Sequoia - Age 'Prior'

Jisca Huisman

16 Feb 2021

Contents

1 In short **2**
1.1 Age Info is Useful . . . 2
1.2 Implementation . . . 2
1.3 Time units . . . 3
1.4 Runtime Messages . . . 3
1.5 Changing the ageprior . . . 3

2 Maths **4**
2.1 AgePrior . . . 4
2.2 Unsampled individuals . . . 4
2.3 Genetics × Age . . . 5

3 AgePrior matrix explained **5**
3.1 `AgePriorExtra` . . . 6

4 Default ageprior **7**

5 Pedigree-based ageprior **8**
5.1 Discrete generations . . . 8
5.2 Overlapping generations . . . 9

6 Small sample correction **15**
6.1 The problem . . . 15
6.2 Smooth . . . 15
6.3 Flatten . . . 17

7 Grandparents & Avuncular **19**
7.1 Maths . . . 19

8 Customise **20**
8.1 Via `args.AP` . . . 20
8.2 Different input pedigree . . . 20
8.3 Other tweaks . . . 20
8.4 From age-difference distribution . . . 21
8.5 Specifications . . . 22

References **22**

1 In short

1.1 Age Info is Useful

The age difference between a pair of individuals can be useful information during pedigree reconstruction. It can exclude some relationship alternatives from consideration, as for example parent and offspring can never be born in the same time unit. It can also help to distinguish between the different types of second degree relatives, as the age difference between siblings is typically much smaller than between grandparent and grand-offspring.

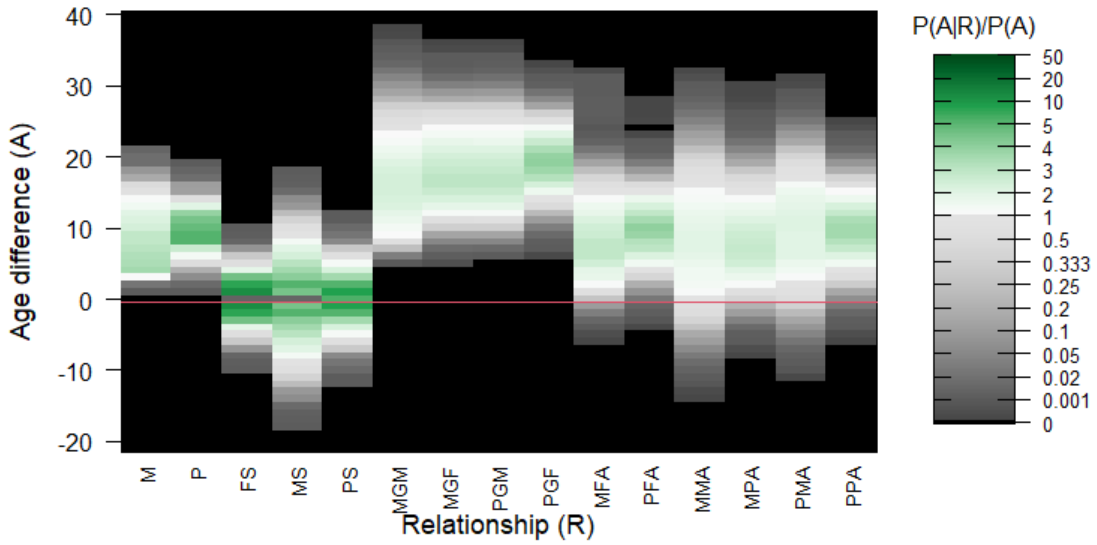


Figure 1: Example of 'ageprior' distribution in a red deer population

1.2 Implementation

While the rule that parent and offspring cannot have an age difference of 0 is universal, the pattern of age difference distributions for various relationships is highly species-specific, or even population-specific (example in Figure 1). Here `sequoia`'s separation between 1) assignment of genotyped parents to genotyped offspring and 2) full pedigree reconstruction including sibship clustering comes in handy. The age difference distribution for each relationship type is estimated based on the former, and then used as input for the latter (Figure 2).

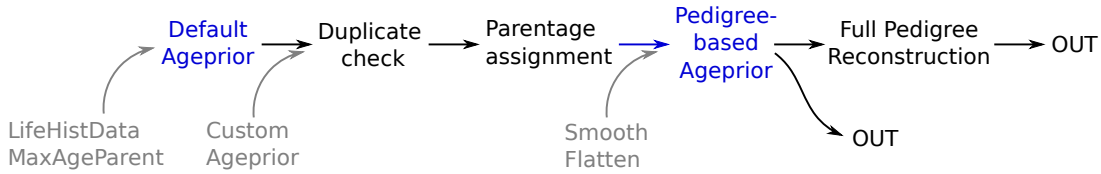


Figure 2: Pipeline overview

More specifically, age information is implemented via the ratio between the proportion of pairs with age difference A among those with relationship R, and the proportion in the entire sample (i.e. $P(A|R)/P(A)$). Scaling by the latter accounts for the fact that any sampling period is finite, so that smaller age differences are always more common than very large age differences.

Some reshuffling of terms (see Maths) shows that this probability ratio 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?

i.e. the probability of having relationship R conditional on the age difference A ($P(R|A)/P(R)$).

As a shorthand, these probability ratios are dubbed 'agepriors,' but please note that this is not an official term, nor a proper prior.

During full pedigree reconstruction, the genetic based likelihood that two individuals are related via relationship R is multiplied by this age based probability ratio. The latter may be 0 for some relationships, and then excludes these relationships from consideration. In other cases, it will 'nudge' the likelihoods up or down.

1.2.1 Passing the threshold

By default, age information is used in a restrictive way: a proposed assignment is only accepted if *both* the genetics-only and the genetics + age likelihood ratio (LLR) pass the assignment threshold. The genetics-only likelihoods do exclude those relationships that are impossible based on the age difference. You can change this behaviour to only the genetics LLR or only the genetics + age LLR having to pass the threshold via the `sequoia()` argument `UseAge`. See the section with that name in the main vignette for further details.

1.3 Time units

Throughout this documentation, 'birth year' is used to refer to the year/month/week/day/(other time unit) of birth/hatching/germination/... . The time units should be chosen such that parent and offspring can never be born in the same time unit, i.e. be smaller than or equal to the minimum age of maturity. `sequoia()` only accepts whole numbers as 'birth years.'

1.4 Runtime Messages

From `sequoia` version 2.0 onwards, you will see heatmaps and messages to inform you about the ageprior that is being used. These are for example

```
Ageprior: Default 0/1, overlapping generations, MaxAgeParent = 5,5

Ageprior: Pedigree-based, discrete generations, MaxAgeParent = 1,1

Ageprior: Pedigree-based, overlapping generations, smoothed, MaxAgeParent =
20,16
```

These are *not* warnings, but inform you on what is going on behind the scenes. This will hopefully minimise unintended age priors as a source of error during pedigree reconstruction. There are four parts to the message:

- `Default 0/1`, `Flat 0/1`, or `Pedigree-based` indicates whether the ageprior at that point is respectively the default based on lifehistory data only/no data; based on user-specified maximum parental age; or or based on the age-differences per relationship in a (scaffold or old) pedigree;
- `discrete` or `overlapping` generations
- `smoothed` and/or `flattened` indicate small-sample corrections for the pedigree-based age prior (no mention means it was not applied)
- Maximum age of mothers, fathers.

1.5 Changing the ageprior

During pedigree reconstruction, arguments can be passed to `MakeAgePrior()` via `sequoia()`'s `args.AP`. It is also possible to call `MakeAgePrior()` separately and use the resulting ageprior (via `sequoia()`'s `Seqlist` argument), or use a 'hand-tailored' ageprior (see Customisation).

Table 1: Terms and abbreviations

Symbol	Term	Definition
BY_i	Birth year	Time unit (year, decade, month) of i's birth/ hatching
$A_{i,j}$	Age difference	$BY_i - BY_j$, i.e. if j is older than i, than $A_{i,j}$ is positive
$R_{i,j}$	Relationship	Relationship between i and j, e.g. parent-offspring
$\alpha_{A,R}$	Ageprior	Probability ratio of relationship R given A, versus R for a random pair

2 Maths

2.1 AgePrior

The ageprior $\alpha_{A,R}$ (Glossary in Table 1) tells us how the probability that two individuals have relationship R changes when learning their age difference A, or

$$\alpha_{A,R} = \frac{P(R|A)}{P(R)} \ . \tag{1}$$

Using Bayes' rule that

$$P(R|A) = \frac{P(A|R)P(R)}{P(A)} \tag{2}$$

it follows that, if $P(A)$ and $P(R)$ are independent,

$$\alpha_{A,R} = \frac{P(R|A)}{P(R)} = \frac{P(A|R)}{P(A)} \ . \tag{3}$$

When a pedigree and birth years are known, $P(A)$ and $P(A|R)$ are estimated as the observed proportions of individual pairs with age-difference A among all pairs ($P(A)$) and among parent-offspring and sibling pairs with relationship R ($P(A|R)$) (see worked-out example in Section Overlapping generations).

