WorldFish

November, 2025

OCFam (Version 1.6): An R package for optimal contribution selection in highly fecund species

WorldFish
YEARS 1975-2025

CGIAR
SUSTAINABLE ANIMAL AND AQUATIC FOODS

Citation
Hamilton, MG. 2025. OCFam (Version 1.6): An R package for optimal contribution selection in highly fecund species. WorldFish. Penang, Malaysia.

About WorldFish
WorldFish is a nonprofit research and innovation institution that creates, advances and translates scientific research on aquatic food systems into scalable solutions with transformational impact on human well-being and the environment. Our research data, evidence and insights shape better practices, policies and investment decisions for sustainable development in low- and middle-income countries.

We have a global presence across 20 countries in Asia, Africa and the Pacific with 460 staff of 30 nationalities deployed where the greatest sustainable development challenges can be addressed through holistic aquatic food systems solutions.

Our research and innovation work spans climate change, food security and nutrition, sustainable fisheries and aquaculture, the blue economy and ocean governance, One Health, genetics and AgriTech, and it integrates evidence and perspectives on gender, youth and social inclusion. Our approach empowers people for change over the long term: research excellence and engagement with national and international partners are at the heart of our efforts to set new agendas, build capacities and support better decision-making on the critical issues of our times.

WorldFish is part of One CGIAR, the world's largest agricultural innovation network.

Acknowledgments
This work was conducted as part of the CGIAR Sustainable Animal and Aquatic Foods (SAAF) Science Program. CGIAR research is supported by contributions to the CGIAR Trust Fund. CGIAR is a global research partnership for a food-secure future dedicated to transforming food, land, and water systems in a climate crisis.

Contact
WorldFish Communications and Marketing Department, Jalan Batu Maung, Batu Maung, 11960 Bayan Lepas, Penang, Malaysia. Email: worldfishcenter@cgiar.org

Creative Commons License

Content in this publication is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0), which permits non-commercial use, including reproduction, adaptation and distribution of the publication provided the original work is properly cited.

© 2025 WorldFish.

Photo credits
Front cover, Matthew Hamilton.

Disclaimer
This publication has not gone through the standard WorldFish science-review procedure. The opinions expressed here belong to the authors, and do not necessarily reflect those of the CGIAR Sustainable Animal and Aquatic Foods (SAAF) Science Program, WorldFish or CGIAR

Table of contents

1. Introduction

OCFam is an R package implementing optimal contribution selection (OCS) for highly fecund species (e.g., fish), where large family sizes and rounding constraints renders classical individual-level OCS impractical. The package provides two primary functions:

1. *OCFamPrep()*: Prepares pedigree data and summarizes historical trends in inbreeding and coancestry.
2. *OCFam()*: Performs optimal contribution selection and generates integer-feasible allocations of parents.

OCFam addresses the gap between the continuous mathematical optimum and operational constraints on parental contributions, ensuring that selected contributions are close to optimal and practically implementable. The package supports both non-overlapping (discrete) and overlapping generation breeding systems.

2. Installation of OCFam

R installation

1. Install R from CRAN (https://cran.r-project.org/)
2. Install the corresponding version of Rtools from CRAN Rtools (required for building packages from source on Windows) (https://cran.r-project.org/bin/windows/Rtools/).
3. (Optional) Install RStudio Desktop from Posit for an enhanced development environment (https://posit.co/download/rstudio-desktop/).

OCFam installation

OCFam is installed from GitHub. Start or restart R, after installing Rtools, and then enter the following R code to install *OCFam* and its dependencies:

```
if (!requireNamespace("optiSel", quietly = TRUE))
install.packages("optiSel")
if (!requireNamespace("AGHmatrix", quietly = TRUE))
install.packages("AGHmatrix")
if (!requireNamespace("dplyr",   quietly = TRUE))
install.packages("dplyr")
if (!requireNamespace("ggplot2",  quietly = TRUE))
install.packages("ggplot2")
if (!requireNamespace("devtools", quietly = TRUE))
install.packages("devtools")

# Install OCFam from GitHub
devtools::install_github("mghamilton/OCFam", dependencies =
TRUE, force = TRUE)

library(OCFam)
```

3. Typical workflow

A typical breeding cycle (i.e. the formation of a new age class) using *OCFam* comprises the following sequential steps (see Annex 1 for examples):

1. Preparation and validation of input data.
2. Evaluation of historical inbreeding and coancestry patterns using *OCFamPrep()*.
3. Specification of breeding design parameters (e.g. kinship constraints, required numbers of sires and dams by sex–age class, etc).
4. Execution of *OCFam()* to determine the optimal set of contributing parents and their respective contributions.
5. Development of a mating plan, preferably using a specialised mating design algorithm (e.g. *AllocateMate*) (Hamilton 2024).

4. *OCFamPrep*

The *OCFamPrep()* function generates tables and a plot to enable analysis of historical trends in average coancestry (k) and inbreeding (Wright's inbreeding coefficient, F) across age classes (Caballero and Toro 2000; Meuwissen 1997; Meuwissen and Sonesson 1998; Wright 1922). These summaries support the definition of appropriate kinship constraints for the next breeding cycle (age class) using the *OCFam()* function (see slides 12-18 in Annex 1). Details of function arguments and outputs, together with basic usage examples, are provided in the R help documentation, available in Annex 2 or by executing the following code in R:

```
help("OCFamPrep")
```

Function arguments

OCFamPrep() function input arguments are:

1. *ped*: individual-level data frame with the following column headings: 'INDIV', 'SIRE', 'DAM', 'FAM', 'BORN' and 'EBV'.
2. *age_class_names*: an optional data frame specifying names for age classes.
3. *gene_flow_vector*: only necessary in the case of overlapping generations.

