intro_hiphop

Authors

Martijn van de Pol, Lyanne Brouwer and Andrew Cockburn - 11 August 2020

What is R package hiphop for?

R package hiphop is a method for parentage assignment using bi-allelic genetic markers like SNPs (Single Nucleotide Polymorphism). It has widespread application (paternity and maternity assignment in a variety of mating systems) and outperforms conventional methods where closely related individuals occur in the pool of possible parents. The method is based on simple exclusion logic by comparing the genotypes of offspring with those of potentials parents using the HIPHOP and HOT test; the rationale behind this paternity assignment method is described in detail in Cockburn et al. (for full reference use citation(“hiphop”)). Here we explain how it can be used with worked examples based on the data set that are used in the above paper and are included in the hiphop package.

The basic idea

The basic idea behind the HIPHOP parentage assignment is that we can compare the genotypes of any combination of offspring with potentials dams and sires (i.e. genetic mother or fathers) by comparing the exclusion scores of these individuals at bi-allelic markers. It elaborates on a prior exclusion method based on the Homozygous Opposite Test (HOT; Huisman 2017; https://doi.org/10.1111/1755-0998.12665). HOT compares the genotype of an offspring with a potential parent: a mismatch is scored when both the offspring and parent are homozygous, but for different alleles. Cockburn et al. suggested an additional -and as we will later see more informative- exclusion criterion called HIPHOP (Homozygous Identical Parents, Heterozygous Offspring are Precluded). This test compares the genotype of an offspring with both potential parents: a mismatch is scored when the offspring is heterozygous and both parents are homozygous for the same allele. Furthermore, although the HOT test was originally designed to compare the genotypes of offspring with one parent (a dyad), it can easily be extended to also compare offspring to both parents (a triad, similar to the HIPHOP test that also compares triads): a mismatch is scored when the offspring is homozygous and at least one of the parents is homozygous for the opposite allele (i.e. the HOT.parents test). The HOT and HIPHOP scores of triads can be combined to provide information about the total of mismatches and is proposed as an improved exclusion criterion for parentage assignment, particularly when distinguishing between closely related potential parents.

Below mismatch table shows all possible combinations of bi-allelic genotypes of offspring-dam-sire combinations for a single locus and which combinations of genotypes are indicative of a mismatch (the R code also illustrates the basic calculations). The total number of mismatches (also called the ‘HOTHIPHOP.parents’ mismatch score) is the sum of the HOT scores of the offspring with the parents (HOT.parents) and the HIPHOP score with the potential parents (offspring-dam-sire triad). Mismatches may occur because (a) the potential dam and/or sire is not the true genetic parent, (b) genotyping errors, and (c) mutations. We are aiming to exclude potential parents by determining if they have more mismatches than can be expected due to genotyping errors and mutation, and thereby identify (i) the true genetic parents and (ii) detect situations where one (or both) of the true parents is not sampled, in which case we (iii) want to identify which parent can still be correctly assigned.

```r
genotyp<-c("AA", "Aa", "aa")
gentable.names<-c("offspring","dam","sire","hot.dam","hot.sire","hot.parents",
         "hiphop","hothiphop.parents")
gentable<-as.data.frame(array(NA,dim = c(length(genotyp)^3,length(gentable.names)),
         dimnames=list(NULL,gentable.names)))
gentable$offspring<-rep(genotyp, each=length(genotyp)*length(genotyp))
gentable$dam<-rep(genotyp, length(genotyp)*length(genotyp))
gentable$sire<-rep(genotyp, length(genotyp), each=length(genotyp))
# the HOT test for the offspring-dam dyad
gentable$hot.dam<-ifelse( (gentable$offspring=="AA" & gentable$dam=="aa") | (gentable$offspring=="aa" &
         gentable$dam=="AA"), 1,0)
# the HOT test for the offspring-sire dyad
gentable$hot.sire<-ifelse( (gentable$offspring=="AA" & gentable$sire=="aa") | (gentable$offspring=="aa"
         & gentable$sire=="AA"), 1,0)
# the HOT test for the offspring-dam-sire triad
gentable$hot.parents<-ifelse(((gentable$offspring=="AA" & (gentable$sire=="aa" | gentable$dam=="aa")) |
         (gentable$offspring=="aa" & (gentable$sire=="AA" | gentable$dam=="AA"))),1,0)
# the hiphop for the dam-sire combination
gentable$hiphop<-ifelse( (gentable$offspring=="Aa" & gentable$sire=="aa" & gentable$dam=="aa") |
         (gentable$offspring=="Aa" & gentable$sire=="AA" &  gentable$dam=="AA"), 1,0)
gentable$hothiphop.parents<-gentable$hot.parents+gentable$hiphop
print(gentable)
```

```
#>    offspring dam sire hot.dam hot.sire hot.parents hiphop hothiphop.parents
#> 1         AA  AA   AA       0        0           0      0                 0
#> 2         AA  Aa   AA       0        0           0      0                 0
#> 3         AA  aa   AA       1        0           1      0                 1
#> 4         AA  AA   Aa       0        0           0      0                 0
#> 5         AA  Aa   Aa       0        0           0      0                 0
#> 6         AA  aa   Aa       1        0           1      0                 1
#> 7         AA  AA   aa       0        1           1      0                 1
#> 8         AA  Aa   aa       0        1           1      0                 1
#> 9         AA  aa   aa       1        1           1      0                 1
#> 10        Aa  AA   AA       0        0           0      1                 1
#> 11        Aa  Aa   AA       0        0           0      0                 0
#> 12        Aa  aa   AA       0        0           0      0                 0
#> 13        Aa  AA   Aa       0        0           0      0                 0
#> 14        Aa  Aa   Aa       0        0           0      0                 0
#> 15        Aa  aa   Aa       0        0           0      0                 0
#> 16        Aa  AA   aa       0        0           0      0                 0
#> 17        Aa  Aa   aa       0        0           0      0                 0
#> 18        Aa  aa   aa       0        0           0      1                 1
#> 19        aa  AA   AA       1        1           1      0                 1
#> 20        aa  Aa   AA       0        1           1      0                 1
#> 21        aa  aa   AA       0        1           1      0                 1
#> 22        aa  AA   Aa       1        0           1      0                 1
#> 23        aa  Aa   Aa       0        0           0      0                 0
#> 24        aa  aa   Aa       0        0           0      0                 0
#> 25        aa  AA   aa       1        0           1      0                 1
#> 26        aa  Aa   aa       0        0           0      0                 0
#> 27        aa  aa   aa       0        0           0      0                 0
```

Parentage analysis: single parent versus parent pair assignment.

Parentage analysis typically comes in two forms:

1. Parentage assignment when one genetic parent is known: single parent analysis

In the conventional case of parentage analysis there is usually contextual information that provides evidence about the identity of one of the genetic parents. The most common case will be where a female has been observed to be involved in reproductive tasks (built a nest, laid or incubated the eggs, lactated young). For the male parent we typically cannot make this assumption, because contextual evidence for paternity is usually less obvious and promiscuity means that paternity is often uncertain and needs to assessed (though in some species with sex role reversal, it may be possible to condition on the social father and focus on maternity assignment).

2. Parentage assignment when no genetic parents are known: parent pair analysis

In other systems we have no or little contextual information about either of the social parents, in which case we can make no a priori assumptions about the genetic parents. In such cases the aim is to assign both the genetic mother (dam) and genetic father (sire) and perform a parent pair analysis. Such cases occur when: (a) Communal spawning: some animals with external fertilisation that breed in aggregations so that it is impossible to know which female laid eggs (e.g. frogs in leks, some fish) and which male fertilized the eggs, as there are no permanent social groups. (b) Polygynandry: multiple females and males breed and deposit their young in a common burrrow or nest (e.g. acorn woodpeckers, Eurasian badgers), which may be difficult to observe so individual offspring cannot be associated with a single parent. There may also be extra-group mating (as has been argued for chimpanzees and Australian magpies).

How does it work?

R package hiphop includes three functions and an example dataset consisting of two input dataframes based on five cohorts (2014-2018) of superb fairy wrens living in Canberra, Australia.

We first load the library.

```
library(hiphop)
```

Example dataset

The superb fairy wren case reflects the first case described above, as there is a social pair, in which previous evidence has shown that the female that incubates the eggs (the social mother) is always the dam. However, the social male is generally not the sire, as the rate of extra-pair paternity is among the highest recorded in bird species. Furthermore, besides the male that is pair bonded to the social mother (the social father), each group contains supernumerary males that sometimes also gain paternity (there is within- and extra-group paternity). Finally, superb fairy wrens have very limited dispersal, almost all males live their entire life on their natal or a neighbouring territory, and as a consequence paternity assignment thus faces the challenge of distinguishing between closely related potential sires (e.g. first order relatives).