2.2 Unsampled individuals

When using the scaffold parentage-only pedigree as input, α_{A_R} is based only on pairs which are both genotyped and which both have a known birth year. During full pedigree reconstruction, we assume that the association between age difference and relationship probability is roughly the same for pairs where either or both individuals are unsampled (i.e. dummy individuals), or for whom the birthyear is unknown:

$$\frac{P(R|A, \text{sampled})}{P(R|\text{sampled})} \approx \frac{P(R|A, \text{unsampled})}{P(R|\text{unsampled})} \tag{4}$$

This does *not* mean that sampling probability is assumed to be independent of A: any sampling period is finite, thus there are always fewer both-sampled pairs with large age differences than with small age differences ($P(A|sampled) \neq P(A|unsampled)$). It *does* assume that the probability to both be sampled is independent of relationship R ($P(A|sampled) \approx P(A|unsampled)$), so that

$$\frac{P(A|R, \text{sampled})}{P(A|\text{sampled})} \approx \frac{P(A|R, \text{unsampled})}{P(A|\text{unsampled})}, \tag{5}$$

i.e. it is assumed that α_{A_R} calculated from both-sampled pairs is valid for pairs where one or both are unsampled.

This assumption is relaxed via the small-sample correction.

2.3 Genetics × Age

During pedigree reconstruction, the ageprior $\alpha_{A,R}$ is multiplied by the genotypes-depended probability,

$$P(R_{i,j}|G_i, G_j, A_{i,j}) = P(R_{i,j}|G_i, G_j) \times \alpha_{R_{i,j},A_{i,j}} \ , \tag{6}$$

as it can be reasonably assumed that conditional on the relationship R, the genotypes G and age difference A are independent (at least on the time scales relevant for pedigree reconstruction). The calculation of $P(R_{i,j}|G_i, G_j)$ is described in detail in Huisman (2017).

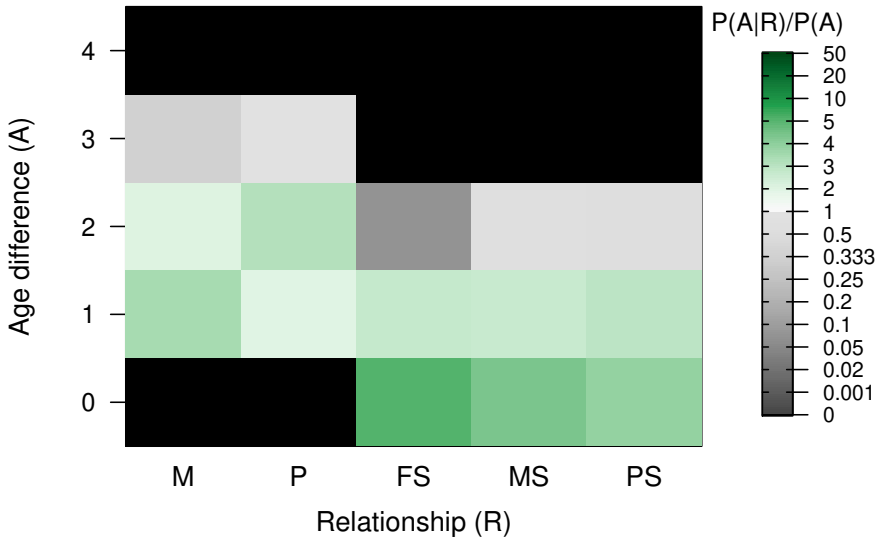
For a pair of individuals, $P(R_{i,j}|G_i, G_j)$ in (6) for various relationships can be obtained with `CalcPairLL()`, and $\alpha_{R_{i,j},A_{i,j}}$ with `GetGetLLRAge()`.

3 AgePrior matrix explained

The ageprior matrix has 5 columns (more in sequoia versions before 2.0), corresponding to parent-offspring and sibling pairs (abbreviations in Table 2). The number of rows equals at least the maximum age of parents (`MaxAgeParent`) plus one: the first row is for individuals born in the same year (age difference $A = 0$), the second row for individuals born one year apart, etc (indicated by the rownames).

```
MakeAgePrior(Pedigree = Ped_griffin, Smooth=FALSE, Flatten=FALSE)  # Note: Ped_griffin includes a birth
```

```
## Ageprior: Pedigree-based, overlapping generations, MaxAgeParent = 3,3
```




```
##       M     P    FS    MS    PS
## 0 0.000 0.000 5.043 4.316 3.862
## 1 3.476 1.967 2.798 2.636 2.971
## 2 2.011 3.167 0.077 0.691 0.583
## 3 0.340 0.959 0.000 0.000 0.000
## 4 0.000 0.000 0.000 0.000 0.000
```

A value of 0 (black squares in the plot) indicates that an age difference / relationship combination is impossible; in the example here this is the case for among others mother-offspring pairs (`M`, first column) and age differences of 0 and 4. Any age differences not in this matrix (i.e. >4 years) are considered impossible (0) for all five relationships. Values greater than 1 (green) indicate a relationship is more likely for that age difference than for a random pair, while values below 1 (grey) indicate the relationship is less likely.

Table 2: AgePriors column names per sequoia version. *: From sequoia v2.0 only in AgePriorExtra

Column (R)	Meaning	Version
M	Mother - offspring	all
P	Father - offspring	all
FS	Full siblings	from 1.0
MS	Maternal siblings (full + half)	all
PS	Paternal siblings (full + half)	all
MGM	Maternal grandmother	up to 1.3 *
PGF	Paternal grandfather	up to 1.3 *
MGF	Maternal grandfather (+ paternal grandmother)	up to 1.3 *
UA	avuncular (niece/nephew – aunt/uncle, full + half)	up to 1.3
(M/P)(M/P/F)A	Avuncular; Mother's/Father's Maternal/Paternal/Full sibling	from 2.0 *