Function outputs

The output of the *OCFamPrep()* function is a R list with the following named components (elements):

1. *plot*: is plot of trends in coancestry and inbreeding coefficients. The number of families per age class is also displayed (Figure 1).
2. *age_class_means*: is a data frame with the following column headings: 'BORN', 'INBREEDING', 'COANCESTRY', 'OVERLAPPING_COANCESTRY'
3. *age_class_K_mat*: is a matrix containing average coancestries within age classes on the diagonals and between age classes on the off-diagonals. Row and column names are the FAM identifiers from *ped*.
4. *fam_K_matrix*: is a data frame containing the between family coancestry matrix (half of the Between-Family Relationship Matrix described in Hamilton (2020)).

5. *family_inbreeding*: is a data frame listing inbreeding coefficients (F) for individuals within families with the following column headings: ‘FAM’ and ‘F’.
6. *ped*: is a data frame with the following column headings: 'INDIV', 'SIRE', 'DAM', 'FAM' 'BORN', 'EBV', 'FAM_SIRE', 'EBV_SIRE', 'FAM_DAM', 'EBV_DAM' and 'EBV_FAM'.
7. *family_ped*: is a data frame with the following columns: 'FAM', 'SIRE', 'DAM', 'BORN', 'FAM_SIRE', 'EBV_SIRE', 'FAM_DAM', 'EBV_DAM' and 'EBV_FAM'.
8. *gene_flow_vector*: is applicable to overlapping generations only (if NA, discrete generations are assumed).
9. *r*: is a weight vector of age classes; r[i] denotes the long-term contribution of age class i (until lifetime is reached) (Meuwissen and Sonesson 1998).
10. *gen_interval*: is the generation interval.

