APIS: An Auto-Adaptive Parentage Inference Software tolerant to missing parents

Ronan GRIOT [1,2], *François ALLAL* [3], *Marc VANDEPUTTE* [2,3]

[1] *SYSAAF, Station LPGP/INRA, Campus de Beaulieu, Rennes, France*
[2] *GABI, INRA, AgroParisTech, Université Paris-Saclay, France*
[3] *MARBEC, Univ. Montpellier, Ifremer, CNRS, IRD, Palavas-les-Flots, France*

2019-07-15

Description

This package include all the functions to assign with APIS (Griot & al., 2019). Parentage assignment is widely used for farmed and natural populations. As most of the likelihood software are based on simulation, the estimation of the simulation parameters is a key point for assignment reliability. Among those parameters, the proportion of missing parent is one of the most important. To avoid estimation of missing parents, we developed APIS (Auto-Adaptive Parentage Inference Software), based on observed average Mendelian transmission probabilities. In this package, you will find all the functions to perform parentage assign with the method described in the paper.

Install and load the package

```
library(devtools)
install_github("rgriot/APIS", dependencies = T)
library(APIS)
```

If an error message occurs during the installation, use the following command :

```
devtools::install_github("rgriot/APIS", args = "--no-multiarch")
library(APIS)
```

Formate your data

APIS requires matrices of characters as inputs. Each matrix has individuals as rows, markers as columns. The individual labels are set as rownames. Each cell is the genotype of one marker, coded "All1/All2". For example "A/A", "A/B", "B/B" for bi-allelic markers and "NA/NA" for missing value. For multi-allelic markers, use the generic coding "All1/All2".

```
data("APIS_offspring")
data("APIS_sire")
data("APIS_dam")

head(APIS_offspring[,1:10])
#>             marker_1 marker_2 marker_3 marker_4 marker_5 marker_6 marker_7
#> offspring_1 "A/B"    "A/A"    "A/B"    "A/A"    "B/B"    "A/B"    "A/A"
#> offspring_2 "A/B"    "A/A"    "A/B"    "B/B"    "A/B"    "A/A"    "A/A"
#> offspring_3 "A/B"    "A/B"    "A/B"    "B/B"    "B/B"    "A/B"    "A/B"
#> offspring_4 "A/A"    "B/B"    "B/B"    "A/B"    "B/B"    "A/B"    "A/B"
#> offspring_5 "A/B"    "A/A"    "A/B"    "A/A"    "B/B"    "A/B"    "A/A"
#> offspring_6 "A/B"    "A/B"    "B/B"    "A/A"    "B/B"    "B/B"    "A/A"
#>             marker_8 marker_9 marker_10
#> offspring_1 "A/B"    "A/B"    "A/A"
#> offspring_2 "B/B"    "A/B"    "A/A"
#> offspring_3 "A/B"    "A/B"    "B/B"
#> offspring_4 "B/B"    "A/B"    "B/B"
#> offspring_5 "A/A"    "A/A"    "A/A"
#> offspring_6 "B/B"    "A/B"    "A/B"
rownames(APIS_offspring[1:6,])
#> [1] "offspring_1" "offspring_2" "offspring_3" "offspring_4" "offspring_5"
#> [6] "offspring_6"
```

Prepare the inputs

APIS main function requires 4 inputs :