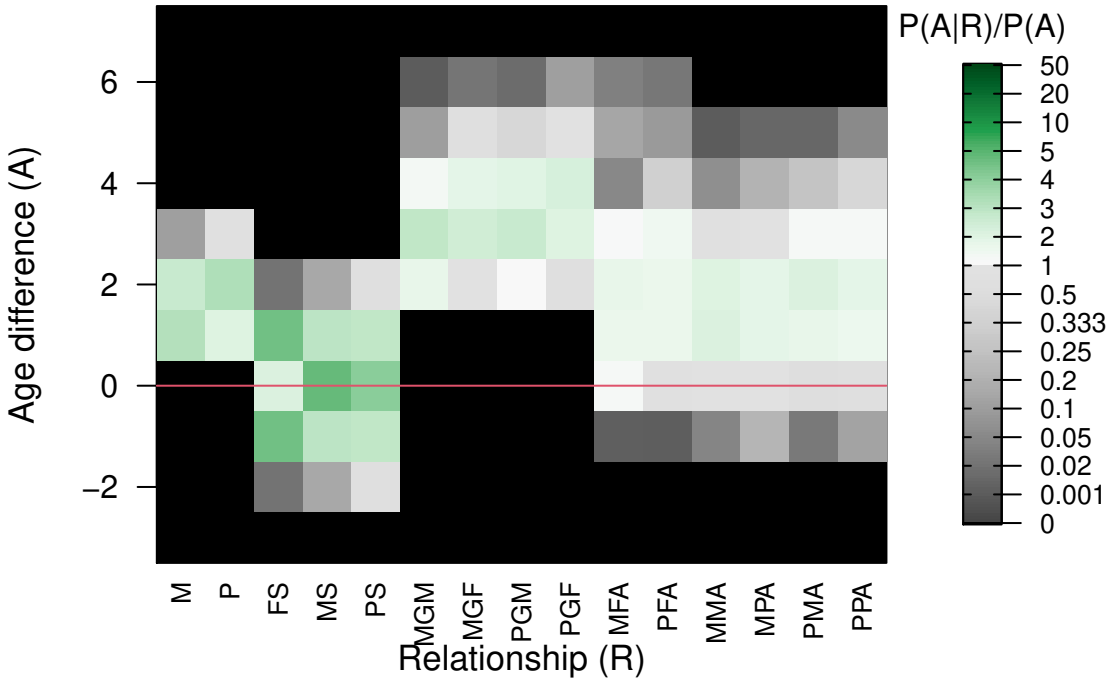
3.1 `AgePriorExtra`

The agepriors for grand-parental en avuncular pairs are calculated from the agepriors for parent-offspring and sibling relationships (details in section Grandparents & Avuncular). These extra columns are not calculated by `MakeAgePrior()`, but by `sequoia()` just prior to parentage assignment and prior to full pedigree reconstruction (see pipeline in Figure 2).

```
# sequoia() output for the griffin data is included in the package:
# see ?SeqOUT_griffin on how it was generated
round(SeqOUT_griffin$AgePriorExtra, 2)
```

```
##       M    P   FS   MS   PS  MGM  MGF  MFA  MMA  MPA  PGM  PGF  PFA  PMA  PPA
## -3 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
## -2 0.00 0.00 0.02 0.16 0.74 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
## -1 0.00 0.00 4.51 2.92 2.81 0.00 0.00 0.01 0.05 0.21 0.00 0.00 0.00 0.03 0.13
## 0  0.00 0.00 2.13 4.77 4.01 0.00 0.00 1.23 0.90 0.99 0.00 0.00 0.76 0.58 0.72
## 1  3.16 2.02 4.51 2.92 2.81 0.00 0.00 1.66 2.12 1.85 0.00 0.00 1.66 1.74 1.59
## 2  2.66 3.28 0.02 0.16 0.74 1.71 0.98 1.78 2.07 1.81 1.06 0.61 1.69 2.16 1.87
## 3  0.11 0.81 0.00 0.00 0.00 2.88 2.49 1.11 0.82 0.94 2.62 2.06 1.46 1.24 1.24
## 4  0.00 0.00 0.00 0.00 0.00 1.33 1.88 0.05 0.07 0.21 1.91 2.24 0.33 0.26 0.42
## 5  0.00 0.00 0.00 0.00 0.00 0.10 0.64 0.14 0.00 0.01 0.42 0.99 0.09 0.01 0.06
## 6  0.00 0.00 0.00 0.00 0.00 0.00 0.03 0.04 0.00 0.00 0.01 0.11 0.03 0.00 0.00
## 7  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

```
PlotAgePrior(SeqOUT_griffin$AgePriorExtra)
```



The ageprior for avuncular pairs is complicated: aunts and uncles are often older than their nieces/nephews, but may also be younger, and the distribution is not necessarily symmetrical around zero. To accommodate this, age differences in `AgePriorExtra` are no longer absolute age differences, but do go negative. Parents and grandparents must always be older than their (grand-)offspring, and the distribution for siblings is always strictly symmetrical.

4 Default ageprior

The default ageprior used during parentage assignment is flat and minimally informative: it only specifies that parent and offspring cannot have an age difference of 0 ($\alpha_{A,R} = 0$ for $A = 0$ and $R = M$ or $R = P$, black in Figure 3) and that all other age / relationship combinations are allowed ($\alpha_{A,R} = 1$, pale green).

```
MakeAgePrior(LifeHistData = LH_HSg5)
```

```
## Ageprior: Flat 0/1, overlapping generations, MaxAgeParent = 6,6
```

```
##   M P FS MS PS
## 0 0 0  1  1  1
## 1 1 1  1  1  1
## 2 1 1  1  1  1
## 3 1 1  1  1  1
## 4 1 1  1  1  1
## 5 1 1  1  1  1
## 6 1 1  0  0  0
## 7 0 0  0  0  0
```

The range of possible age differences is taken from `LifeHistData`:

```
table(LH_HSg5$BirthYear)
```

```
##
## 2000 2001 2002 2003 2004 2005
##   40  192  192  192  192  192
```

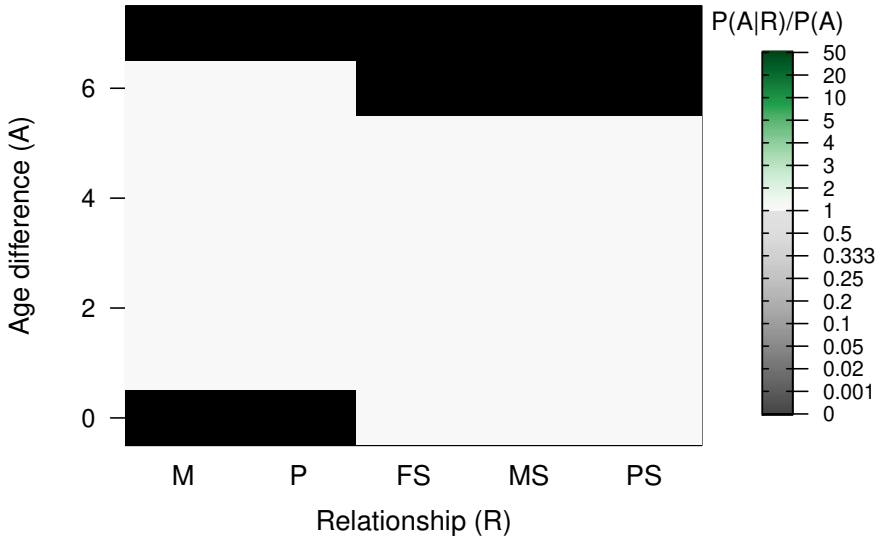


Figure 3: Default minimally informative ageprior

The maximum age of parents `MaxAgeParent` is initially set to the maximum age difference possible among individuals in `LifeHistData` (here 2005-2000 = 5 years) +1. This ensures that all pairs are age-wise considered as potential siblings;[1] the age difference between siblings can never be larger than `MaxAgeParent` −1.

5 Pedigree-based ageprior

5.1 Discrete generations

When a pedigree is provided as input, `MakeAgePrior()` always checks whether generations are non-overlapping, i.e. whether all parents have the same the same age (typically 1), and all siblings have an age difference of 0. If these conditions are met, and there are at least 20 pairs each of mother-offspring, father-offspring, maternal half/full siblings, and paternal half/full siblings, it is concluded that generations are discrete. This auto-detection can be turned off by explicitly declaring `Discrete=TRUE` or `Discrete=FALSE`. If `Discrete=TRUE` is declared, a check is still performed, and any violations result in an error. When discrete generations are automatically inferred or explicitly declared, small-sample corrections `Smooth` and `Flatten` are always set to `FALSE`.

When `Discrete = TRUE` is specified prior to parentage assignment, being full siblings is no longer considered as a potential relationship alternative for candidate parents (age difference of 1), but being an aunt/uncle is (Figure 4). When the genetic information is ambiguous (few SNPs and/or high genotyping error rate) this may increase the assignment rate, but it may also increase the number of incorrect assignments.

Similarly, during full pedigree reconstruction, pairs with an age difference of 0 are considered potentially full siblings, half siblings, cousins (not shown in ageprior plot), or unrelated. In contrast, with overlapping generations they are (typically) also considered as potential full avuncular pairs, which are often difficult to distinguish from half-sibling pairs.[2]

```
# running sequoia to also get the extra ageprior columns:
Seq_HSg5 <- sequoia(SimGeno_example, LH_HSg5, Module="par",
                    args.AP=list(Discrete = TRUE),
```

[1] Up to version 2.0, initial `MaxAgeParent` was set to `diff(BYrange)`, but sibling relationships were also allowed at this age difference, so this change (probably) won't affect parentage assignment or full pedigree reconstruction.

[2] Both individuals need to have already at least one parent assigned to be able to genetically distinguish between half-siblings, full avuncular, and grandparent–grand-offspring