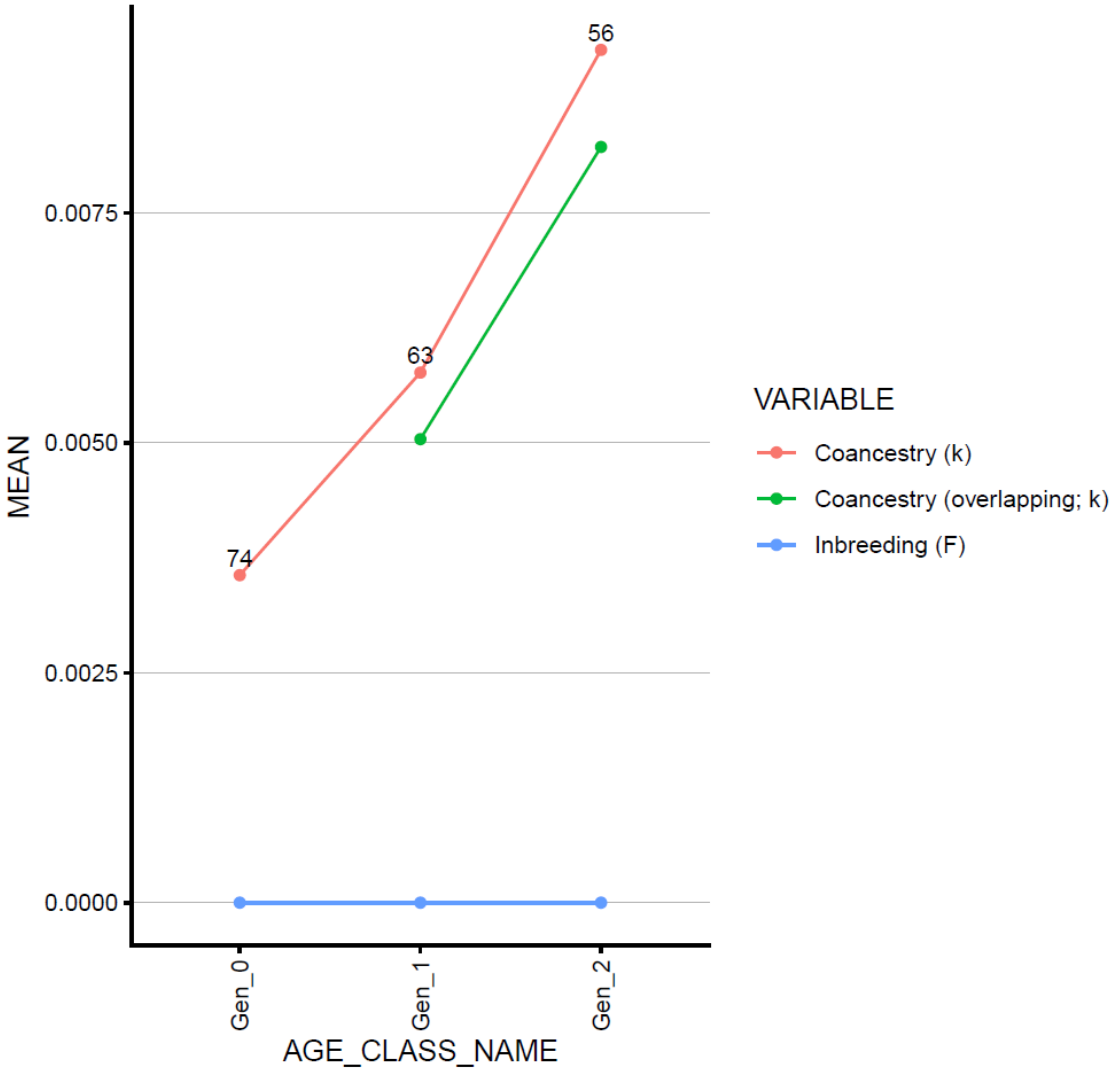


Figure 1. Example of a plot generated by the *OCFamPrep()* function, with the gene_flow_vector specifying 20% parental contributions from the second most recent age class and 80% from the most recent age class. The numbers displayed in the plot represent the number of full-sibling families within each age class.

5. *OCFam*

The *OCFam()* function (see slides 19-23 in Annex 1):

1. Solves the optimisation problem using the optiSel package (Wellmann 2019) to determine continuous optimal contributions for each candidate parent.
2. Aggregates non-fixed individual contributions by full-sibling family (i.e. the families of candidate parents).
3. Identifies families whose summed contributions exceed a predefined threshold.
4. Fixes the contribution of the highest-ranking non-fixed individual within each selected family, subject to specified constraints (e.g., sex-age class contributions). Fixed values represent an integer contribution to the next generation (e.g., half divided by the required number of sires [or dams]).
5. Repeats steps 1–4 with updated fixed contributions until all contributions are integer-feasible within the specified constraints. The threshold for family selection (Step 3) is reduced at each iteration.

Details of *OCFam()* function arguments and outputs, together with basic usage examples, are provided in the R help documentation, available in Annex 3 or by executing the following code in R:

```
help("OCFam")
```

Function arguments

OCFam() function input arguments are:

1. *ped*: a data frame with the following column headings: 'INDIV', 'SIRE', 'DAM', 'FAM', 'SEX', 'BORN', 'EBV', 'N_AS_PARENT_PREV' and 'AVAIL_PARENT'.
2. *sire_count_by_age*: if generations are discrete, this is a scalar specifying the total number of sires required from the youngest age class. If generations are overlapping, this is a vector specifying the number of sires required per age class.
3. *dam_count_by_age*: if generations are discrete, this is a scalar specifying the total number of dams required from the youngest age class. If generations are overlapping, this is a vector specifying the number of dams required per age class.
4. *kinship_constraint*: maximum allowable mean kinship between families in the next age class when maximising mean EBV. If set to NA, no kinship constraint is applied, and average kinship is minimised without consideration of EBV.
5. *iterations*: specifies the number of iterations *OCFam* is to complete. Conceptually, the greater the number of iterations specified, the closer the (integer-feasible) solution will be to the continuous (optimum) solution.
6. *min_prop_fams*: minimum proportion of families in the youngest contributing age class that must contribute at least one parent to the next age class.
7. *max_parents_per_fam*: maximum number of parents allowed from each family.
8. *fast*: logical argument. When TRUE, only candidate parents with the highest estimated breeding values (EBVs) within each family are retained as candidate parents, up to the number defined by *max_parents_per_fam*.

Discrete versus overlapping generations

When adopting overlapping generations (Meuwissen and Sonesson 1998), parents from multiple age classes contribute to the next age class and care must be taken when specifying the following in *OCFam()*:

1. 'BORN' must uniquely and chronologically label individuals in each age class in the *ped* data frame.
2. 'AVAIL_PARENT' must be TRUE for all candidate parents across all contributing age classes in the *ped* data frame.
3. *sire_count_by_age* and *dam_count_by_age* must be vectors, specifying the required number of sires and dams, respectively, from each age class. Age classes are supplied from oldest → youngest.

An example specifying *sire_count_by_age* and *dam_count_by_age* vectors involving two contributing age classes and one immature age class follows.

```
sire_count_by_age <- c(20, 100, 0)
dam_count_by_age  <- c(10, 110, 0)
```

In this case, 20 sires and 10 dams three age classes old, and 100 sire and 110 dams two age classes old will be selected. The most recent age class is not sexually mature and will not contribute parents.

When implementing overlapping generations, the *OCFam()* function enforces fixed numbers of sires and dams within each predefined sex–age class (Kerr *et al.* 2015). This is achieved by constraining the contributions of candidate parents within each sex–age class during the iterative procedure described above. Specifically, once the allocated contribution for a given sex–age class is exhausted, all remaining candidates within that class are assigned a fixed zero contribution, thereby preventing additional contributions from that class and further constraining the solution space in subsequent iterations.

When implementing overlapping generations, *OCFam()* uses between-family kinship matrices for immature age classes, as described in Hamilton (2020), rather than individual-level matrices. For mature age classes, only candidate parents (i.e., individuals flagged as 'AVAIL_PARENT' in the *ped* data frame) are retained during optimal contribution procedures. This approach reduces the dimensionality of kinship matrices and lowers computational burden. To further improve efficiency, set *fast* = TRUE. In this mode, only candidate parents with the highest estimated breeding values (EBVs) within each family are retained, up to the limit defined by *max_parents_per_fam*. Note that when *max_parents_per_fam* is small and fast = *TRUE*, the retained candidates may represent only a small proportion of the living population, which could influence parental selection.