In the next sections we will first use the example dataset to illustrate the workings of the functions in package hiphop. Next we will use superb fairy wrens to illustrate how to conduct single parent assignment of the sires. Finally, we will also use the same dataset to illustrate how to perform parent-pair analyses for the second case (by ignoring the fact that in superb fairy wrens we know the genetic mother a priori).

individuals dataframe

The dataset consist of two dataframes/files. The first input dataframe -called ‘individuals’- consists of a list of 1153 offspring that we would like to assign parentage to and 1373 records of potential parents (252 adult males and 145 adult females). The file has 5 variables:

```
individuals[23:33,]
#>        brood  individual         type social.parent year
#> 23 awRM_6299 899972-rnWR   adult male             1 2014
#> 24 awRM_6299 982804-awRM adult female             1 2014
#> 25 awRM_6299 A58115-NAgn    offspring             0 2014
#> 26 awRM_6299 A58116-GMgn    offspring             0 2014
#> 27 awRM_6299 A58117-OYgn    offspring             0 2014
#> 28 awRM_6300 899972-rnWR   adult male             1 2014
#> 29 awRM_6300 982045-BwgW   adult male             0 2014
#> 30 awRM_6300 982804-awRM adult female             1 2014
#> 31 awRM_6300 A58166-MObw    offspring             0 2014
#> 32 awRM_6300 A58167-YRbw    offspring             0 2014
#> 33 awRM_6300 A58168-AGbw    offspring             0 2014
```

The first variable denotes the brood to which the individual belongs, the second variable is the identity of the individual, the third variable (type) denotes whether it is an offspring, adult male (i.e. a potential sire) or adult female (potential dam). The fifth column (social.parent) describes whether the individual is a social parent at the brood it is associated with (for females 1, if she incubated or build the nest; for males 1, if he was the dominant male pair bonded with the social mother) or not (0; supernumerary males). The final variable (year) is to separate cohorts to be analyzed. Note that parentage analyses is done typically done one cohort at a time, as the set of potential dams and sires will vary across years (as some individuals were not born yet or were already dead in some years).

Thus individuals may appear in multiple years and adults may also appear multiple times within a year if they were involved in multiple broods. For example, the above 10 records show two broods of the same breeding pair (982804-awRM & 899972-rnWR). Both broods produced three offspring, but the second brood had a supernumerary male (982045-BwgW) helping the parents.

It is important to realise that there will usually be potential parents living both within and outside the study area that will not be considered if just the adults associated with a brood are included in this file. Cockburn et al. showed that some young are sired by males living outside the study area, and in other species floaters might be important. A universal problem will be parents living within the study area that do not raise young to the point where DNA can be sampled. Ensuring that such adults are considered as parents can be achieved by giving them a dummy brood ID. As can be seen below, in our dataset we have added such males using brood values such as other353, etc.

```
tail(individuals)
#>         brood  individual       type social.parent year
#> 2521 other353 899575-wbMM adult male             0 2015
#> 2522 other354 899575-wbMM adult male             0 2017
#> 2523 other355 899575-wbMM adult male             0 2018
#> 2524 other356 A58484-sOnr adult male             0 2015
#> 2525 other357 A58484-sOnr adult male             0 2017
#> 2526 other358 A58484-sOnr adult male             0 2018
```

In some mating systems it will not be possible to separate the young into broods and there is no parental care. In such cases all young and potential parents can be given the same brood ID (with brood potentially meaning day of sampling, frog pond, fish tank). We note that the brood ID is not used for calculating the exclusion scores, it is only used to qualify any potential parent as either within- or extra-group (i.e. extra-brood) parent.

genotypes dataframe

The second input dataframe/file -called ‘genotypes’- consists of a list of all genotyped individuals (offspring and potential parents) and their genotypes scored using bi-allelic markers. In the example dataset it comprises 1407 individuals scored at 1376 loci.

```
genotypes[1:5,1:13]
#>             L545 L1022 L232 L696 L829 L1218 L7 L160 L857 L1475 L27 L495 L1100
#> 898377-onWA    0     2    0    0    2     2  2    2    2     2   0    2     2
#> 898797-YweY    0     2    0    2    2     0  0    2    2     0   2    2     0
#> 899199-OgoY    0     2    0    0    0     2  2    0    2     0   0    2     0
#> 899484-NybR    0     0    0    0    0     2  0    1    0     0   0    2     0
#> 899575-wbMM    0     2    0    0    2     2  2    1    2     0   0    1     2
```

The column names are alphanumerics used to identify each SNP (Single Nucleotide Polymorphism) and the row names identify each individual. Values are specified as follows:

- 0 - homozygous for one allele (usually the common allele, but this is not necessary)
- 1 - homozygous for the alternative allele
- 2 - heterozygous at this locus
- NA - a null result, where genotyping failed.

If an individual was not sampled and/or genotyped it should not be included in the genotypes dataset.

The functions

R package hiphop only contains 3 functions: inspect, hothiphop and topmatch.

1. The inspect function is used for data inspection to detect any anomalies in genotypic data.
2. The hothiphop function calculates all the mismatch scores according to the HOT and HIPHOP test for all possible offspring-potential dam-potential sire combinations.
3. The topmatch function summarizes the best matches for each offspring according to the output of the hothiphop function. The default lists the top three offspring-potential dam-potential sire combinations that have the lowest mismatch score, and if social parents are specified, provides results for the social parents as well (if they are not ranked in the top three). Options defined below allow the sorting criterion to be varied, the number of triads listed to be increased and decreased, and the analysis conditioned on knowing one parent (for single parent assignment).

A worked example to illustrate how the functions work

The hothiphop function extracts the list of potential dams and potential sires for a specific cohort of offspring from the individuals dataset. For that cohort it will consider all adult females listed that year as potential dams and all adult males listed that year as potential sires. For reasons given above, we analyse one cohort/year at a time.

select a cohort

We thus first subset the individuals dataset to one cohort, here the 2018 cohort of superb fairy wrens.

```
ind2018<-subset(individuals, individuals$year==2018)
```

inspect the data

Next we inspect the genotypes’ summaries and identify any individual that has not been genotyped:

```
inspection<-inspect(ind=ind2018, gen=genotypes)
head(inspection)
#>      brood  individual         type social.parent year sampled homozygote.0
#> 1 other287 A58192-NBwr   adult male             0 2018       1    0.6337209
#> 2 other288 A58331-GNgn   adult male             0 2018       1    0.6446221
#> 3 other289 A58815-pOny   adult male             0 2018       1    0.6148256
#> 4 other290 A58816-pWob   adult male             0 2018       1    0.6133721
#> 5 other291 A58325-NNbw   adult male             0 2018       1    0.6286337
#> 6 other292 A58258-RNgn adult female             1 2018       1    0.6395349
#>   homozygote.1 heterozygote.2     missing number.loci
#> 1    0.1235465      0.2369186 0.005813953        1368
#> 2    0.1220930      0.2311047 0.002180233        1373
#> 3    0.1155523      0.2667151 0.002906977        1372
#> 4    0.1170058      0.2652616 0.004360465        1370
#> 5    0.1199128      0.2441860 0.007267442        1366
#> 6    0.1235465      0.2296512 0.007267442        1366
inspection[which(inspection$sampled==0),]
#>        brood  individual       type social.parent year sampled homozygote.0
#> 41 fBab_7322 SOC018-fBab adult male             1 2018       0            0
#>    homozygote.1 heterozygote.2 missing number.loci
#> 41            0              0       1           0
```

The summary lists, in addition to some details of the individual, the proportion of loci for which the individual is homozygous, heterozygous or has missing data. We see that most individuals have very few missing loci data (mostly <1%), but that there is one social father (SOC018-fBab) that has not been sampled.

calculate the mismatch scores

Next we run the mismatch analysis for all offspring-potential dam-potential sire combinations for the 2018 cohort using the hothiphop function. This function calculates the proportion of genetic mismatches according to the HIPHOP and HOT test.

