS       3      0      3   1197     35     22    145    198
#   MHS      0      0      0      0   1342      0    200    218
#   PHS      0      0      0      0      0   2758    476    446
#   U        6      6     12      1     31     13 415419  75064
#   X        0      0      0      0      0      0      0      0
```

This shows for example that of the mothers (M) in Pedigree 1 (top row), 530 were correctly assigned (as in `comp$Counts["GG",, ]`), while in 2 cases their offspring was assigned as their parent (O), and in 11 instances were unrelated (U) to their true offspring when tracing only 1 generation back, so 'U' potentially includes assignments as aunt or

grandmother. When choosing `GenBack=2`, a larger table is generated, which does include avuncular relationships and grandparent.s

6.2.4 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)
```

6.3 Comparison pedigree-based and genomic relatedness

One way to compare two pedigrees is by comparing the pedigree-based relatednesses calculated from each. In R, pedigree relatedness can for example be calculated by the R package `kinship2` – relatedness is defined as twice the kinship coefficient.

`kinship2` requires a pedigree with columns id - dadid - momid - sex, and requires that individuals either have two parents, or zero:

```
Ped <- PedPolish(SeqOUT$Pedigree[, 1:3],
                 FillParents = TRUE)  # add fake single parents
Ped <- merge(Ped, MyLifeHistData, by.x="id", by.y="ID", all.x=TRUE)
library(kinship2)
# change sex coding: for kinship2 1=male, 2=female
Ped.fix <- with(Ped_, kinship2::fixParents(id=id, dadid=sire, momid=dam,
                                sex=c("female", "male", "unknown")[Sex]))
# create a pedigree object:
ped.k <- with(Ped.fix, kinship2::pedigree(id, dadid, momid, sex, missid=0))
# calculate kinship matrix:
kin <- kinship(ped.k)
# turn matrix into dataframe:
kin.df <- data.frame("IID1" = rep(rownames(kin.M), each=ncol(kin.M)),
                     "IID2" = rep(colnames(kin.M), times=nrow(kin.M)),
                     "k.ped" = c(t(kin.M)),
                     stringsAsFactors=FALSE)
# calculate pedigree relatedness:
kin.df$R.ped <- kin.df$k.ped *2
```

In absence of a previous pedigree, or when it is not obvious whether the previous or newly inferred pedigree is correct, one could 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`. 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("IID1", "IID2", "SNPS", "R.SNP")
Rel.snp <- Rel.snp[Rel.snp$IID1 != Rel.snp$IID2,]
```

```
#
library(data.table)
Rel.gt <- merge(data.table(Rel.snp[,c(1,2,4)], key=c("IID1", "IID2")),
                data.table(kin.df, key=c("IID1", "IID2")), all.x=TRUE)
Rel.gt <- as.data.frame(Rel.gt)  # turn back into regular dataframe
rm(PedStats, Rel.snp, kin.df)   # clean up large dataframes
#
round(cor(Rel.gt[, c("R.SNP", "R.ped")], 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)
#
# or, if you want to add a diagonal line to the plot:
hb <- hexbin(Rel.gt$R.SNP~Rel.gt$R.ped, xbins=100,
             xbnds=c(-.1,.9), ybnds=c(-.1, .9),
             xlab="Pedigree relatedness", ylab="Genomic relatedness")
hbp <- plot(hb, maxcnt=10^6.5, colramp=ColF, trans=log10,
           inv=function(x) 10^x, colorcut=seq(0,1,length=14))
hexVP.abline(hbp$plot.vp, a=0, b=1)
```

7 Other

7.1 Find non-assigned likely 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 incompatible, 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–grand-offspring 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 (parent-offspring), FS (full siblings), HA (3rd degree relative) or U (unrelated).

These pairs can be identified by 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.

7.2 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 3) 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 3: 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

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

7.3 Hermaphrodites

Hermaphrodites can be specified in `LifeHistData` with sex '4', which then 'under the hood' are treated 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`.

7.3.1 Pedigree prior

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 pedigree 'prior' can be provided as follows:

```
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)
```

Then, when two configurations are genetically equally likely, the one that matches the prior pedigree is assigned. Parent–offspring pairs in the pedigree prior that are not genetically a match will never be assigned.

Note that the functionality for hermaphrodites has not been as extensively tested as the rest of the program, and especially the option to cluster sibships has not been exhaustively tested - use with caution.

7.4 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`.

8 FAQ

8.1 Error messages when reading in data

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 that weren't caught by `CheckGeno` to give an informative error message. For very large datasets, specialist tools may be required to get the data in a standardised format.

8.2 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 ageprior. Sometimes *reducing* the number of SNPs may increase assignment rate, by removing those with the highest error rate and/or SNPs in close LD.

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 A few SNPs with a high (apparent) error rate may throw off pedigree reconstruction if their erroneous signal is not sufficiently offset out by a large number of more accurate SNPs. Such SNPs can be identified with function `SnpStats`, using an existing pedigree or one from an preliminary round of parentage assignment. If there are clear outliers with regards to estimated error rate, it may be worth exploring pedigree reconstruction without these SNPs.

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 complementary 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, can be identified with `GetMaybeRel`.

Insufficient age information A parent will only be assigned if it is known to be older than the individual with which it genetically forms a parent–offspring pair, or if a complementary co-parent is identified. Purely genetically, it is impossible to tell who the parent is in a parent–offspring pair. But once an individual has been assigned parents, any remaining individuals with which it forms a parent-offspring pair must be its offspring, and will be assigned as such. Thus, a high proportion of sampled parents may somewhat compensate for unknown birth years.

Providing minimum and maximum possible birth years can substantially increase assignment rate. The risk is that when intervals are taken too narrowly, parent–offspring pairs are flipped the wrong way around, 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.

Age information will also help to distinguish between the three different types of second degree relatives (half siblings, grandparent – grand-offspring and full avuncular (aunt/uncle – niece/nephew). These 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 only partial overlap of either with the age distribution of avuncular relationships.

Uninformative age prior Informative age priors may increase assignment rate when the power of the genetic data is limited, and to help distinguish between different types of second degree relatives. The ageprior used for full pedigree reconstruction is estimated from the birth year differences of parent-offspring pairs assigned during the initial parentage assignment. When only few parents are assigned, or when many birth years are unknown, this ageprior may 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.

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 insufficient to make the distinction, no assignment will be made.

When inbreeding and complex relationships (e.g. paternal half-sibling as well as maternal half-aunt) are rare, ignoring these typically increases assignment rate (`Complex="simp"`). Similarly, when polygamy is rare and not of particular interest, assuming a monogamous mating system typically increases assignment rate (`Complex="mono"`). However, these choices will risk erroneous assignments when complex relationships or polygamy, respectively, do occur.

8.3 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.

The only way in which a known death date can inform pedigree reconstruction by sequoia, is as an upper limit to the estimated birth year, when the exact birth year is unknown.

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

Field pedigree available 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 available When many thousands of SNPs are typed, it is possible to calculate the genomic relatedness (R_{grm}) between all pairs of individuals (see Section 6.3). 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 $R_{ped} - R_{grm} > 0.2$), 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 do 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.

8.5 I have genotyped additional individuals, how do I update the pedigree?

When new individuals have been genotyped, such as a new cohort of offspring, 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 they may have.

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.