Determining the kinship constraint

Multiple approaches, or combinations of approaches, to determining the appropriate *kinship_constraint* can be conceived, including:

1. Using the *OCFamPrep()* function to inform an appropriate *kinship_constraint*.
2. Adopting a fixed change in *kinship_constraint* for each successive age class – a fixed change of 0.005 (0.5%) per generation corresponds to a realised effective population size based on coancestries ($N_{\mathrm{ec}} = 1/2\Delta\bar{k}$) of 100 (Cervantes *et al.* 2011;

Leroy *et al.* 2013). Note that, in the case of overlapping generations, a generation is not equivalent to an age class.

3. Running *OCFam()* multiple times using different *kinship_constraint* values to evaluate changes in the mean EBV of the selected parents, thereby enabling informed judgement of the most appropriate constraint (i.e. identifying the value at which further increases in *kinship_constraint* result in only minor changes in EBV). Note that specifying 'kinship_constraint = NA' determines the minimum achievable kinship constraint (i.e. the minimum threshold for *kinship_constraint*). See slides 22 and 23 in Annex 1 for an example.

Function outputs

The output of the *OCFam()* function is a R list with the following named components (elements):

1. *fam_contbn*: is a data frame containing total parental contributions by family with the following column headings: 'FAM', 'N_INDIV_TOTAL', 'N_AS_PAST_PARENT', 'N_INDIV'.
2. *parent_contbn*: is a data frame containing details of the selected parents, with the following column headings: 'INDIV', 'SIRE', 'DAM', 'FAM', 'BORN', 'EBV', 'N_AS_PARENT_PREV', 'RANK, AVAIL_PARENT', 'AVAIL_OR_PAST_PARENT', 'FAM_SIRE', 'FAM_DAM', 'BREED', 'MATURE', 'IS_CANDIDATE', 'N_TOTAL_AS_PARENT' and 'ADDED_IN_LAST_ITERATION'.
3. *candidate_parents*: is data frame describing the candidate parents with the following column headings: 'INDIV', 'N_AS_PARENT_PREV', 'LB', 'UB', 'EXCLUDE_MAX_PARENTS_PER_FAM'.
4. *fit_out*: is an R object containing the optiSel constraints and outputs under the final iteration.

Upon completion, the *fit_out* object should be examined to confirm that all specified constraints have been satisfied (e.g. see fit_out$mean; slides 21-23 in Annex 1). The *parent_contbn* data frame represents the finalised set of selected parents to produce the next age class (slide 21 in Annex 1). Based on this list, a mating plan should then be constructed using an appropriate mating design algorithm (e.g., AllocateMate; Hamilton, 2024) to minimize inbreeding in the next age class.

.

References

Caballero A, Toro MA (2000) Interrelations between effective population size and other pedigree tools for the management of conserved populations. *Genetical Research* **75**, 331-343. https://doi.org/10.1017/S0016672399004449

Cervantes I, Goyache F, Molina A, Valera M, Gutierrez JP (2011) Estimation of effective population size from the rate of coancestry in pedigreed populations. *Journal of Animal Breeding and Genetics* **128**, 56-63. https://doi.org/10.1111/j.1439-0388.2010.00881.x

Hamilton MG (2024) 'AllocateMate (Version 2): An R package for mate allocation.' CGIAR Research Program on Fish Agri-Food Systems, Penang, Malaysia. https://doi.org/20.500.12348/6270.

Kerr RJ, McRae TA, Dutkowski GW, Tier B (2015) Managing the rate of increase in average co-ancestry in a rolling front tree breeding strategy. *Journal of Animal Breeding and Genetics* **132**, 109–120. https://doi.org/10.1111/jbg.12157

Leroy G, Mary-Huard T, Verrier E, Danvy S, Charvolin E, Danchin-Burge C (2013) Methods to estimate effective population size using pedigree data: Examples in dog, sheep, cattle and horse. *Genetics Selection Evolution* **45**. https://doi.org/10.1186/1297-9686-45-1

Meuwissen THE (1997) Maximizing the response of selection with a predefined rate of inbreeding. *Journal of Animal Science* **75**, 934-940. https://doi.org/10.2527/1997.754934x

Meuwissen THE, Sonesson AK (1998) Maximizing the response of selection with a predefined rate of inbreeding: Overlapping generations. *Journal of Animal Science* **76**, 2575-2583.

Wellmann R (2019) Optimum contribution selection for animal breeding and conservation: the R package optiSel. *BMC Bioinformatics* **20**, 25. https://doi.org/10.1186/s12859-018-2450-5

Wright S (1922) Coefficients of inbreeding and relationship. *American Naturalist* **56**, 330-338.