```
                    CalcLLR=FALSE, Plot=FALSE, quiet=TRUE)
PlotAgePrior(Seq_HSg5$AgePriorExtra)
```

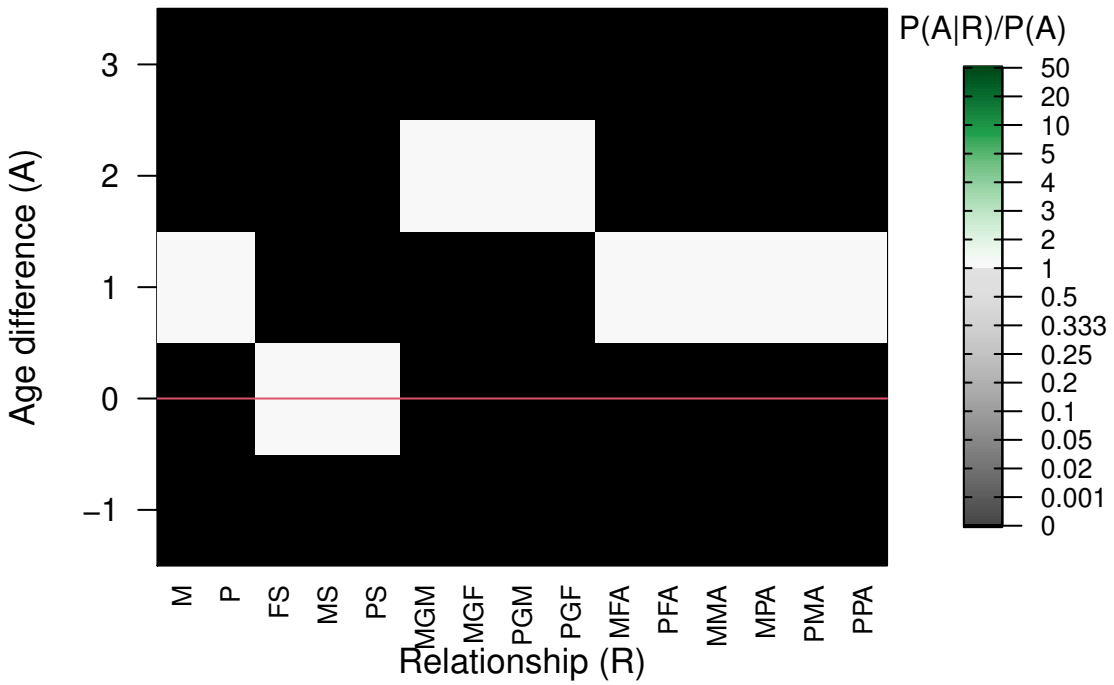


Figure 4: Discrete generations

5.1.1 Generation interval different from 1

When generations do not overlap, it is often most straightforward to use generation number instead of birth year, such that parents and offspring are born 1 time unit apart. This is assumed to be the case when `Discrete=TRUE` is specified, but can be altered via `MaxAgeParent` (which in this situation is the *only* allowed age for parents). For example, pink salmon have a strict two-year life-cycle with odd-year and even-year populations:

```
MakeAgePrior(Discrete = TRUE, MaxAgeParent = 2, Plot=FALSE)
```

```
## Ageprior: Flat 0/1, discrete generations, MaxAgeParent = 2,2
```

```
##   M P FS MS PS
## 0 0 0  1  1  1
## 1 0 0  0  0  0
## 2 1 1  0  0  0
## 3 0 0  0  0  0
```

5.2 Overlapping generations

Estimation of the agepriors from a pedigree and birth years normally requires little to no user input, but it can be useful to understand what goes on 'under the hood' in case you wish to customise the ageprior, when things go wrong, or just out of curiosity.

The process is here illustrated using an imaginary population of griffins, were each year exactly 20 baby griffins hatch. From 2001 to 2010, all juveniles are sampled, and parents assigned.

```
tail(Ped_griffin)      # the pedigree
```

```
##              id         dam        sire birthyear
## 195 i195_2010_M i157_2008_F i163_2009_M      2010
## 196 i196_2010_M i148_2008_F i127_2007_M      2010
## 197 i197_2010_F i170_2009_F i127_2007_M      2010
## 198 i198_2010_M i166_2009_F i178_2009_M      2010
## 199 i199_2010_F i165_2009_F i141_2008_M      2010
## 200 i200_2010_F i166_2009_F i142_2008_M      2010
```

We can calculate the ageprior matrix, and also return all the intermediate steps by specifying `Return='all'` (we come back to `Smooth` and `Flatten` later):

```
AP.griffin <- MakeAgePrior(Ped_griffin, Smooth=FALSE, Flatten=FALSE,
                           Return="all")
```

```
## Ageprior: Pedigree-based, overlapping generations, MaxAgeParent = 3,3
```

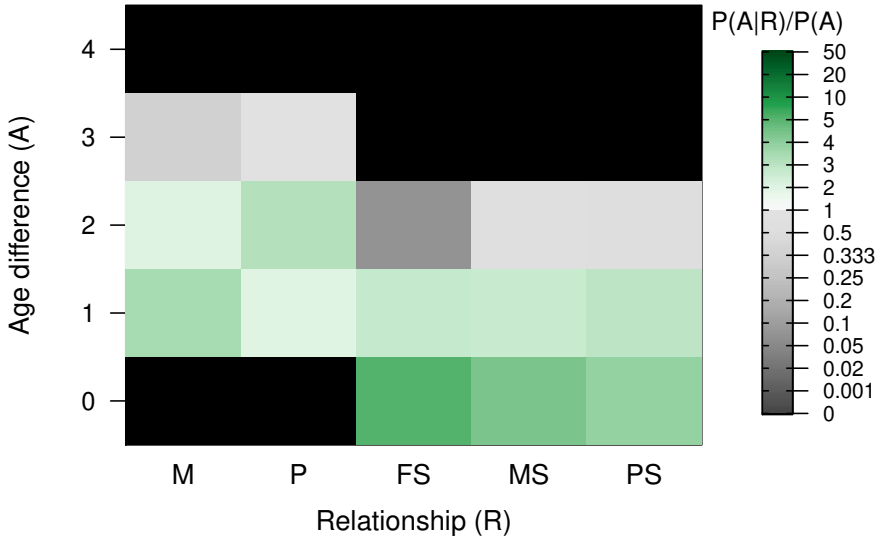


Figure 5: Ageprior for overlapping generations (griffin example)

```
names(AP.griffin)
```

```
## [1] "BirthYearRange" "MaxAgeParent"   "tblA.R"         "PA.R"
## [5] "LR.RU.A.raw"    "Weights"        "LR.RU.A"        "Specs.AP"
```

The order of elements in the output list is also the order in which they are calculated inside `MakeAgePrior()`.

5.2.1 `BirthYearRange`

`BirthYearRange` is the minimum and maximum birth year in `LifeHistData` (or in the `birthyear` column in the pedigree). This determines the maximum age difference between pairs of (sampled) individuals.

```
AP.griffin$BirthYearRange
```

```
## [1] 2001 2010
```

5.2.2 `MaxAgeParent`

The maximum age of female and male parents, `MaxAgeParent`, is initially set equal to the maximum age difference between any kind of pairs (`diff(BirthYearRange)`) +1 (see Default Ageprior), when not explicitly

specified as input. If the pedigree includes at least 20 mother-offspring and/or father-offspring pairs with known age difference, the maximum parental age for that sex is updated to the maximum of the user-specified and pedigree-calculated values.

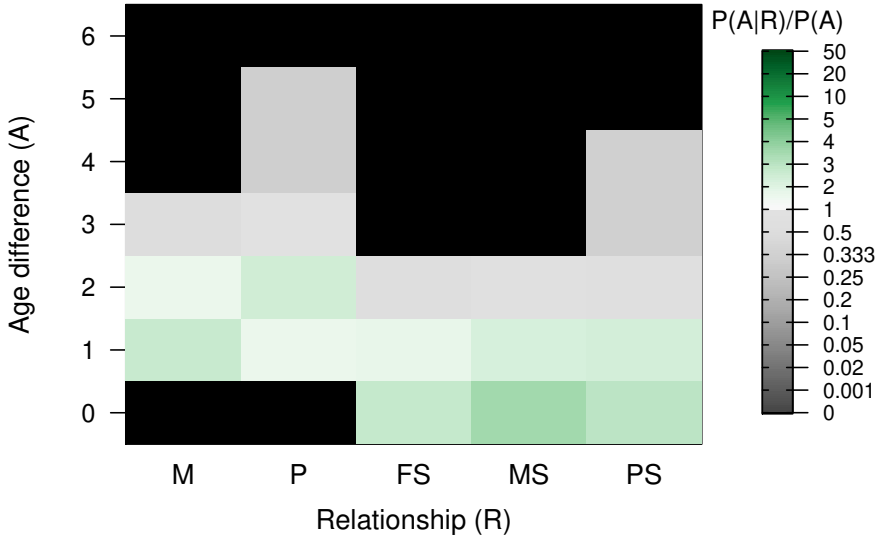
```
AP.griffin$MaxAgeParent
```

```
## M P
## 3 3
```

For example, if you know that males can live and breed to age 5, but you have only sampled males up to age 3, you can and should adjust the age prior – this will namely also affect the maximum age difference between paternal siblings during the full pedigree reconstruction. This adjustment works via `Flatten`.