In the 2018 cohort there are 149 offspring, 42 potential dams and 73 potential sires (each superb fairy wren brood has a social pair and 0-5 supernumerary adult males).

```
print(c(length(unique(ind2018$individual[which(ind2018$type=="offspring")])),
        length(unique(ind2018$individual[which(ind2018$type=="adult female")])),
        length(unique(ind2018$individual[which(ind2018$type=="adult male")]))))
#> [1] 149  42  73
```

This means that there are about half a million possible combinations of offspring-dam-sire (149x42x73), so the next code may take several minutes to run:

```
combinations<-hothiphop(ind=ind2018, gen=genotypes)
```

```
head(combinations)
#>   year    brood   offspring potential.dam potential.sire hothiphop.parents
#> 1 2018 BBW_7279 A59401-iOar   A58258-RNgn    A58192-NBwr               275
#> 2 2018 BBW_7279 A59401-iOar  A59012-zOogo    A58192-NBwr               242
#> 3 2018 BBW_7279 A59401-iOar    A58110-BBW    A58192-NBwr               197
#> 4 2018 BBW_7279 A59401-iOar   A58954-fGny    A58192-NBwr               319
#> 5 2018 BBW_7279 A59401-iOar   A58908-fBab    A58192-NBwr               287
#> 6 2018 BBW_7279 A59401-iOar   A59232-cWar    A58192-NBwr               311
#>   hiphop hot.parents hot.dam hot.sire hothiphop.dam hothiphop.sire
#> 1    122         153      72       99           194            221
#> 2    109         133      53       99           162            208
#> 3     90         107      17       99           107            189
#> 4    143         176     118       99           261            242
#> 5    135         152      87       99           222            234
#> 6    143         168      96       99           239            242
#>   loci.dyad.dam loci.dyad.sire loci.triad offspring.heterozygosity
#> 1          1356           1358       1349                0.2401171
#> 2          1357           1358       1349                0.2401171
#> 3          1355           1358       1347                0.2401171
#> 4          1358           1358       1350                0.2401171
#> 5          1362           1358       1354                0.2401171
#> 6          1357           1358       1349                0.2401171
#>   social.mother.sampled social.father.sampled is.dam.social is.sire.social
#> 1                     1                     1             0              0
#> 2                     1                     1             0              0
#> 3                     1                     1             1              0
#> 4                     1                     1             0              0
#> 5                     1                     1             0              0
#> 6                     1                     1             0              0
#>   is.dam.within.group is.sire.within.group social.mother social.father
#> 1                   0                    0    A58110-BBW   A58794-uWny
#> 2                   0                    0    A58110-BBW   A58794-uWny
#> 3                   1                    0    A58110-BBW   A58794-uWny
#> 4                   0                    0    A58110-BBW   A58794-uWny
#> 5                   0                    0    A58110-BBW   A58794-uWny
#> 6                   0                    0    A58110-BBW   A58794-uWny
```

This output is large and contains all the required information to assign parentage. Basically it lists all possible offspring-potential.dam-potential.sire combinations and their mismatch scores (unsorted), the number of loci this was based on, and some relevant information about the social parents and potential dam and sires. For a detailed explanation see the help file of the function ('?hothiphop()')

summarize the best matching parents

However, due to the large number of combinations it can be difficult to find the salient information, so the results can be summarized into a more insightful format by extracting the top matches for each offspring using the 'topmatch' function. By default for each offspring the 'topmatch' returns the 3 top matches for male/female combinations according to the chosen ranking criteria, and additionally if they are not yet included, the results for the triad comprising the social parents.

```
best.hothiphop<-topmatch(x=combinations, ranking="hothiphop.parents")
best.hothiphop[1:8,]
#>   year    brood   offspring rank          dam         sire           dam.type
#> 1 2018 BBW_7279 A59401-iOar    1   A58110-BBW  A59058-zMyr      social parent
#> 2 2018 BBW_7279 A59401-iOar    2   A58110-BBW   A59372-sOb      social parent
#> 3 2018 BBW_7279 A59401-iOar    3 A59012-zOogo  A59058-zMyr extra-group parent
#> 4 2018 BBW_7279 A59401-iOar   55   A58110-BBW  A58794-uWny      social parent
#> 5 2018 BBW_7279 A59402-iMow    1   A58110-BBW  A59058-zMyr      social parent
#> 6 2018 BBW_7279 A59402-iMow    2   A58110-BBW A59201-cNbwb      social parent
#> 7 2018 BBW_7279 A59402-iMow    3   A58110-BBW   A59372-sOb      social parent
#> 8 2018 BBW_7279 A59402-iMow  120   A58110-BBW  A58794-uWny      social parent
#>                  sire.type hothiphop.parents hiphop hot.parents hot.dam
#> 1       extra-group parent                37      0          37      17
#> 2       extra-group parent               109     43          66      17
#> 3       extra-group parent               117     47          70      53
#> 4            social parent               191    116          75      17
#> 5       extra-group parent                33      1          32      18
#> 6 within-group subordinate               104     62          42      18
#> 7       extra-group parent               112     58          54      18
#> 8            social parent               201    133          68      18
#>   hot.sire hothiphop.dam hothiphop.sire below.threshold threshold loci.dyad.dam
#> 1       20            17             20               1     99999          1355
#> 2       53            60             96               1     99999          1355
#> 3       20           100             67               1     99999          1357
#> 4       70           133            186               1     99999          1355
#> 5       14            19             15               1     99999          1356
#> 6       28            80             90               1     99999          1356
#> 7       42            76            100               1     99999          1356
#> 8       63           151            196               1     99999          1356
#>   loci.dyad.sire loci.triad offspring.heterozygosity social.mother.sampled
#> 1           1355       1344                0.2401171                     1
#> 2           1358       1347                0.2401171                     1
#> 3           1355       1346                0.2401171                     1
#> 4           1360       1349                0.2401171                     1
#> 5           1356       1344                0.2660819                     1
#> 6           1368       1356                0.2660819                     1
#> 7           1361       1349                0.2660819                     1
#> 8           1362       1350                0.2660819                     1
#>   social.father.sampled social.mother social.father condition           ranking
#> 1                     1    A58110-BBW   A58794-uWny      none hothiphop.parents
#> 2                     1    A58110-BBW   A58794-uWny      none hothiphop.parents
#> 3                     1    A58110-BBW   A58794-uWny      none hothiphop.parents
#> 4                     1    A58110-BBW   A58794-uWny      none hothiphop.parents
#> 5                     1    A58110-BBW   A58794-uWny      none hothiphop.parents
#> 6                     1    A58110-BBW   A58794-uWny      none hothiphop.parents
#> 7                     1    A58110-BBW   A58794-uWny      none hothiphop.parents
#> 8                     1    A58110-BBW   A58794-uWny      none hothiphop.parents
#>   ranking2
#> 1     none
#> 2     none
#> 3     none
#> 4     none
#> 5     none
#> 6     none
#> 7     none
#> 8     none
```

For offspring A59401-iOar (the first four lines of output) we can see that the triad A58110-BBW (dam) & A59058-zMyr (sire) has by far the fewest HOTHIPHOP.parents mismatches (37; HOTHIPHOP.parents = HOT score of the parents/triad + HIPHOP score of triad), much less than the second ranked dyad (109 mismatches; the same dam with another sire). The best ranked dam A58110-BBW is the social mother, while the best ranked sire A59058-zMyr is an extra-group parent (as it is not associated with the same brood as the offspring in the individuals dataset). In line 4 we can see that the social father also has many more mismatches (191), and is unlikely to be the sire. In rows 4-8 we see another offspring from the same brood, which in this case is sired by the same extra-group male A59058-zMyr (33 mismatches).

options to summarize the best matches in different ways

It is possible to set a threshold value that determines whether the dam, sire or triad should be accepted as parents (see later for determining a threshold).

```
best<-topmatch(x=combinations, ranking="hothiphop.parents", thres=62)
best[9:12,c(1:9, 16,17)]
#>    year     brood   offspring rank         dam        sire      dam.type
#> 9  2018 fBab_7322 A59439-iMwr    1 A58908-fBab A58909-fMrn social parent
#> 10 2018 fBab_7322 A59439-iMwr    2 A58908-fBab A59065-zOyb social parent
#> 11 2018 fBab_7322 A59439-iMwr    3 A58908-fBab A58815-pOny social parent
#> 12 2018 fBab_7322 A59439-iMwr   NA A58908-fBab        <NA> social parent
#>             sire.type hothiphop.parents below.threshold threshold
#> 9  extra-group parent               163               0        62
#> 10 extra-group parent               164               0        62
#> 11 extra-group parent               165               0        62
#> 12      social parent                NA               0        62
```

In our dataset, we found empirically that for this species and dataset (level of genotyping error) the true parents never had more than 62 HOTHIPHOP.parents mismatches (Cockburn et al.), and by setting the threshold to this value we can see that in the column 'below.threshold' all triads above (below) this value are labelled as 0 (1). If not specified, the default threshold value is a large value (99999) meaning that all triads are below the threshold.