Annex 1. Parent selection using OCFam

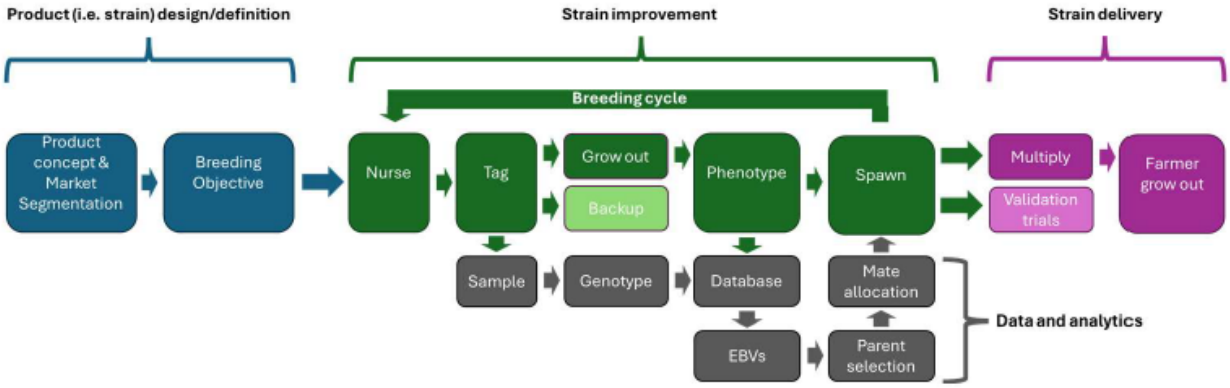
Parent selection

Matthew Hamilton
2025

WorldFish

1

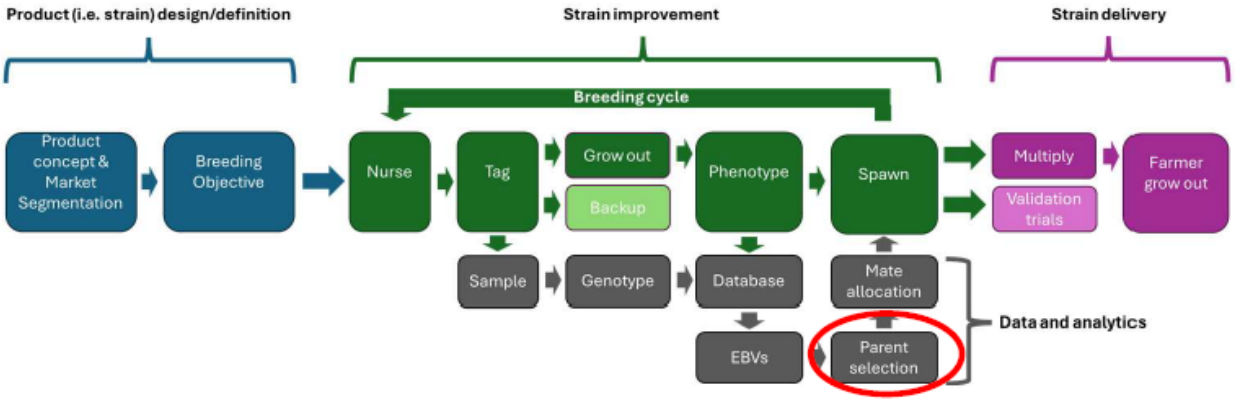
Routine genetic improvement and dissemination



WorldFish

2

Routine genetic improvement and dissemination



3

Parent selection – key considerations

1. Achieve genetic gain
   - Select individuals of superior genetic value (i.e. superior EBVs)

2. Minimise average relatedness
   - Select individuals that are not closely related

4

Why minimise average relatedness?

1. To maintain genetic variation
   - required to achieve genetic gain over the long-term (i.e. many generations)

2. To constrain inbreeding over the long-term

5

Inbreeding

Parent selection to minimise **long-term** inbreeding

Mate allocation to minimise **short-term** inbreeding

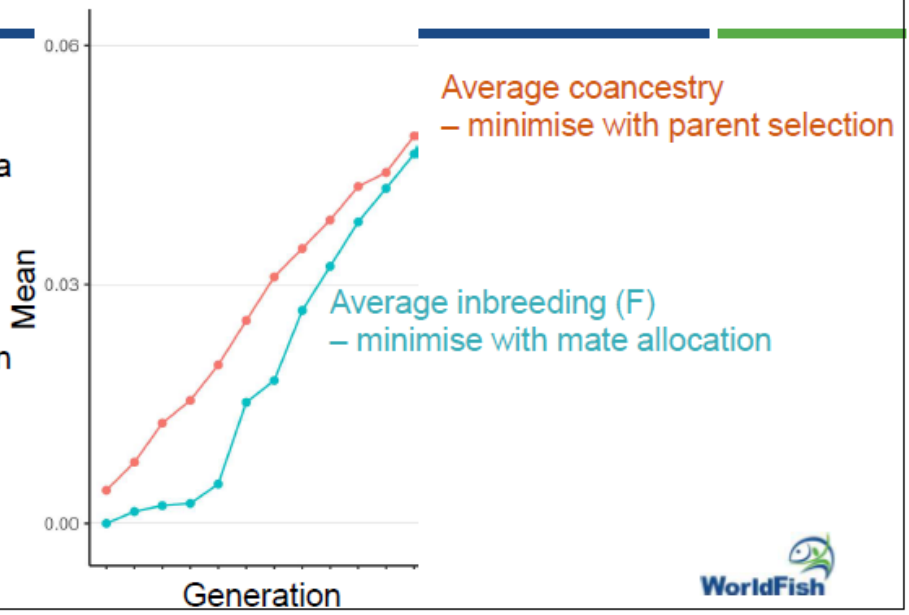
- the next generation only

6

Why minimise average relatedness?