- off.genotype = matrix of offspring genotypes coded as explained above
- sire.genotype = matrix of sires genotypes coded as explained above
- dam.genotype = matrix of dams genotypes coded as explained above
- error = accepted assignment error rate (What is the maximum error rate I accept in my assignment results ?)

```
head(APIS_offspring[,1:10])
#>             marker_1 marker_2 marker_3 marker_4 marker_5 marker_6 marker_7
#> offspring_1 "A/B"    "A/A"    "A/B"    "A/A"    "B/B"    "A/B"    "A/A"
#> offspring_2 "A/B"    "A/A"    "A/B"    "B/B"    "A/B"    "A/A"    "A/A"
#> offspring_3 "A/B"    "A/B"    "A/B"    "B/B"    "B/B"    "A/B"    "A/B"
#> offspring_4 "A/A"    "B/B"    "B/B"    "A/B"    "B/B"    "A/B"    "A/B"
#> offspring_5 "A/B"    "A/A"    "A/B"    "A/A"    "B/B"    "A/B"    "A/A"
#> offspring_6 "A/B"    "A/B"    "B/B"    "A/A"    "B/B"    "B/B"    "A/A"
#>             marker_8 marker_9 marker_10
#> offspring_1 "A/B"    "A/B"    "A/A"
#> offspring_2 "B/B"    "A/B"    "A/A"
#> offspring_3 "A/B"    "A/B"    "B/B"
#> offspring_4 "B/B"    "A/B"    "B/B"
#> offspring_5 "A/A"    "A/A"    "A/A"
#> offspring_6 "B/B"    "A/B"    "A/B"
head(APIS_sire[,1:10])
#>        marker_1 marker_2 marker_3 marker_4 marker_5 marker_6 marker_7
#> sire_1 "A/A"    "A/B"    "A/B"    "B/B"    "B/B"    "B/B"    "A/B"
#> sire_2 "A/B"    "A/A"    "A/B"    "A/B"    "B/B"    "B/B"    "A/B"
#> sire_3 "B/B"    "A/B"    "A/B"    "A/B"    "A/B"    "A/A"    "A/B"
#> sire_4 "A/A"    "A/B"    "A/B"    "B/B"    "B/B"    "A/B"    "A/B"
#> sire_5 "A/B"    "A/B"    "B/B"    "A/B"    "B/B"    "A/A"    "A/B"
#> sire_6 "B/B"    "A/A"    "B/B"    "A/A"    "B/B"    "A/B"    "A/A"
#>        marker_8 marker_9 marker_10
#> sire_1 "A/B"    "A/B"    "A/A"
#> sire_2 "A/B"    "B/B"    "A/B"
#> sire_3 "B/B"    "A/B"    "A/B"
#> sire_4 "B/B"    "A/B"    "B/B"
#> sire_5 "A/B"    "A/A"    "B/B"
#> sire_6 "A/A"    "A/B"    "A/B"
head(APIS_dam[,1:10])
#>       marker_1 marker_2 marker_3 marker_4 marker_5 marker_6 marker_7
#> dam_1 "A/B"    "A/A"    "B/B"    "A/A"    "B/B"    "A/B"    "A/A"
#> dam_2 "A/A"    "A/A"    "A/B"    "A/B"    "A/B"    "A/B"    "A/A"
#> dam_3 "A/B"    "A/A"    "A/B"    "A/B"    "A/B"    "A/A"    "A/A"
#> dam_4 "A/B"    "A/B"    "A/B"    "B/B"    "B/B"    "A/B"    "A/A"
#> dam_5 "A/A"    "A/A"    "A/B"    "A/B"    "A/B"    "A/A"    "A/A"
#> dam_6 "A/B"    "A/A"    "A/B"    "A/B"    "B/B"    "A/B"    "A/B"
#>       marker_8 marker_9 marker_10
#> dam_1 "A/B"    "A/B"    "A/A"
#> dam_2 "B/B"    "B/B"    "A/B"
```

```
#> dam_3 "B/B"    "A/B"    "A/B"
#> dam_4 "A/B"    "B/B"    "B/B"
#> dam_5 "A/A"    "A/B"    "A/A"
#> dam_6 "A/B"    "A/B"    "A/B"
error <- 0.05 #I accept 5% of errors in the results
```

Running the assignment

The main function to perform parentage assignment with APIS the "APIS" function. Use the function as below, with default parameters for exclusion threshold and preselection of parents for maximizing the reliability.

```
result <- APIS(off.genotype = APIS_offspring,
               sire.genotype = APIS_sire,
               dam.genotype = APIS_dam,
               error = error)
```

Analyse the results

APIS gives you 3 different outputs :