If the threshold value is set to a value between 0 and 1 then the number of mismatches are recalculated to the proportion of mismatches (number of mismatches / number of loci that are not NA), which is a more accurate score if individuals differ strongly in the number of loci with missing data.

```
best<-topmatch(x=combinations, ranking="hothiphop.parents", thres=0.045)
best[9:12,c(1:9, 16,17)]
#>    year     brood   offspring rank         dam        sire      dam.type
#> 9  2018 fBab_7322 A59439-iMwr    1 A58908-fBab A58909-fMrn social parent
#> 10 2018 fBab_7322 A59439-iMwr    2 A58908-fBab A59065-zOyb social parent
#> 11 2018 fBab_7322 A59439-iMwr    3 A58908-fBab A58815-pOny social parent
#> 12 2018 fBab_7322 A59439-iMwr   NA A58908-fBab        <NA> social parent
#>             sire.type hothiphop.parents below.threshold threshold
#> 9  extra-group parent         0.1202065               0     0.045
#> 10 extra-group parent         0.1212121               0     0.045
#> 11 extra-group parent         0.1216814               0     0.045
#> 12      social parent                NA               0     0.045
```

Finally, if the threshold value is set to a value between 0 and -1 then the number of mismatches are recalculated to the proportion of mismatches, but in contrast to above example the mismatch scores are expressed as the number of mismatched divided by the number of loci at which the offspring is heterozygote (for the HIPHOP test) or divided by number of loci at which the offspring is homozygote (for the HOT test). Thus, by setting 'thres=-0.99' we aim to correct for the fact that heterozygous offspring will accumulate more HIPHOP and less HOT mismatches, all else being equal. For calculating whether triads are below.threshold, they will be compared to abs(thres).

```
best<-topmatch(x=combinations, ranking="hothiphop.parents", thres=-0.99)
best[9:12,c(1:9, 16,17)]
#>    year     brood   offspring rank         dam        sire      dam.type
#> 9  2018 fBab_7322 A59439-iMwr    1 A58908-fBab A59065-zOyb social parent
#> 10 2018 fBab_7322 A59439-iMwr    2 A58908-fBab A58816-pWob social parent
#> 11 2018 fBab_7322 A59439-iMwr    3 A58908-fBab A58815-pOny social parent
#> 12 2018 fBab_7322 A59439-iMwr   NA A58908-fBab        <NA> social parent
#>             sire.type hothiphop.parents below.threshold threshold
#> 9  extra-group parent         0.2987475               1   (-)0.99
#> 10 extra-group parent         0.3118211               1   (-)0.99
#> 11 extra-group parent         0.3159056               1   (-)0.99
#> 12      social parent                NA               1   (-)0.99
```

The ‘topmatch’ function can rank matches via several criteria specified by the parameter ‘ranking’. Options are: “hothiphop.parents” , “hiphop” ‘hot.dam’, ‘hot.sire’ as well as ‘hothiphop.dam’ and ‘hothiphop.sire’ (see ?topmatch() for explanation of how these criteria differ). For example:

```
best.hiphop<-topmatch(x=combinations, ranking="hiphop")
best.hiphop[1:4,1:13]
#>   year    brood   offspring rank          dam        sire           dam.type
#> 1 2018 BBW_7279 A59401-iOar    1   A58110-BBW A59058-zMyr      social parent
#> 2 2018 BBW_7279 A59401-iOar    2   A58110-BBW  A59372-sOb      social parent
#> 3 2018 BBW_7279 A59401-iOar    3 A59012-zOogo A59058-zMyr extra-group parent
#> 4 2018 BBW_7279 A59401-iOar  449   A58110-BBW A58794-uWny      social parent
#>            sire.type hothiphop.parents hiphop hot.parents hot.dam hot.sire
#> 1 extra-group parent                37      0          37      17       20
#> 2 extra-group parent               109     43          66      17       53
#> 3 extra-group parent               117     47          70      53       20
#> 4      social parent               191    116          75      17       70
```

At first glance, the HOTHIPHOP.parents scores may appear most informative as it sums all possible HIPHOP and HOT mismatches of the triad. However, the HOT test is disproportionately affected by genotyping error, and thus potentially HIPHOP can give cleaner comparisons in some cases than HOTHIPHOP.parents. As a result of above complexity (as well as other dependencies of HOT and HIPHOP scores on the rate of hetero- and homozygosity of the offspring and parents), the choice and best definition of exclusion criterion is not a trivial matter. Therefore, we refer to Cockburn et al. for a more in depth discussion of the pros and cons of different scoring criteria and how they can differ in their conclusions in parentage assignment.

We note that when ranking on the HOT test scores of dyads (e.g. on HOT.dam as shown below), that the top ranked triads will always contain the same dams (but with different sires), as the HOT.dam score -in contrast to the HIPHOP and HOT.parents score- is not calculated on a triad, but on an offspring-parent dyad and thus does not depend on the genotype of the other parent.

```
best.hot.dam<-topmatch(x=combinations, ranking="hot.dam")
best.hot.dam[1:3,1:13]
#>   year    brood   offspring rank        dam        sire      dam.type
#> 1 2018 BBW_7279 A59401-iOar    1 A58110-BBW A58794-uWny social parent
#> 2 2018 BBW_7279 A59401-iOar    2 A58110-BBW A58192-NBwr social parent
#> 3 2018 BBW_7279 A59401-iOar    3 A58110-BBW A58331-GNgn social parent
#>            sire.type hothiphop.parents hiphop hot.parents hot.dam hot.sire
#> 1      social parent               191    116          75      17       70
#> 2 extra-group parent               197     90         107      17       99
#> 3 extra-group parent               212     89         123      17      115
```

To compare the HOT scores of dams it is necessary to specify that you are only interested in one record per dam, which can be achieved by setting the unique argument:

```
best.hot.dam<-topmatch(x=combinations, ranking=c("hot.dam","hiphop"), unique="dam")
best.hot.dam[1:4,1:13]
#>   year    brood   offspring rank          dam        sire           dam.type
#> 1 2018 BBW_7279 A59401-iOar    1   A58110-BBW A59058-zMyr      social parent
#> 2 2018 BBW_7279 A59401-iOar    1   A58110-BBW A58794-uWny      social parent
#> 3 2018 BBW_7279 A59401-iOar    2 A59012-zOogo A59058-zMyr extra-group parent
#> 4 2018 BBW_7279 A59401-iOar    3  A58258-RNgn A59058-zMyr extra-group parent
#>            sire.type hothiphop.parents hiphop hot.parents hot.dam hot.sire
#> 1 extra-group parent                37      0          37      17       20
#> 2      social parent               191    116          75      17       70
#> 3 extra-group parent               117     47          70      53       20
#> 4 extra-group parent               144     59          85      72       20
```

Now we get the top 3 unique dams ranked according to hot.dam score (as well as the social mother, if not in the top 3), and because we have also included a second ranking argument “hiphop” the dams are shown with the sires with whom they have the lowest hiphop score.

Furthermore, we can use the ‘condition’ parameter to also condition the parentage analysis on prior contextual knowledge of the parentage of one parent. In most species prior knowledge will relate to the social mother, as the fact that she is lactating, laid or incubated the eggs may provide such information (assuming e.g. egg dumping does not occur).

```
best<-topmatch(x=combinations, ranking="hothiphop.parents", condition="mother")
best[1:4,1:13]
#>   year    brood   offspring rank        dam         sire      dam.type
#> 1 2018 BBW_7279 A59401-iOar    1 A58110-BBW  A59058-zMyr social parent
#> 2 2018 BBW_7279 A59401-iOar    2 A58110-BBW   A59372-sOb social parent
#> 3 2018 BBW_7279 A59401-iOar    3 A58110-BBW A59201-cNbwb social parent
#> 4 2018 BBW_7279 A59401-iOar   38 A58110-BBW  A58794-uWny social parent
#>                  sire.type hothiphop.parents hiphop hot.parents hot.dam
#> 1       extra-group parent                37      0          37      17
#> 2       extra-group parent               109     43          66      17
#> 3 within-group subordinate               122     70          52      17
#> 4             social parent              191    116          75      17
#>   hot.sire
#> 1       20
#> 2       53
#> 3       41
#> 4       70
```

Now we can see that only the female that was classified as social parent is considered as dam, while all males are considered as potential sire. The conditioning on one parent will facilitate the assignment of the parentage of the other parent, unless the unconditioned parent is incorrect (see later on how to verify this assumption). For some specific species (e.g. with sex role reversal) one may instead want to condition on the social father instead.