Average coancestry is a measure of average relatedness

The coefficient of coancestry (k_{ij}) between two individuals is equal to half the additive relationship (a_{ij})



WorldFish

7

Trade-offs

We could simply select the fish with the best EBVs

BUT

- these fish are likely to be from a small number of families (i.e. closely related)
- average relatedness (and ultimately inbreeding) will increase very rapidly over generations

We need to constrain average relatedness between selected parents, while achieving genetic gains

- It is a trade off

WorldFish

8

Within family selection

Simple

- For example, identify the best female and best male in each family
- Can be implemented without EBVs

Effective in minimizing relatedness between selected parents

Suboptimal in terms of achieving genetic gains

There are additional arguments for using this approach in tilapia due to its extended spawning period and associated difficulty in estimating accurate and comparable breeding values

WorldFish

9

Optimal contributions selection (OCS)

OCS is a method of parent selection that maximizes genetic gains while controlling average relatedness (i.e. coancestry) and long-term inbreeding.

- Meuwissen THE (1997) Maximizing the response of selection with a predefined rate of inbreeding. Journal of Animal Science 75, 934-940.
- Meuwissen THE, Sonesson AK (1998) Maximizing the response of selection with a predefined rate of inbreeding: Overlapping generations. Journal of Animal Science 76, 2575-2583.
- Wellmann R (2019) Optimum contribution selection for animal breeding and conservation: the R package optiSel. BMC Bioinformatics 20, 25. https://doi.org/10.1186/s12859-018-2450-5
- Hamilton MG (2020) Optimal contribution selection in highly fecund species with overlapping generations. Journal of Heredity 111, 646-651. https://doi.org/10.1093/jhered/esaa051

WorldFish

10

Optimal contributions selection

It has some practical difficulties when applied to highly fecund species like fish

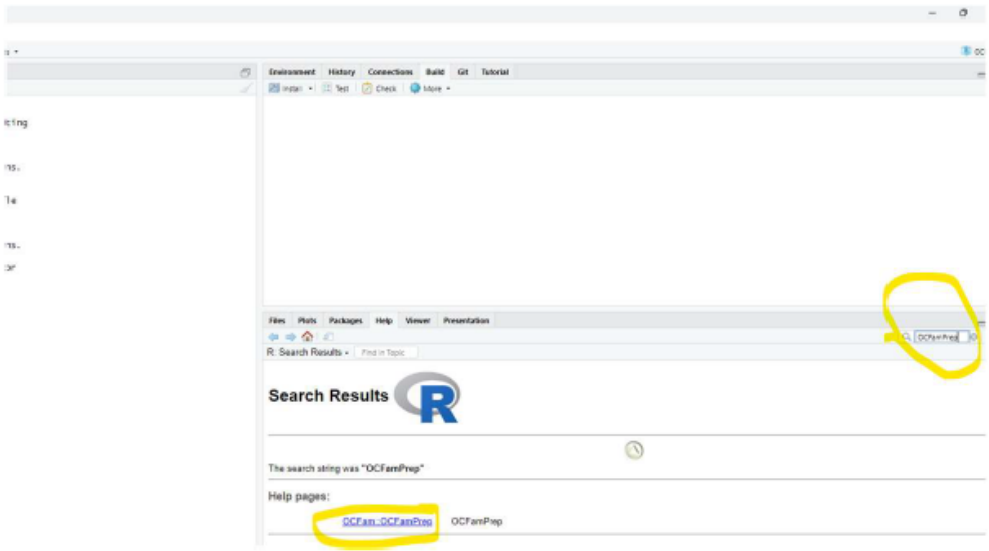
OCFam

- an R package comprised of two functions: OCFamPrep and OCFam
- uses the optiSel R package to solve the optimisation problem https://doi.org/10.1186/s12859-018-2450-5
- addresses rounding issues
- reduces computational complexity by adopting "between-family relationship matrices" https://doi.org/10.1093/jhered/esaa051

WorldFish

11

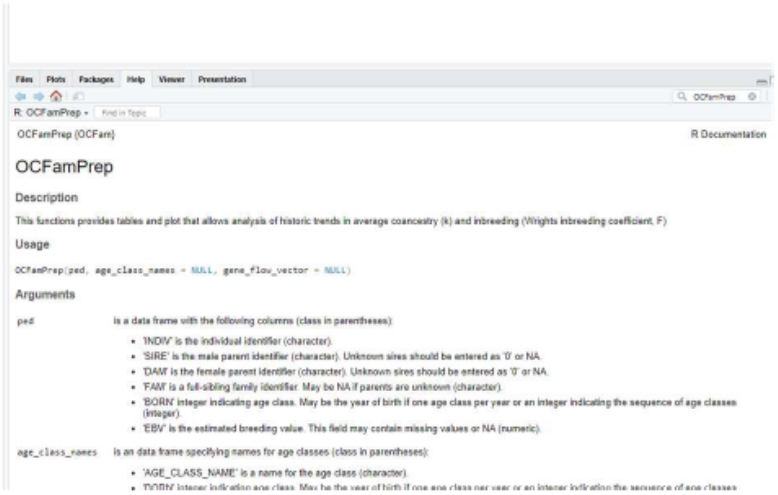
Optimal contributions selection using OCFam
OCFamPrep help



WorldFish

12

Optimal contributions selection using OCFam
OCFamPrep help



Files Plots Packages Help Viewer Presentation

R: OCFamPrep

OCFamPrep {OCFam} R Documentation

OCFamPrep

Description

This functions provides tables and plot that allows analysis of historic trends in average coancestry (k) and inbreeding (Wrights inbreeding coefficient, F)