- pedigree

Pedigree header	Description
off	Offspring ID
sire	Sire ID
dam	Dam ID

- log containing Mendelian transmission probabilities, mismatches and deltas for the first 3 parent pairs

Log header	Description
offspring	offspring ID
mrk_genotype	number of markers genotyped
sire1	ID of the most likely sire
dam1	ID of the most likely dam
mismatch1	number of mismatches for the most likely parent pair (sire1, dam1)
mendel1	average Mendelian transmission probability of the most likely parent pair (sire1, dam1)
sire2	ID of the second most likely sire
dam2	ID of the second most likely dam
mismatch2	number of mismatches for the second most likely parent pair (sire2, dam2)
mendel2	average Mendelian transmission probability of the second most likely parent pair (sire2, dam2)
delta_Pmendel12	mendel1 - mendel2
sire3	ID of the third most likely sire
dam3	ID of the third most likely dam

Log header	Description
mismatch3	number of mismatches for the third most likely parent pair (sire3, dam3)
mendel3	average Mendelian transmission probability of the third most likely parent pair (sire3, dam3)
delta_Pmendel23	mendel2 - mendel3

- graphs of the distributions of deltas, Mendelian transmission probabilities and mismatches

According to the graphs, you can change the thresholds to improve your assignment.

If you want to set up your threshold on Mendelian probabilities, use :

```
new.result <- personalThreshold(APIS.result = result,
                                method = 'Pmendel',
                                threshold = 0.7)
```

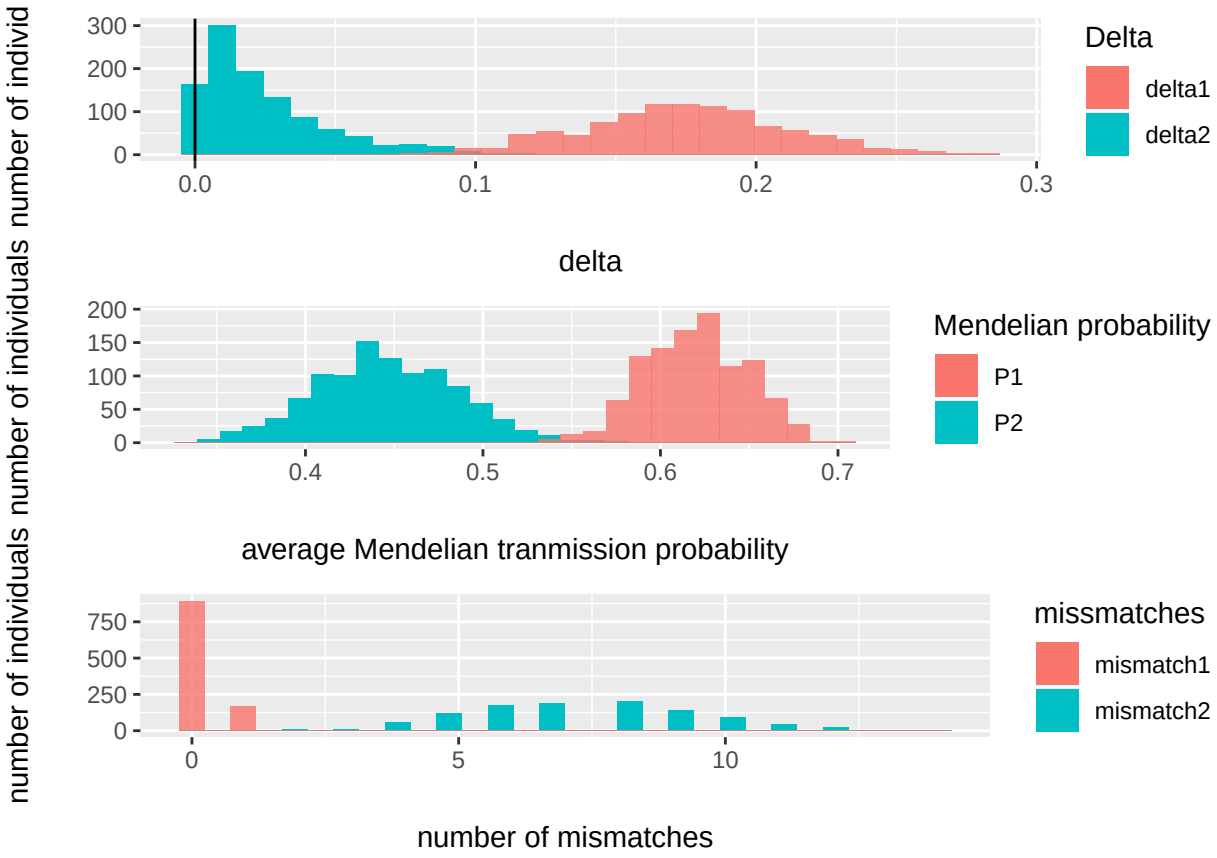
If you want to set up your threshold on mismatches, use :

```
new.result <- personalThreshold(APIS.result = result,
                                method = 'exclusion',
                                threshold = 1)
```