Finally, it is possible to increase the number of combinations being ranked, by changing the value of the top parameter (the default is top=3, which shows the top 3 combinations plus the social parents). For example:

```
best<-topmatch(x=combinations, ranking="hothiphop.parents", top=5)
best[1:6,1:7]
#>   year    brood   offspring rank          dam         sire           dam.type
#> 1 2018 BBW_7279 A59401-iOar    1   A58110-BBW  A59058-zMyr      social parent
#> 2 2018 BBW_7279 A59401-iOar    2   A58110-BBW   A59372-sOb      social parent
#> 3 2018 BBW_7279 A59401-iOar    3 A59012-zOogo  A59058-zMyr extra-group parent
#> 4 2018 BBW_7279 A59401-iOar    4   A58110-BBW A59201-cNbwb      social parent
#> 5 2018 BBW_7279 A59401-iOar    5  A58258-RNgn  A59058-zMyr extra-group parent
#> 6 2018 BBW_7279 A59401-iOar   55   A58110-BBW  A58794-uWny      social parent
```

Recommendations for the two different type of parentage analysis cases

Previously we identified two types of cases in which paternity analysis can be conducted. Next we provide recommendations on how to proceed in such cases:

1. Parentage assignment when one genetic parent is known: single parent analysis

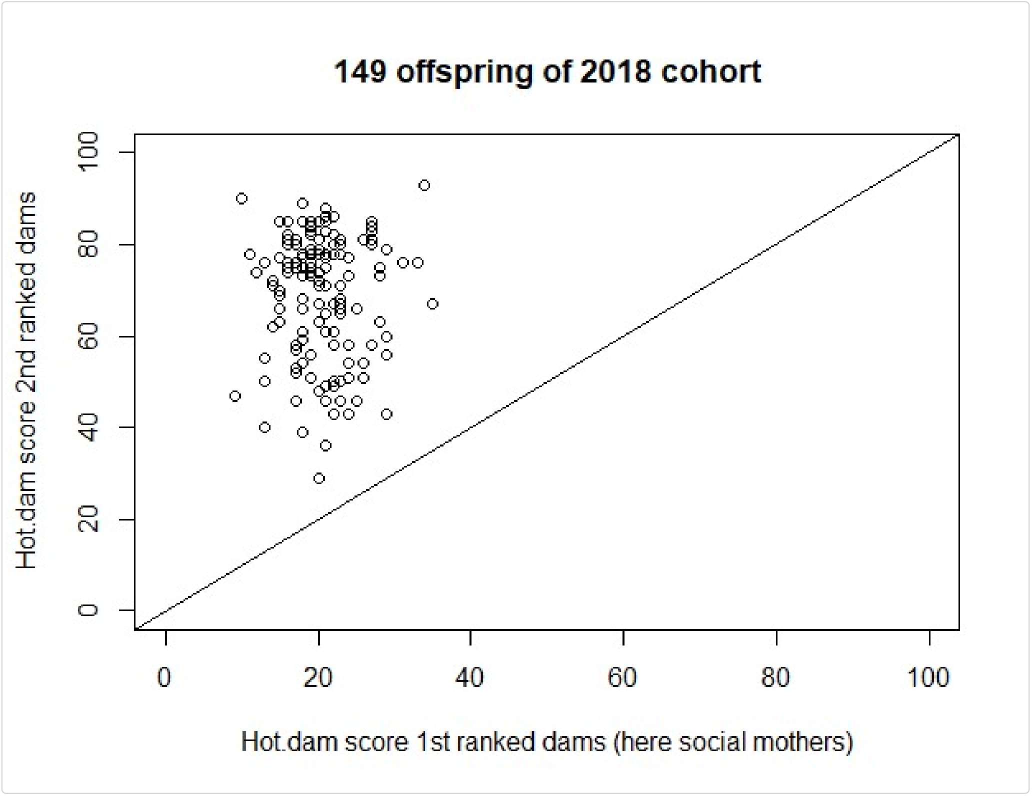
The classical procedure is to first verify the assumption that the mother is known (using the HOT.dam criteria), and then condition on the social mother to determine the most plausible sire, which we can do with either the HOTHIPHOP.parents or HIPHOP criteria. Which of these two ranking score is most useful may depend on the amount of genotyping error and rates of hetero- and homo-zygosity in the population. The choice is thus largely an empirical question and it is useful to consider both scores (for more discussion see Cockburn et al.).

Thus we first compare the dams with the lowest HOT.dam scores to verify whether there were any cases in which the social mother has more mismatches than expected due to genotyping error.

```
best.dams<-topmatch(x=combinations, ranking=c("hot.dam","hiphop"), unique="dam")
best.dams[1:8,1:13]
#>   year    brood   offspring rank          dam        sire           dam.type
#> 1 2018 BBW_7279 A59401-iOar    1   A58110-BBW A59058-zMyr      social parent
#> 2 2018 BBW_7279 A59401-iOar    1   A58110-BBW A58794-uWny      social parent
#> 3 2018 BBW_7279 A59401-iOar    2 A59012-zOogo A59058-zMyr extra-group parent
#> 4 2018 BBW_7279 A59401-iOar    3  A58258-RNgn A59058-zMyr extra-group parent
#> 5 2018 BBW_7279 A59402-iMow    1   A58110-BBW A59058-zMyr      social parent
#> 6 2018 BBW_7279 A59402-iMow    1   A58110-BBW A58794-uWny      social parent
#> 7 2018 BBW_7279 A59402-iMow    2 A59012-zOogo A59058-zMyr extra-group parent
#> 8 2018 BBW_7279 A59402-iMow    3  A58258-RNgn A59058-zMyr extra-group parent
#>            sire.type hothiphop.parents hiphop hot.parents hot.dam hot.sire
#> 1 extra-group parent                37      0          37      17       20
#> 2       social parent              191    116          75      17       70
#> 3 extra-group parent               117     47          70      53       20
#> 4 extra-group parent               144     59          85      72       20
#> 5 extra-group parent                33      1          32      18       14
#> 6       social parent              201    133          68      18       63
#> 7 extra-group parent               122     53          69      61       14
#> 8 extra-group parent               124     52          72      65       14
which(best.dams$rank==1 & best.dams$dam.type!="social parent")
#> integer(0)
```

We see that in this cohort there were no cases where the social mother was not the first ranked dam (i.e. with lowest hot.dam score).

```
best.triad<-best.dams[which(best.dams$rank==1),]
best.triad<-best.triad[!duplicated(best.triad[c("offspring","dam")]),] #we remove the duplicates,
         because if the dam with the lowest hot.dam score is the social mother she can be listed twice:
         once with the social father and one with the sire that gives the lowest HIPHOP score
plot(best.triad$hot.dam,best.dams$hot.dam[which(best.dams$rank==2)], xlim=c(0,100), ylim=c(0,100),
         xlab="Hot.dam score 1st ranked dams (here the social mothers)", ylab="Hot.dam score 2nd ranked
         dams", main="149 offspring of 2018 cohort")
abline(0,1)
```