Usage

```
OCFamPrep(ped, age_class_names = NULL, gene_flow_vector = NULL)
```

Arguments

ped	is a data frame with the following columns (class in parentheses):

- 'INDIV' is the individual identifier (character).
- 'SIRE' is the male parent identifier (character). Unknown sires should be entered as '0' or NA.
- 'DAM' is the female parent identifier (character). Unknown sires should be entered as '0' or NA.
- 'FAM' is a full-sibling family identifier. May be NA if parents are unknown (character).
- 'BORN' integer indicating age class. May be the year of birth if one age class per year or an integer indicating the sequence of age classes (integer).
- 'EBV' is the estimated breeding value. This field may contain missing values or NA (numeric).

age_class_names	is an data frame specifying names for age classes (class in parentheses):

- 'AGE_CLASS_NAME' is a name for the age class (character).
- 'BORN' integer indicating age class. May be the year of birth if one age class per year or an integer indicating the sequence of age classes

WorldFish

13

Optimal contributions selection using OCFam
OCFamPrep inputs

ped.csv

	A	B	C	D	E	F
1	INDIV	SIRE	DAM	FAM	BORN	EBV
149	G0_4525	0	0	NA	0	130
150	G0_4529	0	0	NA	0	130
151	G0_4578	0	0	NA	0	139
152	G1_0013	G0_0126	G0_3487	G0_0126_0	1	141
153	G1_0014	G0_0126	G0_3487	G0_0126_0	1	140
154	G1_0152	G0_0181	G0_1737	G0_0181_0	1	126
155	G1_0153	G0_0181	G0_1737	G0_0181_0	1	122
156	G1_0239	G0_0251	G0_3396	G0_0251_0	1	143
157	G1_0240	G0_0251	G0_3396	G0_0251_0	1	134
158	G1_0241	G0_0251	G0_3396	G0_0251_0	1	132

age_class_name.csv

	A	B
1	AGE_CLASS_NAME	BORN
2	Founders	0
3	Gen_0	1
4	Gen_1	2
5	Gen_2	3
6		

WorldFish

14

Optimal contributions selection using OCFam

OCFamPrep inputs

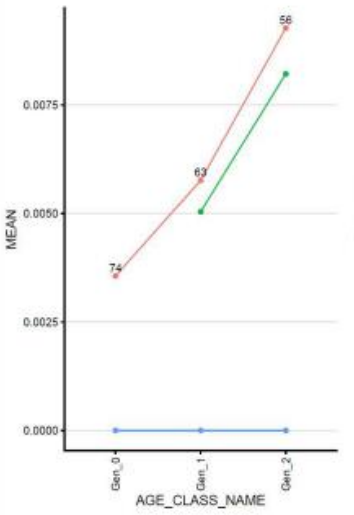
```
> #Retrieve example data
> ped_small <- OCFam::ped_small
> age_class_names <-  data.frame(AGE_CLASS_NAME = c("Founders", "Gen_0", "Gen_1", "Gen_2"), BORN = c(0, 1, 2, 3))
> gene_flow_vector <- c(0.2,0.8)
>
> #Show example data
> tail(ped_small)
      INDIV    SIRE     DAM BORN EBV             FAM N_AS_PARENT_PREV AVAIL_PARENT    SEX
625 G3_2067 G2_1488 G2_1633    3 151 G2_1488_G2_1633                 0          TRUE female
626 G3_2068 G2_1488 G2_1633    3 167 G2_1488_G2_1633                 0          TRUE   male
627 G3_2070 G2_1332 G2_1814    3 133 G2_1332_G2_1814                 0          TRUE female
628 G3_2078 G2_1332 G2_1814    3 120 G2_1332_G2_1814                 0          TRUE   male
629 G3_2097 G2_1332 G2_1814    3 126 G2_1332_G2_1814                 0          TRUE female
630 G3_2098 G2_1332 G2_1814    3 134 G2_1332_G2_1814                 0          TRUE   male
> age_class_names
  AGE_CLASS_NAME BORN
1       Founders    0
2          Gen_0    1
3          Gen_1    2
4          Gen_2    3
> gene_flow_vector
[1] 0.2 0.8
>
> #Run OCFamPrep function
> OCFamPrep_output <- OCFam::OCFamPrep(ped = ped_small,
+                                      age_class_names = age_class_names,
+                                      gene_flow_vector = gene_flow_vector
+ )
```