Examples

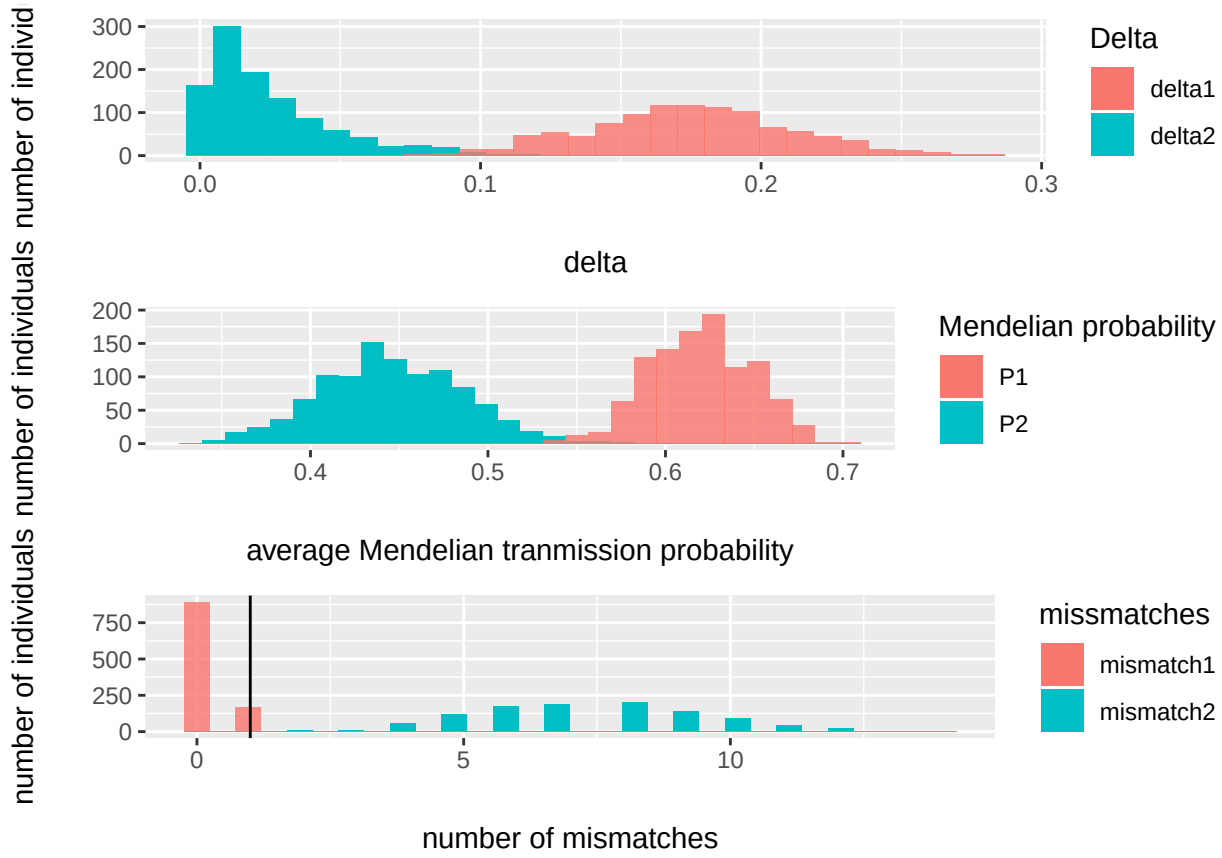
Full data

This example uses the 100 markers data sets provided by the package. This example set is from Griot & al, (2019)