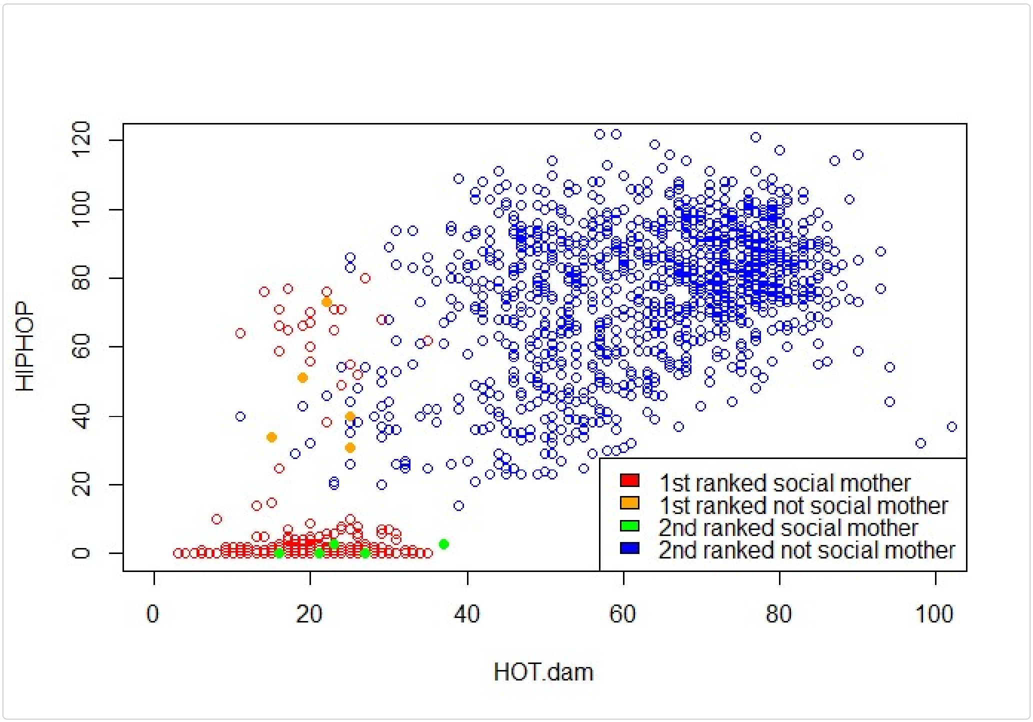
We can also see that these first ranked dams, which we already confirmed were the social mothers, had lower HOT.dam scores than the second ranked dams, but we note that there is one case in which the 2nd ranked dam has a HOT.dam score of 29 mismatches, which is within the range of HOT.dam scores of social mothers. In such situations, which typically involves close relatives of the social mother, the HOTHIPHOP.parents or HIPHOP score provides additional information to distinguish the true dam, but only if the true sire is sampled.

```
best.dams[which(best.dams$offspring=="A59419-iNgn"),1:13]
#>    year     brood   offspring rank         dam        sire           dam.type
#> 17 2018 rgMM_7278 A59419-iNgn    1 982682-rgMM A59187-lNyn      social parent
#> 18 2018 rgMM_7278 A59419-iNgn    1 982682-rgMM A59204-cGyn      social parent
#> 19 2018 rgMM_7278 A59419-iNgn    2 A30173-ngNA A59187-lNyn extra-group parent
#> 20 2018 rgMM_7278 A59419-iNgn    3 A30171-ynWA A59204-cGyn extra-group parent
#>             sire.type hothiphop.parents hiphop hot.parents hot.dam hot.sire
#> 17 extra-group parent                42      0          42      20       22
#> 18      social parent               177     96          81      20       71
#> 19 extra-group parent              68     20          48      29       22
#> 20      social parent             205     85         120      67       71
```

This output shows that although the 2nd ranked dam A30173-ngNA had a HOT.dam score close to that from the social mother 982682-rgMM (29 vs. 20), from the high HIPHOP score (20) it is clear she was not the genetic mother, but that the social mother was indeed the genetic mother (as she had 0 HIPHOP mismatches, which also indicates the true sire was sampled, as the HIPHOP score is calculated on a triad). Please note that first line of above ouput shows the social mother with the sire with whom she has the lowest HIPHOP score (in this case an extra-group sire), while the second line shows the social mother with the social father, which resulted in a many HIPHOP mismatches (96)

A more visual way of determining whether the social mother is also the genetic mother can be done by combining the above two steps in one plot. This approach is best illustrated on all cohorts. Below we provide the code to do this on all cohorts, but users can only run the 2018 cohort if this takes too long (if so, start at line with plot-function).

```
# first run the hothiphop on all cohorts
comb.18<-hothiphop(ind=subset(individuals,individuals$year==2018), gen=genotypes)
comb.17<-hothiphop(ind=subset(individuals,individuals$year==2017), gen=genotypes)
comb.16<-hothiphop(ind=subset(individuals,individuals$year==2016), gen=genotypes)
comb.15<-hothiphop(ind=subset(individuals,individuals$year==2015), gen=genotypes)
comb.14<-hothiphop(ind=subset(individuals,individuals$year==2014), gen=genotypes)
# then run topmatch function on all cohorts
best.18<-topmatch(x=comb.18, ranking=c("hot.dam","hiphop"), unique="dam")
best.17<-topmatch(x=comb.17, ranking=c("hot.dam","hiphop"), unique="dam")
best.16<-topmatch(x=comb.16, ranking=c("hot.dam","hiphop"), unique="dam")
best.15<-topmatch(x=comb.15, ranking=c("hot.dam","hiphop"), unique="dam")
best.14<-topmatch(x=comb.14, ranking=c("hot.dam","hiphop"), unique="dam")
#combine results
combinations.all<-do.call("rbind", list(comb.18, comb.17, comb.16,comb.15, comb.14))
best.dams<-do.call("rbind", list(best.18, best.17, best.16,best.15, best.14))
#remove the cases where the social mother was the first ranked dam, as then she is listed twice with
         rank 1
best.triad<-best.dams[which(best.dams$rank==1),]
best.triad<-best.triad[!duplicated(best.triad[c("offspring","dam")]),]
# plot the data
plot(best.triad$hot.dam ,best.triad$hiphop, col = "red", xlab="HOT.dam", ylab="HIPHOP", xlim=c(0,100),
         ylim=c(0,120))
points(best.dams$hot.dam[which(best.dams$rank==2 & best.dams$dam.type!="social
         parent")],best.dams$hiphop[which(best.dams$rank==2 &  best.dams$dam.type!="social parent")],
         col = "blue")
points(best.dams$hot.dam[which(best.dams$rank==1 & best.dams$dam.type!="social
         parent")],best.dams$hiphop[which(best.dams$rank==1 & best.dams$dam.type!="social parent")],
         col = "orange", pch = 19)
points(best.dams$hot.dam[which(best.dams$rank==2 & best.dams$dam.type=="social
         parent")],best.dams$hiphop[which(best.dams$rank==2 &  best.dams$dam.type=="social parent")],
         col = "green", pch = 19)
legend("bottomright",c("1st ranked social mother", "1st ranked not social mother", "2nd ranked social
         mother", "2nd ranked not social mother"), fill = c("red", "orange", "green", "blue"))
```



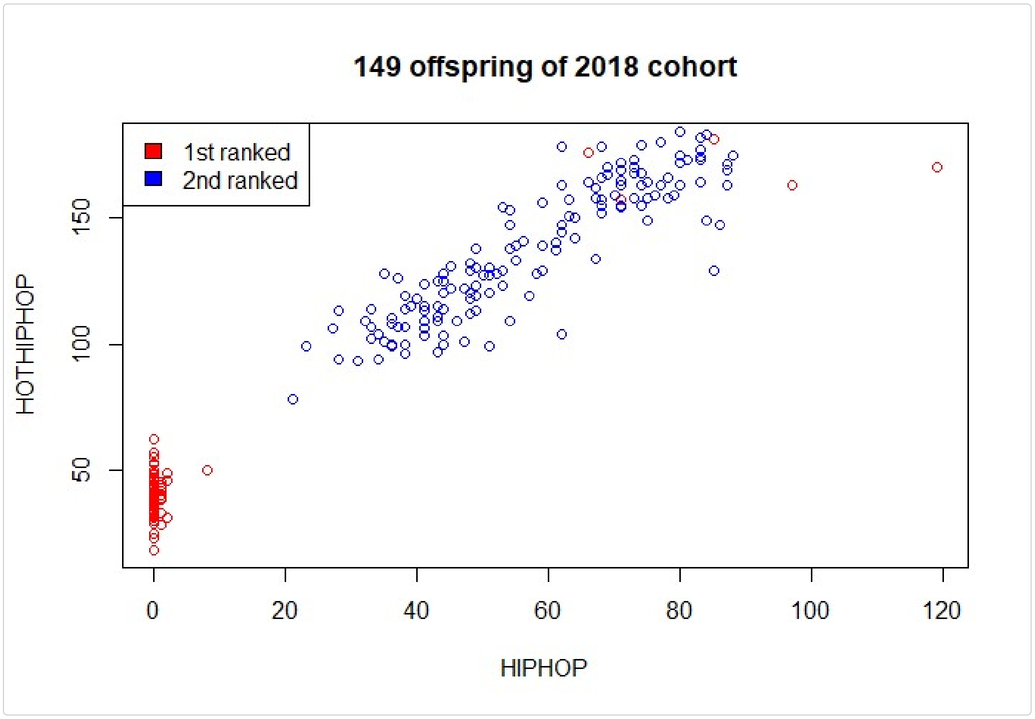
Here we plotted the HOT.dam and HIPHOP score of the first and second ranked dam (ranked according to HOT.dam score, with the best matching sire according to the HIPHOP score) against their HIPHOP score. We also use different colors depending on whether they are the social mother or not. We see that that first ranked social mothers typically have both low HOT.dam and HIPHOP scores, while the occasional second ranked female with low HOT.dam scores (<38 mismatches, within the range of social mothers) always had >16 HIPHOP mismatches. Furthermore, we can also see that there are ~25 1st ranked social mothers with low HOT.dam scores but high HIPHOP scores, which we will discuss in the next section are situations where the sire was not sampled. Finally, even in the rare case when the social mother is not the first ranked dam on the HOT score (green dots; 5 out of 1153 offspring), she is still clearly identifiable as the genetic mother due to the low HIPHOP score (and non-social mothers that are ranked 1st on HOT.dam score always have high HIPHOP scores).