```
# Specify maximum age for fathers, but estimate max mother age from pedigree
APX <- MakeAgePrior(Ped_griffin, Smooth=FALSE, MaxAgeParent = c(NA, 5))
```

```
## Ageprior: Pedigree-based, overlapping generations, flattened, MaxAgeParent = 3,5
```



5.2.3 Counts: `tblA.R`

The list element `tblA.R` contains the raw counts of each age difference A per relationship R:

```
AP.griffin[["tblA.R"]]
```

```
##       M  P FS  MS PS    X
## 0     0  0  3  89 59 1900
## 1   105 58  2 103 86 3600
## 2    54 83  0  24 15 3200
## 3     8 22  0   0  0 2800
## 4     0  0  0   0  0 2400
## 5     0  0  0   0  0 2000
## 6     0  0  0   0  0 1600
## 7     0  0  0   0  0 1200
## 8     0  0  0   0  0  800
## 9     0  0  0   0  0  400
## 10    0  0  0   0  0    0
## 11    0  0  0   0  0    0
```

For this table, the relationship between each pair of individuals according to the pedigree is obtained with `GetRelM()`, and a matrix with the age differences between all pairs is calculated using `outer()`:

```
AgeDifM <- outer(Ped_griffin$birthyear, Ped_griffin$birthyear, "-")
# closest relationship for each pair:
RelM <- GetRelM(Ped_griffin, patmat=TRUE, GenBack=1)
table(c(AgeDifM), c(RelM))  # vectorising first speeds up table() a lot
```

```
##
##        FS    M  MHS    O    P  PHS    S    U
##   -9    0    0    0    0    0    0    0  400
##   -8    0    0    0    0    0    0    0  800
##   -7    0    0    0    0    0    0    0 1200
##   -6    0    0    0    0    0    0    0 1600
##   -5    0    0    0    0    0    0    0 2000
##   -4    0    0    0    0    0    0    0 2400
##   -3    0    0    0   30    0    0    0 2770
##   -2    0    0   24  137    0   15    0 3024
##   -1    2    0  101  163    0   83    0 3251
##   0     6    0  172    0    0  112  200 3510
##   1     2  105  101    0   58   83    0 3251
##   2     0   54   24    0   83   15    0 3024
##   3     0    8    0    0   22    0    0 2770
##   4     0    0    0    0    0    0    0 2400
##   5     0    0    0    0    0    0    0 2000
##   6     0    0    0    0    0    0    0 1600
##   7     0    0    0    0    0    0    0 1200
##   8     0    0    0    0    0    0    0  800
##   9     0    0    0    0    0    0    0  400
```

Some 'polishing' of this table is required, as each pair of individuals is included twice:

```
# - 'MaxT' ensures sufficient, constant number of rows across intermediary tables
MaxAgeParent <- c(NA, NA)
MaxAgePO <- ifelse(!is.na(MaxAgeParent), MaxAgeParent, diff(AP.griffin$BirthYearRange)+1)  # in actual
MaxT <- max(MaxAgePO+1, diff(AP.griffin$BirthYearRange))

# - factor() ensures levels with 0 count are included
# - drop negative age differences: all pairs are included 2x
tblA.R <- table(factor(AgeDifM, levels = 0:MaxT),
                factor(RelM, levels = c("M", "P", "FS", "MHS", "PHS")))
```

And a further improvement by using every relationship for each pair, rather than only their closest relationship (a pair may e.g. be both mother-offspring & paternal half-siblings):

```
RelA <- GetRelM(Ped_griffin, patmat=TRUE, GenBack=1, Return="Array")
tblA.R <- sapply(c("M", "P", "FS", "MHS", "PHS"),
                 function(r) table(factor(AgeDifM[RelA[,,r]==1],
                                          levels = 0:MaxT)))
head(tblA.R)
```

```
##     M  P FS MHS PHS
## 0   0  0  6 172 112
## 1 105 58  2 101  84
## 2  54 83  0  24  15
## 3   8 22  0   0   0
## 4   0  0  0   0   0
## 5   0  0  0   0   0
```

The ageprior is for 'maternal siblings,' which includes both maternal half-siblings and full siblings, and similarly for paternal siblings. Lastly, a column with the total count of pairs per age difference is added (i.e. the denominator in the calculation of $\alpha_{A,R}$. For this reference age-difference distribution all individuals in the pedigree with known birth year in `LifeHistData` are used.

```
tblA.R <- cbind(tblA.R,
                  "MS" = tblA.R[,"MHS"] + tblA.R[,"FS"],
                  "PS" = tblA.R[,"PHS"] + tblA.R[,"FS"])
tblA.R <- tblA.R[, c("M", "P", "FS", "MS", "PS")]  # drop MHS & PHS columns


# add column for age diff. across all relationships (except 'Self' pairs)
tblA.R <- cbind(tblA.R,
                "X" = table(factor(AgeDifM[c(RelM)!="S"], levels = 0:MaxT)))

# All pairs are included 2x in AgeDifM/RelM/RelA, to fix this:
# - drop the negative age differences  (done with factor() above)
# - divide counts for row A=0 by 2  (including 'X' column)
tblA.R["0", ] <- tblA.R["0", ] / 2

# check that this is really the output from MakeAgePrior():
all(tblA.R == AP.griffin[["tblA.R"]])
```

```
## [1] TRUE
```

5.2.4 P(A|R): `PA.R`

The probability of age difference A conditional on relationship R is estimated as the proportion of pairs with relationship R and known age difference, that have age difference A.

```
NAK.R <- colSums(tblA.R, na.rm=TRUE)  # total count per relationship
NAK.R
```

```
##     M     P    FS    MS    PS     X
##   167   163     5   216   160 19900
```

```
PA.R <- sweep(tblA.R, 2, STATS=NAK.R, FUN="/")
head(round(PA.R, 2))
```

```
##      M    P  FS   MS   PS    X
## 0 0.00 0.00 0.6 0.41 0.37 0.10
## 1 0.63 0.36 0.4 0.48 0.54 0.18
## 2 0.32 0.51 0.0 0.11 0.09 0.16
## 3 0.05 0.13 0.0 0.00 0.00 0.14
## 4 0.00 0.00 0.0 0.00 0.00 0.12
## 5 0.00 0.00 0.0 0.00 0.00 0.10
```

Pairs where one or both individuals have an unknown age difference, are excluded from both the numerator and the denominator. It is assumed that the age difference distribution of these pairs of relatives follows the same distribution as for relatives for which the age difference *is* known (Section Unsampled individuals).

5.2.5 Full sibling correction

Sometimes full sibling pairs are rare, causing an imprecise estimate of their age-difference distribution. A reasonable assumption is that the distribution for full siblings is broadly similar to "happens to be both

maternal and paternal sibling."[3] If many half-sibling pairs have been sampled, their age distribution is used to improve the estimate of $P(A|FS)$ (this is not optional):

```
MinPairs.AgeKnown <- 20   # not user-setable.
# Use the product of the distributions if there are many MS & PS,
# and the average across the two distributions if there are a medium number:
if (NAK.R["FS"] / min(NAK.R[c("MS", "PS")]) < 0.5 &
    all(NAK.R[c("MS", "PS")] > MinPairs.AgeKnown)) {
  if (all(NAK.R[c("MS", "PS")] > 5*MinPairs.AgeKnown)) {
    FS.tmp <- PA.R[,"MS"] * PA.R[,"PS"]
    FS.tmp <- FS.tmp/sum(FS.tmp)
  } else {
    FS.tmp <- apply(PA.R[, c("MS", "PS")], 1, mean)
  }
  PA.R[,"FS"] <- (PA.R[,"FS"] + FS.tmp)/2
}

round(PA.R[1:5, ], 2)  # note change in 'FS' column
```

```
##      M    P   FS   MS   PS    X
## 0 0.00 0.00 0.48 0.41 0.37 0.10
## 1 0.63 0.36 0.51 0.48 0.54 0.18
## 2 0.32 0.51 0.01 0.11 0.09 0.16
## 3 0.05 0.13 0.00 0.00 0.00 0.14
## 4 0.00 0.00 0.00 0.00 0.00 0.12
```

5.2.6 Ageprior P(A|R)/P(A): `LR.RU.A.raw`

The age-difference based probability ratio $\alpha_{A,R}$ is calculated by dividing $P(A|R)$ (each relationship column) by $P(A)$ (the `X` column):

```
LR.RU.A <- sweep(PA.R[, 1:5], 1, STATS=PA.R["X"], FUN="/")
head(round(LR.RU.A, 2))
```

```
##      M    P   FS   MS   PS
## 0 0.00 0.00 5.04 4.32 3.86
## 1 3.48 1.97 2.80 2.64 2.97
## 2 2.01 3.17 0.08 0.69 0.58
## 3 0.34 0.96 0.00 0.00 0.00
## 4 0.00 0.00 0.00 0.00 0.00
## 5 0.00 0.00 0.00 0.00 0.00
```

```
# check that this is identical to output from MakeAgePrior():
LR.RU.A[!is.finite(LR.RU.A)] <- 0
all(abs(LR.RU.A  - AP.griffin[["LR.RU.A.raw"]]) < 0.001)  # allow for rounding errors
```