When I look at the mismatch distributions, I prefer to use exclusion and allow for 1 mismatches (Figure 2).

```
new.result <- personalThreshold(APIS.result = result,
                                method = 'exclusion',
                                threshold = 1)
```

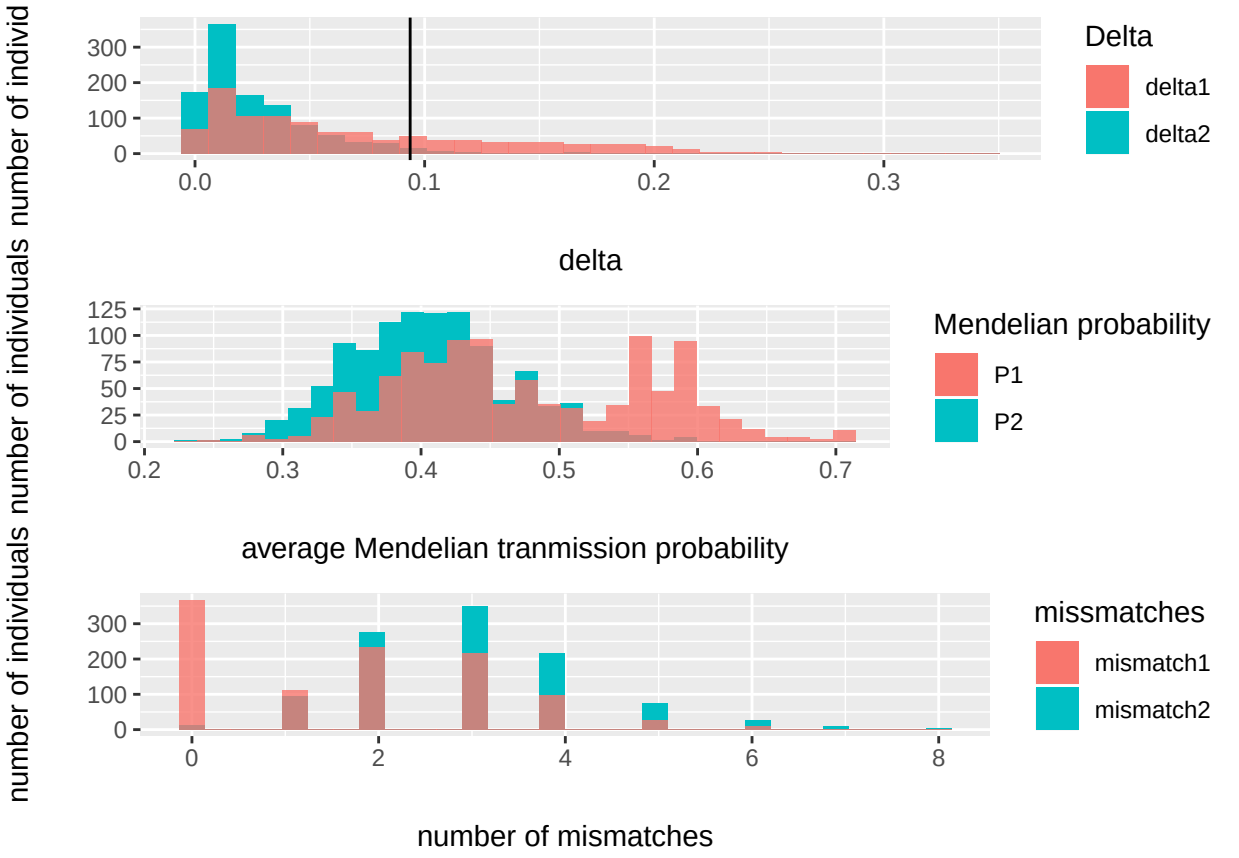



```
#> ---------------------------------------
#>               APIS SUMMARY
#> ---------------------------------------
#> Method for personal threshold : exclusion
#> Threshold : 1
#> Assignment rate : 99.531%
```

Degraded data

This example uses 35 markers from the example set provided by the package.