Above figure emphasizes that the HOT test in itself is thus insufficient to distinguish parentage among closely related individuals (as there were ~55 blue 2nd ranked triads of non-social mothers with HOT.dam scores similar to social mothers), but the HIPHOP (and HOTHIPHOP.parents) score is capable of doing so.

Once it is confirmed that the assumption that the social mother is always the dam is highly plausible, then we can condition on the mother and rank the sires, using either HIPHOP or HOTHIPHOP.parents scores.

```
best.parents.hiphop<-topmatch(x=combinations, condition="mother", ranking="hothiphop.parents")
```

```
plot(best.parents.hiphop$hiphop[which(best.parents.hiphop$rank==1)],best.parents.hiphop$hothiphop.parents[which(best.parents.hi
        col = "red", xlab="HIPHOP", ylab="HOTHIPHOP.parents", main="149 offspring of 2018 cohort")
points(best.parents.hiphop$hiphop[which(best.parents.hiphop$rank==2)],best.parents.hiphop$hothiphop.parents[which(best.parents.
        col = "blue")
legend("topleft",c("1st ranked", "2nd ranked"), fill = c("red", "blue"))
```



We can see that in many cases the best ranked triad has 0 or 1 mismatches for the HIPHOP test score, suggesting that the true sire is identified in this triad. In this case both the HIPHOP and HOTHIPHOP.parents score distinguish well between the true sires (1st ranked sires) and others (2nd ranked sires). We note that in the 2018 data under consideration, there were five outliers with high mismatch rates (the five red dots in the blue cloud).

```
best.parents.hiphop[which(best.parents.hiphop$rank==1 & best.parents.hiphop$hiphop>20),c(1:13,22,23)]
#>    year     brood   offspring rank         dam        sire      dam.type
#> 9  2018 fBab_7322 A59439-iMwr    1 A58908-fBab A58909-fMrn social parent
#> 13 2018 fBab_7322 A59440-iOar    1 A58908-fBab A58484-sOnr social parent
#> 21 2018 sNab_7237 A59620-hGwn    1 A59357-sNab A59045-zGab social parent
#> 25 2018 sNab_7237  A59621-hYy    1 A59357-sNab  A59362-sBy social parent
#> 29 2018 sNab_7237 A59622-hBgr    1 A59357-sNab A59041-zGgw social parent
#>              sire.type hothiphop.parents hiphop hot.parents hot.dam hot.sire
#> 9  extra-group parent               163     97          66      16       59
#> 13 extra-group parent               170    119          51      14       48
#> 21 extra-group parent               181     85          96      27       82
#> 25 extra-group parent               157     71          86      29       69
#> 29 extra-group parent               176     66         110      35       93
#>    social.mother.sampled social.father.sampled
#> 9                      1                     0
#> 13                     1                     0
#> 21                     1                     1
#> 25                     1                     1
#> 29                     1                     1
```

These five cases are from two broods. Three chicks from sNab_7237 had HIPHOP mismatches ranging from 66 to 85. The nest was close to the edge of the study area, and we believe it was sired by a foreign (unsampled) male. The second brood (fBab_7322) of two chicks also had high mismatches (97 & 119); Cockburn et al. discuss this case in detail. The nest was close to the edge of the study area but the social father was unsampled (see 0 in column social.father.sampled), so in this case we cannot ascribe paternity, and also cannot say whether the chicks were within-pair or not.

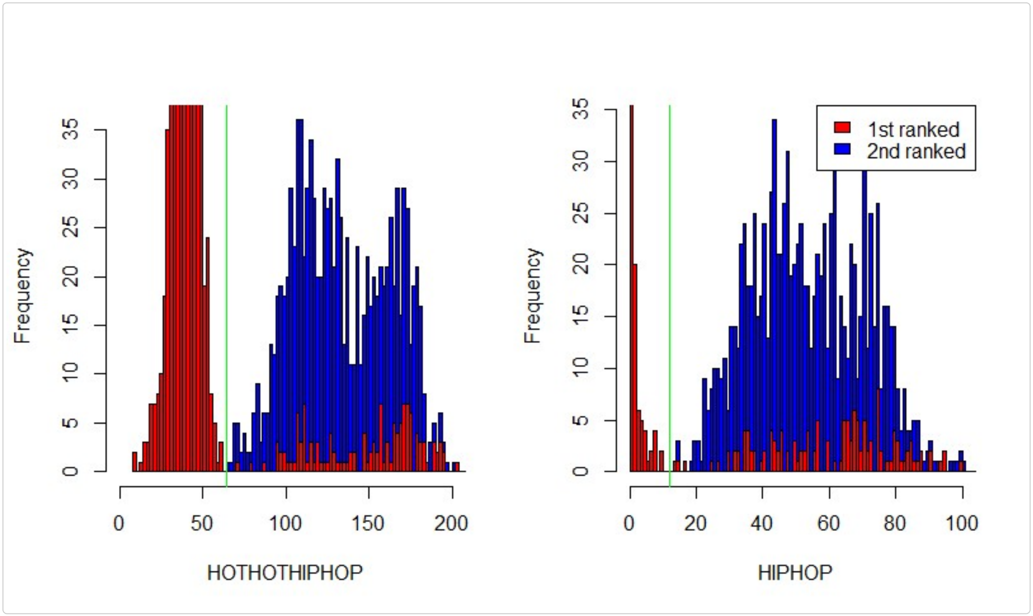
2. Parentage assignment when no genetic parents are known: parent pair analysis

To illustrate the procedure for parent pair assignment (i.e. none of the genetic parents are known from contextual information), we again use the superb fairy wren dataset, but will now assume that we have no a priori information about the genetic mother and will focus on both maternity and paternity assignment. The superb fairy wren dataset only consist of offspring for which the true dam was sampled, which is unrealistic for many other study systems. To illustrate how to deal with situations of less complete sampling (also for the males), we removed 20% of the social mothers and 20% of the males from the individuals dataset.

```
ff<-which(individuals$type=="adult female")
individuals.rem<-individuals[-ff[seq(5,length(ff),5)],] #this removes every fifth female in the
        individuals dataframe
mm<-which(individuals.rem$type=="adult male")
individuals.rem<-individuals.rem[-mm[seq(5,length(mm),5)],] #this removes every fifth male in the
        individuals dataframe
```

When both social parents are unknown, the aims is to assign both maternity and paternity. The first step is to determine whether we get a clearly separated bi-modal distribution of HOTHIPHOP.parents or HIPHOP scores between the first ranked and higher ranked parents. Ideally this is done on a large dataset, so on all cohorts available (to skip the long run time, one could instead subset the individuals.rem dataframe below to the year 2018).