```
## [1] TRUE
```

This is the non-flattened, non-smoothed ageprior, stored in list element `[["LR.RU.A.raw"]]`. Since in this example `Smooth=FALSE` and `Flatten=FALSE`, it is identical to list element `[["LR.RU.A"]]`, which is the default output (`Return='LR'`).

[3] This is not necessarily the case. For example, in species that produce multiple offspring each year, and breed for multiple years, within-year (within-clutch) siblings are more likely to be full siblings than between-year siblings

6 Small sample correction

Calculating the proportion of pairs with relationship R and age difference A in a pedigree is reasonably straightforward, but generating a matrix that is useful for subsequent pedigree reconstruction under a wide range of 'non-ideal' situations is not.

6.1 The problem

A major challenge is that when an age/relationship combination is not observed in the input pedigree, its ageprior value will be 0, and without correction that age/relationship combination will be deemed impossible during full pedigree reconstruction. For example, if mother-offspring pairs with age differences of 1, 2, 3, and 5 years are observed, it would infer that females never breed at age 4. This seems highly unlikely from a biological perspective, and is much more likely a sampling artefact. This does matter during full pedigree reconstruction: if the true maternal grandmother is 8 years older than one of her grandchildren, she would not be assigned, because mothers cannot be age 4.

As another example, imagine that no candidate fathers at all were sampled. Based on the parentage-only scaffold pedigree, the counts in the father column of `tblA.R` would be 0 for all age differences. Consequently, all age differences would be disallowed for father-offspring pairs during subsequent full pedigree reconstruction, including for dummy fathers, and no paternal sibships would be clustered.

Incorrect agepriors can also lead to false positives: when it is wrongly believed the age difference of a genetic half-sib pair makes it impossible for them to be maternal siblings, they will likely be assigned as paternal half-siblings. More individuals may then be added to this erroneous paternal sibship, potentially setting off a snowball effect of wrong assignments.

To minimise these problems, `MakeAgePrior()` applies two generic, one-size-fits-most corrections: a `Smooth`ing of dips and padding of the tails, and a `Flatten`ing approach whereby a weighed average of the pedigree-based and flat default ageprior is taken (Figure 6).

6.2 Smooth

When `Smooth=TRUE`, the tails of the distributions are 'stretched' and any dips are smoothed over. It defaults to `TRUE`, as there is only rarely a detrimental effect of smoothing when none is required. Exception is when generations are discrete; when this is detected or declared by the user, `Smooth` is set to `FALSE` automatically.

A 'dip' is defined as a value that is less than 10% of the average of its neighbouring values (age differences of one year less/more), and both neighbours have values > 0. It is then changed to the average of the two neighbours. Peaks are not smoothed out, as these are less likely to cause problems than dips, and are more likely to be genuine characteristics of the species.

The front & end tails of the distribution are altered as follows:

```
SmoothAP <- function(V, MinP) {
  Front <- max(1, min(which(V > MinP)), na.rm=TRUE)
  End <- min(max(which(V > MinP)), length(V), na.rm=TRUE)
  if (Front > 1 & V[Front] > 3*MinP)    V[Front -1] <- MinP
  if (End < length(V))                                V[End +1] <- V[End]/2
  if ((End+1) < length(V) & V[End+1] > 3*MinP)  V[End+2] <- MinP
  # ... dip-fixing ...
  V
}

LR.RU.A <- apply(LR.RU.A, 2, SmoothAP, MinP = 0.001)
```

There is no way to change the behaviour of `Smooth`, except turning it on or off. If for example a longer tail is required at either end, or more thorough smoothing is needed, you will need to edit the ageprior matrix yourself (see Customisation).

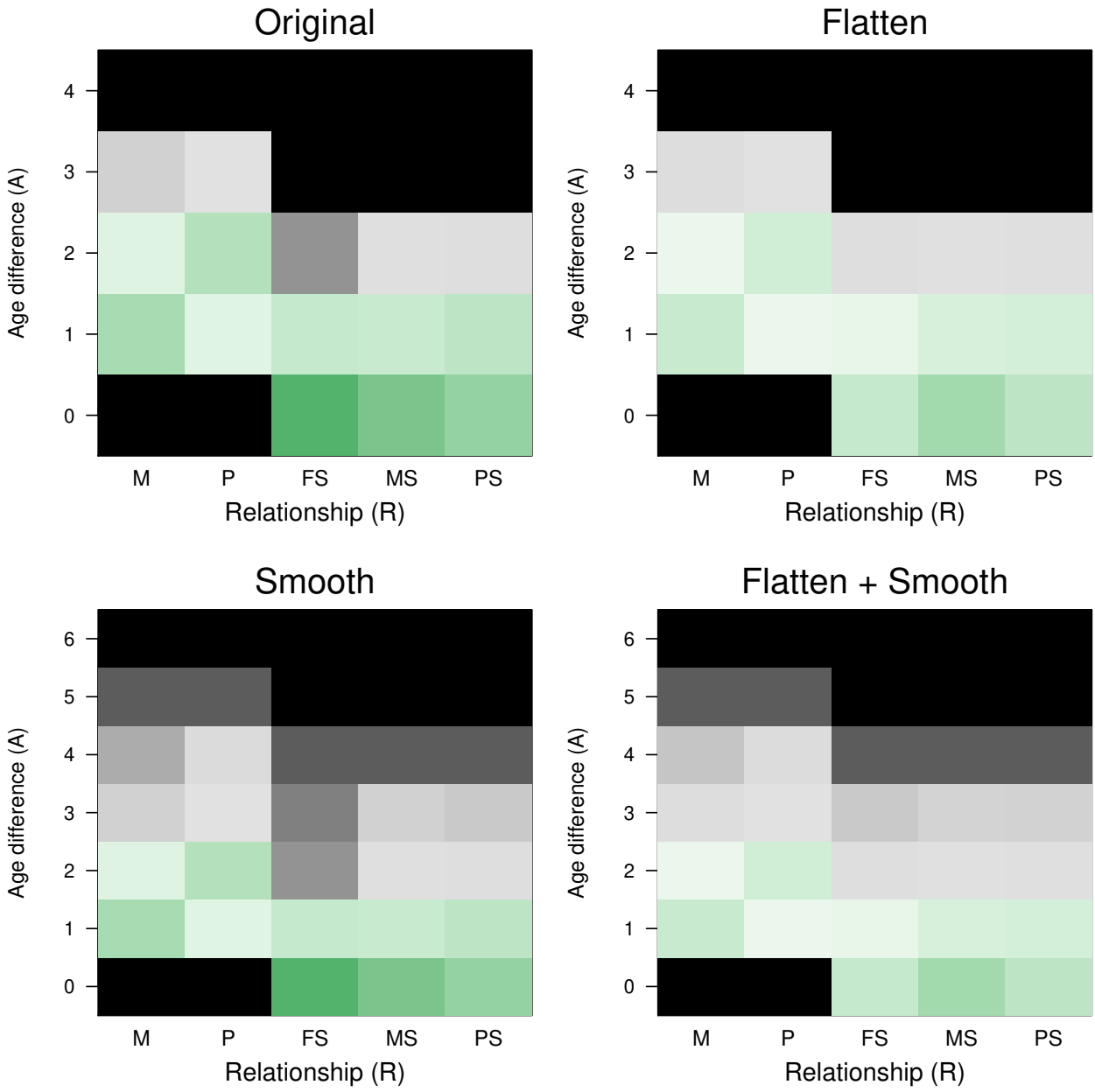


Figure 6: Smoothed & Flattened agepriors for overlapping generations (griffin example). Note the different y-axes; non-displayed age differences have an ageprior of zero for all R.

Table 3: Pedigree-based ageprior (left) and default, flat ageprior (right).

	M	P	FS	MS	PS		M	P	FS	MS	PS
0	0.000	0.000	5.043	4.316	3.862	0	0	0	1	1	1
1	3.476	1.967	2.798	2.636	2.971	1	1	1	1	1	1
2	2.011	3.167	0.077	0.691	0.583	2	1	1	1	1	1
3	0.340	0.959	0.000	0.000	0.000	3	1	1	0	0	0
4	0.000	0.000	0.000	0.000	0.000	4	0	0	0	0	0

Table 4: Flattened ageprior

	M	P	FS	MS	PS
0	0.000	0.000	2.761	3.574	2.918
1	2.698	1.655	1.783	2.270	2.321
2	1.693	2.467	0.598	0.760	0.721
3	0.548	0.972	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000

6.3 Flatten

The ageprior is 'flattened' by taking a weighed average between the ageprior based on the pedigree and birth years, and a flat default ageprior based on the maximum age of parents only (user-specified, pedigree-estimated, or maximum age difference in `LifeHistData`):

```
AP.pedigree <- MakeAgePrior(Ped_griffin, Smooth=FALSE, Flatten=FALSE,
                            quiet=TRUE, Plot=FALSE)
AP.default <- MakeAgePrior(MaxAgeParent = 3, quiet=TRUE, Plot=FALSE)

knitr::kable(list(AP.pedigree, AP.default),
             caption = "Pedigree-based ageprior (left) and default, flat ageprior (right).",
             booktabs=TRUE)
```

$$\alpha*_{A,R} = W_R \times \frac{P(A|R, \text{sampled})}{P(A, \text{sampled})} + (1 - W_R) \times I_{A,R} \tag{7}$$

where W_R is a weight based on N_R, the number of pairs with relationship R and known age difference. $I_{A,R}$ is an indicator whether the age-relationship combination is possible (1) or not (0) based on the maximum parental age (i.e., the flat default ageprior). $\alpha_{A,R} = P(A|R, \text{sampled})/P(A, \text{sampled})$ is the raw pedigree-based ageprior.