If you want to use similar power as those used in Griot & al, (2019), you can subet the 35, 42 or 50 first markers to reach a power of 0.90, 0.95, 0.99.



In this situation, the theoretical assignment power is low and there are missing parents. The distribution graphs do not give you more information about a new threshold value.

Thus, the better option is to keep APIS results.

Other parameters of APIS function

APIS function can handle 2 other parameters :

- exclusion.threshold : For exclusion procedure, the number of mismatches allowed by the user. Use default value for Mendelian transmission probability procedure.
- preselect.Parent : Use of a parent preselection function based on Mendelian incompatibilities.
  - Default value : FALSE
  - One interger (n) : Selection of n sires and n dams
  - Two integers (n, m) : Selection of n sires and m dams

- nb.cores : Number of cores to use. By default, only 2 cores are used. If your computer has more cores, you can use the function detectCores() from "parallel" package (Warnings: Do not use all the available cores, if your computer has 8 cores, only use 6 or 7 for the analysis)

Use default value to get the most accurate results (except for the number of cores). If a value is specified, this will decrease computation time but can decrease assignment reliability.

Acknowlegments

This work was partially financially supported in the GeneSea project (n° R FEA 4700 16 FA 100 0005) by the French Government and the European Union (EMFF, European Maritime and Fisheries Fund) at the "Appels à projets Innovants" managed by the FranceAgrimer Office. The doctoral scholarship of Ronan Griot was partially supported by the ANRT (doctoral scholarship n° 2017/0731) and SYSAAF.

Annexe

```
print(sessionInfo(), locale=FALSE)
#> R version 3.6.1 (2019-07-05)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 7 x64 (build 7601) Service Pack 1
#>
#> Matrix products: default
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#> [1] APIS_0.1.0     devtools_2.1.0 usethis_1.5.1
#>
#> loaded via a namespace (and not attached):
#>  [1] xfun_0.8          remotes_2.1.0     colorspace_1.4-1
#>  [4] doSNOW_1.0.16     testthat_2.1.1    htmltools_0.3.6
#>  [7] snow_0.4-3        yaml_2.2.0        rlang_0.4.0
#> [10] pkgbuild_1.0.3    pillar_1.4.2      glue_1.3.1
#> [13] withr_2.1.2       sessioninfo_1.1.1 foreach_1.4.4
#> [16] stringr_1.4.0     munsell_0.5.0     gtable_0.3.0
#> [19] codetools_0.2-16  memoise_1.1.0     evaluate_0.14
#> [22] labeling_0.3      knitr_1.23        callr_3.3.0
#> [25] doParallel_1.0.14 ps_1.3.0          curl_3.3
#> [28] parallel_3.6.1    Rcpp_1.0.1        backports_1.1.4
#> [31] scales_1.0.0      desc_1.2.0        pkgload_1.0.2
#> [34] fs_1.3.1          gridExtra_2.3     ggplot2_3.2.0
#> [37] digest_0.6.20     stringi_1.4.3     processx_3.4.0
#> [40] rprojroot_1.3-2   grid_3.6.1        cli_1.1.0
#> [43] tools_3.6.1       magrittr_1.5      lazyeval_0.2.2
#> [46] tibble_2.1.3      crayon_1.3.4      pkgconfig_2.0.2
#> [49] prettyunits_1.0.2 assertthat_0.2.1  rmarkdown_1.14
#> [52] iterators_1.0.10  R6_2.4.0          compiler_3.6.1
```