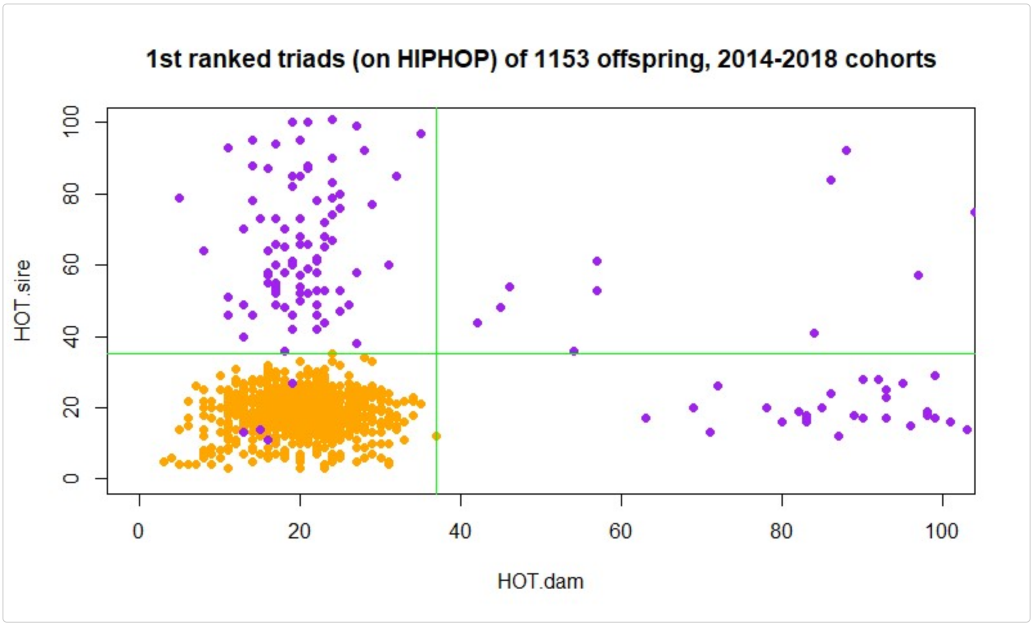
```
combinations.rem<-hothiphop(ind=individuals.rem, gen=genotypes)
best.hhhh<-topmatch(x=combinations.rem, ranking="hothiphop.parents")
h1 <- hist(best.hhhh[which(best.hhhh$rank==1), "hothiphop.parents"], breaks = 100)
h2 <- hist(best.hhhh[which(best.hhhh$rank==2), "hothiphop.parents"], breaks = 50)
plot(h2, col = "blue", xlim = c(0, 200),  xlab="HOTHIPHOP.parents", main=NULL)
plot(h1, col = "red", xlim = c(0, 200), add = T)
legend("topright",c("1st ranked", "2nd ranked"), fill = c("red", "blue"))
```



As case 2 concerns situations where the social parents are unknown, we have no information a priori whether the 1st ranked individuals are indeed the true dam-sire combinations, or whether due to genotyping error some of the true combinations ended up as 2nd or higher ranked. However from both histograms it is clear there is a bi-modal distribution, and in such situations we can make plausible assumptions that if only first ranked individuals are in the left peak that these are likely only true parents. Specifically, we see that there is a red peak at the left with low <12 HIPHOP (or <64 HOTHIPHOP.parents in left histogram) mismatch scores, which likely reflect the true dam-sire triads with mismatches due to genotyping error and mutations. In addition we have a blue peak with higher scores that are likely to reflect incorrect combinations, as these combinations have (much) higher mismatch scores (due to incompatible genotypes as well as genotyping error and mutations).

There are quite some red bars (1st ranked triads) within the peak of blue bars (2nd ranked triads). Based on above rationale it follows that these are incorrect 1st ranked combinations that may reflect cases where the true sire (and/or dam) was not sampled (in fact most of these are the offspring for which we removed the social mothers and sires at the start of this section). For these combinations we could look at the HOT.dam and HOT.sire scores to determine whether the sire or dam was likely to be unsampled (or both).

```
plot(best.hhhh[which(best.hhhh$rank==1 & best.hhhh$hiphop<12),"hot.dam"]
        ,best.hhhh[which(best.hhhh$rank==1 & best.hhhh$hiphop<12),"hot.sire"], col ="orange" ,
        xlab="HOT.dam", ylab="HOT.sire", xlim=c(0,100), ylim=c(0,100), pch = 19, main="1st ranked
        triads (on HIPHOP) 2018 cohort")
points(best.hhhh[which(best.hhhh$rank==1 & best.hhhh$hiphop>=12),"hot.dam"]
        ,best.hhhh[which(best.hhhh$rank==1 & best.hhhh$hiphop>=12),"hot.sire"], col ="purple", pch =
        19 )
abline(v=37, h=35, col="green")
legend("topright",c("HIPHOP < 12", "HIPHOP >= 12"), fill = c("orange", "purple"))
```



We have plotted the first ranked triads (according to HIPHOP score) for all 1153 offspring 2014-2018 cohorts. Based on above histograms, triads with HIPHOP scores lower than 12 (orange) are presumed to be the true parents, while triads with higher HIPHOP scores are assumed to be cases where at least one of the true parents was not sampled. From this figure we can see that true triads never have HOT.dam and HOT.sire scores higher than 37 and 35 respectively (see green lines in figure above), and we can use this information to identify which parent was unsampled.

1. The purple triads in the top left quadrant represents cases where the sire was unsampled.
2. The purple triads in the bottom right quadrant represents cases where the dam was unsampled.
3. The purple triads in the top right quadrant represents cases where we both the dam and sire were unsampled.
4. We can also see that there are four purple triads in the bottom left quadrant that have HOT.dam and HOT.sire scores that are similar to true dams and sires. This highlights again that HOT.dam and HOT.sire scores by themselves are not conclusive for assigning parentage in this dataset, because their HIPHOP scores were too high for them to be the true parents. These triads likely represent cases where either the true sire and/or true dam was unsampled, but instead a close relative was ranked first. Thus a high HOT.dam above the green threshold is evidence that the dam was unsampled, but a HOT.dam value below the threshold is not conclusive evidence that the true dam was sampled (same rationale holds for using HOT.sire to assign sires). This implies that for purple triads in the right two quadrants we can be certain the true dam was not sampled, but we cannot be certain that the sire in that triad was the true sire, as there is a small chance a closely related unsampled male was the true sire instead. Similarly, for purple triads in the top two quadrants we can be certain the true sire was not sampled, but we cannot be certain that the dam in that triad was the true sire, as there is a small chance a closely related unsampled female was the true dam. In situations where there would be no purple triads in the bottom left quadrant, we would be able to assign the parent even if one of the parents was unsampled.

We end by noting that the parent-pair method we described above could also be used for the single parent assignment case for this dataset, but that in contrast to the single parent assignment approach described earlier it does not require one to look at HOT.dam scores:

1. We look at the histogram of HIPHOP (or HOTHIPHOP.parents) scores of 1st and 2nd ranked triads and determine if there are two peaks separated by a gap, which can be used to identify a threshold.
2. We identify all triads which have HIPHOP scores less than this threshold, for these triads we have identified the true sires and dams.
3. We check whether all dams in these triads identified in step 2 are social mothers.
4. If so, this means that all triads above the HIPHOP threshold in which the social mother was sampled must be cases where the sire was unsampled, while if the social mother was unsampled, we can not assign the sire with complete certainty in this dataset.

In conclusion

The aim of the HIPHOP package/ approach is to exclude potential parents by determining if they have more mismatches than can be expected due to genotyping errors and mutation, and thereby identify (i) the true genetic parents and (ii) detect situations where one (or both) of the true parents is not sampled, in which case we (iii) want to identify which parent can still be correctly assigned. In this vignette we showed that both single parent and parent pair assignment based on HIPHOP or HOTHIPHOP.parents can assign genetic parents correctly, while this is not possible using only the HOT test. We can also identify offspring for which parent(s) were unsampled, but in this dataset we cannot with complete certainty identify a parent correctly if the other parent was not sampled (though if the sire (or dam) has a HOT.sire (HOT.dam) score in the range of what is typical for true sires (dams), we still have high certainty, but we just cannot exclude that a closely related unsampled parent may exist that was the true sire (or dam)). Assigning parents if one of the parents is not sampled, may be easier to do (i) in species that do not live among close relatives (superb fairy wrens have strong spatial genetic clustering due to their extreme male philopatry), (ii) if more loci are available for scoring, or (iii) if genotyping error is reduced, as the latter two may separate the HOT scores of true sires (or dams) from closely related sires (or dams) more clearly.

Troubleshooting

Asking questions, suggesting improvements and reporting bugs

Please post all queries to the R-hiphop google group at https://groups.google.com/d/forum/r-hiphop

Dealing with very large datasets

The hothiphop function checks all possible offspring-potential.dam-potential.sire combinations. If the number of possible dams, sires and/or offspring is very large the number of combinations will become very high, which may lead to memory problems in R. In such cases a workaround is to limit the number of offspring in the individuals dataset to a smaller number (but still include all potential dams and sires). For example, one could include the first 10 offspring in the individuals file, run the hothiphop function, next run the hothiphop function for the next 10 offspring, and append the results to the results from the previous set (parallelization is another option).