```
AP.flattened <- MakeAgePrior(Ped_griffin, Smooth=FALSE, Flatten=TRUE, quiet=TRUE, Plot=FALSE)
```

6.3.1 Weighing factor & `lambdaNW`

The weighing factor W_R follows a asymptotic curve when plotted against sample size (Figure 7), and is calculated as

$$W_R = 1 - \exp(-\lambda_{N,W} * N_R) \ . \tag{8}$$

By default $\lambda_{N,W} = -\log(0.5)/100$ (input parameter `lambdaNW`), which corresponds to $W_R < 0.5$ if $N_R < 100$; i.e. if there are fewer than 100 pairs with known age difference, the flat ageprior weighs heavier than the pedigree-inferred ageprior, and the opposite if there are more than 100 pairs. When $\lambda_{N,W}$ is large (say > 0.2), `Flatten` is effectively only applied for relationships with very small sample size (e.g. $W = 0.86$ at $N = 10$),

while when $\lambda_{N,W}$ is small (say < 0.0001) there is effectively no contribution of the pedigree even with very large sample size (e.g. $W = 0.10$ at $N = 1000$).

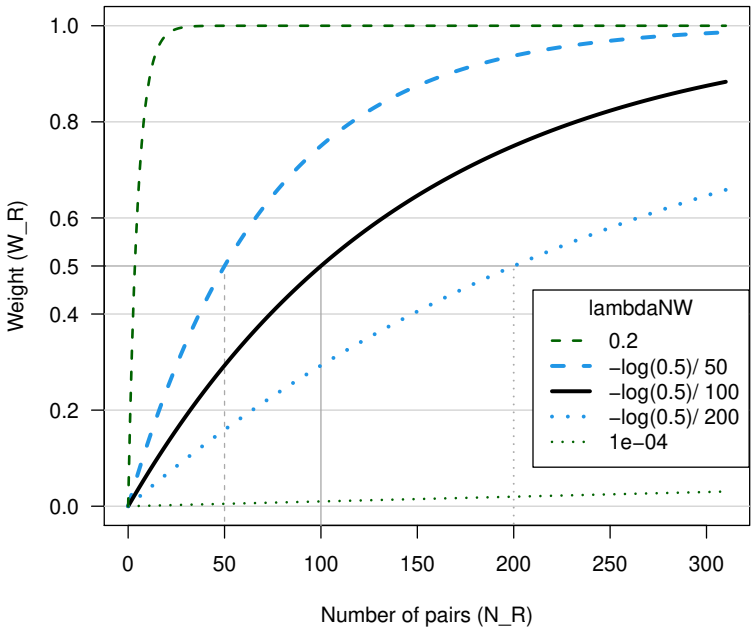


Figure 7: Weights versus number of pairs with relationship R, for weighed average between pedigree-derived ageprior and flat 0/1 ageprior

Since the number of pairs with known age difference will differ between relationship types, so will the weights:

```
AP.griffin$Weights
```

```
##      M      P     FS     MS     PS
## 0.6857 0.6769 0.0341 0.7762 0.6701
```

By chance there are more maternal than paternal sibling pairs in this pedigree, due to one large maternal sibship (see `SummarySeq(Ped_griffin)`), and thus the weight for 'MS' is larger than for 'PS.' We can use the pair counts per relationship to double check the weights:

```
NAK.R   # counts per relationship (column sums of tblA.R)
```

```
##     M     P    FS    MS    PS     X
##   167   163     5   216   160 19900
```

```
lambdaNW <- -log(0.5)/100      # default input value
W.R <-  1 - exp(-lambdaNW * NAK.R[1:5])
round(W.R, 4)
```

```
##      M      P     FS     MS     PS
## 0.6857 0.6769 0.0341 0.7762 0.6701
```

6.3.2 Sample size vs. sampling probability

In theory, the sampling probability per relationship could be estimated, and used to calculate some correction factor for equation (3). In practice, this requires some strong assumptions about the number of assignable parents (i.e. distinction between pedigree founders and non-founders) and the number of sibling pairs (i.e. distribution of sibship sizes). Moreover, this correction factor would depend on the total number of sampled individuals in a non-nonsensical way: removing individuals without assigned parents from the pedigree would 'magically' increase our faith in the ageprior estimated from the remainder of the pedigree.

Therefore, the degree of correction by `Flatten` depends only on the *number* of individual pairs per relationship with known age difference, not on a proportion.

7 Grandparents & Avuncular

7.1 Maths

The age difference between say a grand-offspring and its maternal full uncle, depends on the age difference between the individual and its mother, and between its mother and her full brother. It is assumed that these age differences are independent, so that the ageprior for maternal full-avuncular pairs can be calculated directly from the agepriors for mother-offspring and full sibling pairs.

If individual i is born in year 0, then the probability that its maternal full uncle j is born in year x is proportional to the ageprior $\alpha_{A=x,R=MFA}$. This probability is calculated as the probability that i's mother k is born in year y, summed over all possible y (see Figure 8, with example $y = 3$).

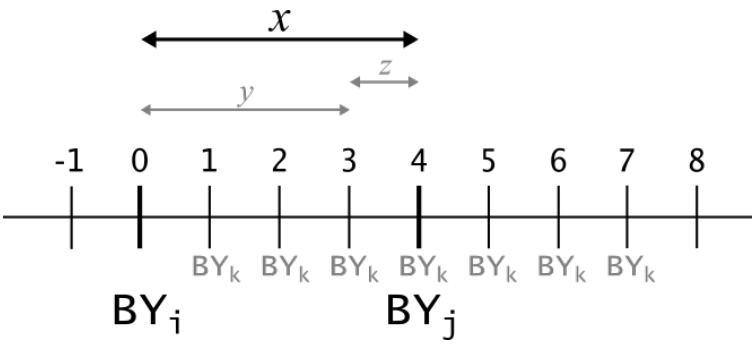


Figure 8: Age difference between individual i and aunt/uncle j, and possible 'birth years' for i's parent k.

The probability that k is born in year y is a function of both the age difference between mother-offspring pair i and k ($A_{i,k} = y$, $y > 0$) and between full siblings k and j ($A_{j,k} = x - y = z$).

The probability that uncle j is born in year x can then be written as:

$$P(A_{i,j} = x|R_{i,j} = MFA) \propto \sum_{y=0}^{Tmax} \sum_{z=-Tmax}^{Tmax} I(y + z = x) \cdot \tag{9}$$

$$P(A_{i,k} = y|R_{i,k} = M)P(A_{k,j} = z|R_{k,j} = FS) \tag{10}$$

where $I(y + z = x)$ is an indicator which equals 1 if y and z sum to x, and equals 0 otherwise.

For avuncular pairs, both z and x may be negative; z if uncle j is a younger sibling of mother k ($BY_j < BY_k$), and x if the age difference between the siblings is larger than between mother and offspring ($BY_j < BY_i$). For grand-offspring – grandparent pairs, both $A_{i,k}$ and $A_{k,j}$ are strictly positive.

7.1.1 Ageprior to Probability to Ageprior

To calculate the avuncular and grand-parental age difference probabilities, first the agepriors for the parental and sibling pairs are scaled so that or each relationship they sum to 1.

After calculation, the grandparent and avuncular age-difference probabilities are scaled to be comparable with the parent-offspring and sibling agepriors. Scaling by the total number of pairs with that age difference (column 'X' in `tblA.R`) is often problematic, because the large age differences possible for grand-parental pairs are often sparsely sampled, or not at all. A very small denominator results in an inflated ageprior, while

a denominator of zero results in an undefined ageprior. Instead, the grandparent and avuncular age-difference probabilities are scaled by the mean of the (column) sums of the mother-offspring and father-offspring agepriors, which are typically quite similar to each other and to the column sums of the sibling agepriors.

7.1.2 change from v1.3 to v2.0

The ageprior distribution for avuncular pairs (aunts/uncles) has changed considerably from sequoia v1.3.3 to v2.0. Earlier, it was assumed that the ageprior for maternal and paternal full- and half- aunts and uncles was approximately similar, and approximately symmetrical around zero. From version 2.0 both these assumptions have been dropped, and there are now 6 avuncular classes (FS, MS, or PS of dam or sire) with agepriors that are not symmetrical around zero.

The agepriors for grandparents and avuncular relationships have always been estimated from parent and sibling age distributions, but this estimation has now been moved from `MakeAgePrior()` in R (up to v.1.3.3) to the Fortran part of `sequoia()`. The extra columns are calculated just *before* parentage assignment or sibship clustering, and returned in the output as `AgePriorExtra`. This is in contrast to the 'regular' ageprior, which is updated just after parentage assignment (see also pipeline in Figure 2).

8 Customise

Customising the ageprior is generally not necessary, as a sample-specific ageprior will be estimated after parentage assignment based on the scaffold-pedigree.

Parentage assignment is rarely improved by a non-default ageprior, unless the number of SNPs is so low or their quality so poor that it is difficult to genetically distinguish between parent-offspring and full sibling pairs.

The ageprior used for full pedigree reconstruction may require tweaking if for example the assigned parents reflect only a portion of the biologically possible parental ages, or if some assigned parents are wrong and have a biologically impossible age.

8.1 Via `args.AP`

During pedigree reconstruction, you can pass arguments to `MakeAgePrior()` via `sequoia()`'s argument `args.AP`. Use this to explicitly specify `Discrete`, `Smooth` or `Flatten`, or to increase `MaxAgeParent`.

8.2 Different input pedigree

If you have a reasonable quality pedigree that contains many more individuals than have been SNP-genotyped, for example based on field observations and/or microsatellite data, an ageprior estimated from that old pedigree may be more informative than the one estimated from a limited number of SNP-genotyped parent-offspring pairs. A pedigree cannot be passed via `args.AP`, but this can be done as follows instead:

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

The ageprior in `SeqList` will be used for both parentage assignment and full pedigree reconstruction.

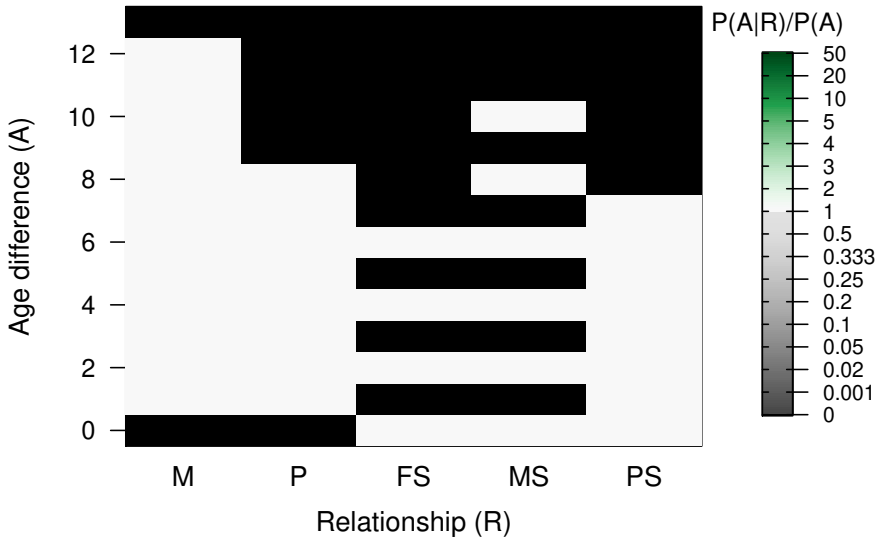
8.3 Other tweaks

The easiest way to create a fully custom ageprior is to generate an approximate ageprior with `MakeAgePrior`, and tweak it by hand or using your own function. For example, imagine a species where females breed only every other year, but may have twins:

```
AP <- MakeAgePrior(MaxAgeParent = c(12, 8), Plot=FALSE)
```

```
## Ageprior: Flat 0/1, overlapping generations, MaxAgeParent = 12,8
```

```
AP[as.character(seq(1,12,by=2)), c("MS", "FS")] <- 0
PlotAgePrior(AP)
```



8.3.1 Age at first reproduction

Or, if you wish to specify the minimum age at first reproduction, to e.g. age 3 for females, and age 4 for males:

```
AP <- MakeAgePrior(MaxAgeParent = c(12, 8), Plot=FALSE)
```

```
## Ageprior: Flat 0/1, overlapping generations, MaxAgeParent = 12,8
```

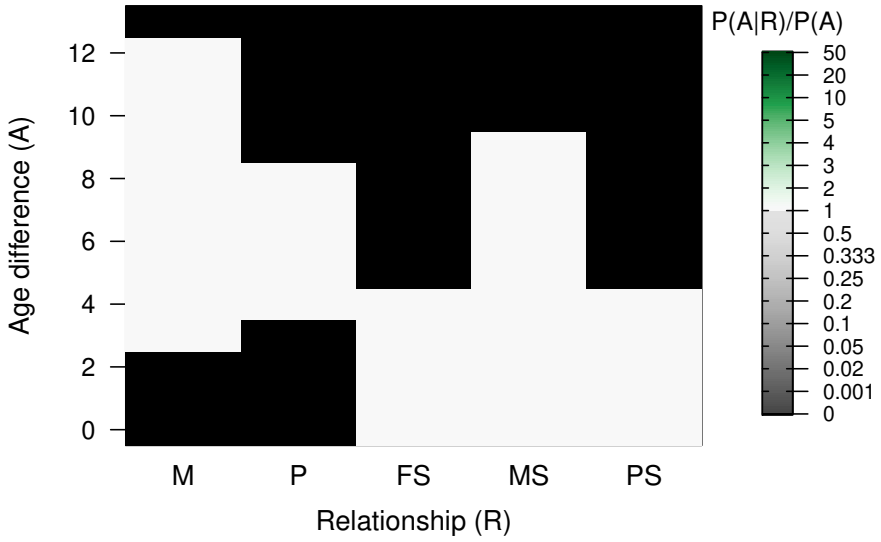
```
AP[c("1","2"), "M"] <- 0
AP[c(1:3) +1, "P"] <- 0   # 1st row = age difference of 0.

# check if valid ageprior:
chk <- sequoia(SimGeno_example, LifeHistData=LH_HSg5,
               SeqList = list(AgePriors = AP),
               Module="pre", quiet=TRUE, Plot=FALSE)
```

```
## Error in CheckAP(SeqList[["AgePriors"]]): Sibling column  MS  not consistent with parent column
```

```
# Fix: a female may breed from age 3 to age 12
# --> maternal sibs have max age dif. of 9 years
AP[c(10:12) +1, "MS"] <- 0     # +1 because 1st row is '0'
# a male may breed from age 4 up to age 8  --> max age dif of PS is 4 years
AP[c(5:12) +1, c("PS", "FS")] <- 0
PlotAgePrior(AP)
```

8.4 From age-difference distribution

If you know approximately the distributions of parental ages, and the age differences between siblings, you might be able to create a table similar to `tblA.R` and follow the subsequent steps described here to generate



an ageprior matrix. The most challenging may be to generate the age-difference distribution of 'all individuals' (column X), used for standardisation.

8.5 Specifications

The ageprior must abide by the following rules:

- a matrix or dataframe with values between 0 and 1000 inclusive. In the `MakeAgePrior()` output, the smallest possible non-zero value is 1/1000, to avoid excessive weight of the ageprior relative to the genetic likelihood;
- columns named M, P, FS, MS, and PS, any other columns are ignored;
- each relationship must be possible (> 0) for at least one age difference;
- rownames are ignored and rows are presumed to be for age difference 0, 1, 2, ..., *except* when rownames include negative numbers (as for `AgePriorExtra`), then negative-numbered rows are deleted and the remainder are presumed to be ordered and for age difference 0, 1, 2, ...;
- any age differences not included (i.e. $>$ `nrows` -1) are presumed impossible for all types of parents and siblings (0 in all 5 columns);
- the maximum age difference between siblings must be consistent with the maximum breeding tenure of a parent (minimum – maximum age of reproduction).

To check whether your ageprior matrix satisfies these criteria, use it in any function that takes `AgePriors` (or `SeqList`) as input, which will call the internal function `CheckAP()`.

References

Huisman, Jisca. 2017. "Pedigree Reconstruction from SNP Data: Parentage Assignment, Sibship Clustering and Beyond." *Molecular Ecology Resources* 17 (5): 1009–24.