WorldFish

15

Optimal contributions selection using OCFam

OCFamPrep outputs



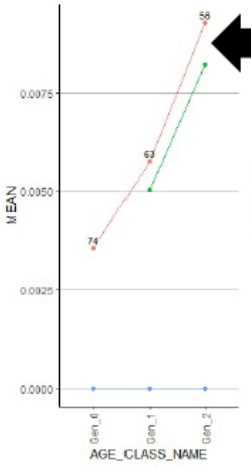

```
> OCFamPrep_output$age_class_means
  AGE_CLASS_NAME BORN N_FAMS INBREEDING  COANCESTRY OVERLAPPING_COANCESTRY
1       Founders    0     NA         NA          NA                     NA
2          Gen_0    1     74          0 0.003560993                     NA
3          Gen_1    2     63          0 0.005763416             0.005039727
4          Gen_2    3     56          0 0.009272361             0.008216997
> OCFamPrep_output$age_class_K_mat
             1           2           3
1 0.003560993 0.003378378 0.003416083
2 0.003378378 0.005763416 0.005823944
3 0.003416083 0.005823944 0.009272361
> OCFamPrep_output$fam_K_matrix[1:5,1:5]
                 G0_0126_G0_3487 G0_0181_G0_1737 G0_0251_G0_3396 G0_0252_G0_1476 G0_0314_G0_2574
G0_0126_G0_3487            0.25            0.00            0.00            0.00            0.00
G0_0181_G0_1737            0.00            0.25            0.00            0.00            0.00
G0_0251_G0_3396            0.00            0.00            0.25            0.00            0.00
G0_0252_G0_1476            0.00            0.00            0.00            0.25            0.00
G0_0314_G0_2574            0.00            0.00            0.00            0.00            0.25
> head(OCFamPrep_output$family_inbreeding)
                          FAM F
G0_0126_G0_3487 G0_0126_G0_3487 0
G0_0181_G0_1737 G0_0181_G0_1737 0
G0_0251_G0_3396 G0_0251_G0_3396 0
G0_0252_G0_1476 G0_0252_G0_1476 0
G0_0314_G0_2574 G0_0314_G0_2574 0
G0_0372_G0_3114 G0_0372_G0_3114 0
```

WorldFish

16

Coancestry predicts future inbreeding

Average coancestry increase per generation is a predictor of the future rate of inbreeding (ΔF)



slope $= \Delta\bar{k}$

$\Delta\bar{k}$, $\Delta\bar{f}$ and ΔC are commonly used to represent the **average coancestry increase per generation**.

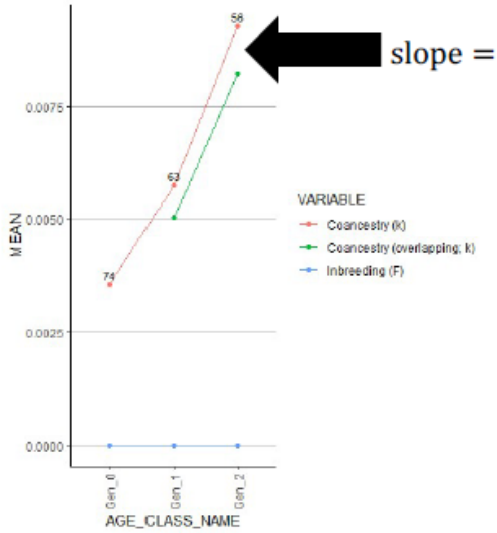
"the increase in coancestry equals the future increase in inbreeding" (Meuwissen, 1997)
https://doi.org/10.2527/1997.754934x

"In a single population randomly mated, inbreeding (F) and coancestry (f) coefficients will increase with generations such that $\bar{f}_t = \bar{F}_{t+1}$ (or $\bar{F}_{t+2}$ without self fertilization). In non-random mating populations there may be a larger delay between $\bar{f}$ and $\bar{F}$." (Caballero & Toro, 2000)
https://doi.org/10.1017/S0016672399004449

WorldFish

17

Effective population size based on coancestries (N_{ec})



slope $= \Delta\bar{k}$

$$N_{\mathrm{ec}} = \frac{1}{2\Delta\bar{k}}$$

Cervantes et al. (2011)
https://doi.org/10.1111/j.1439-0388.2010.00881.x

Leroy et al. (2013)
https://doi.org/10.1186/1297-9686-45-1

WorldFish

18

Optimal contributions selection using OCFam
OCFam inputs

ped.csv (ped_small from OCFam help in R)

	A	B	C	D	E	F	G	H	I
1	INDIV	SIRE	DAM	BORN	EBV	FAM	N_AS_PARENT_PREV	AVAIL_PARENT	SEX
2	G0_0033	0	0	0	150	NA	0	FALSE	female
3	G0_0116	0	0	0	151	NA	0	FALSE	female
4	G0_0126	0	0	0	150	NA	0	FALSE	male
5	G0_0131	0	0	0	144	NA	0	FALSE	male
189	G3_2067	G2_1488	G2_1633	3	151	G2_1488_G2_1633	0	TRUE	female
190	G3_2068	G2_1488	G2_1633	3	167	G2_1488_G2_1633	0	TRUE	male
191	G3_2070	G2_1332	G2_1814	3	133	G2_1332_G2_1814	0	TRUE	female
192	G3_2078	G2_1332	G2_1814	3	120	G2_1332_G2_1814	0	TRUE	male
193	G3_2097	G2_1332	G2_1814	3	126	G2_1332_G2_1814	0	TRUE	female
194	G3_2098	G2_1332	G2_1814	3	134	G2_1332_G2_1814	0	TRUE	male

WorldFish

19

Optimal contributions selection using OCFam
OCFam inputs

```
#Run OCFam function
OCFam_out_small <- OCFam::OCFam(ped = ped_small,
                                sire_count_by_age = 60,
                                dam_count_by_age = 60,
                                kinship_constraint = NA, #This is a small example.  Cannot constrain kinship_constraint.
                                iterations = 10,
                                min_prop_fams = 0.9,
                                max_parents_per_fam = 4,
                                fast = TRUE
)

OCFam_out_small$fit_out$mean
head(OCFam_out_small$fam_contbn)
head(OCFam_out_small$parent_contbn)
```

WorldFish

20

Optimal contributions selection using OCFam

OCFam outputs

```
> OCFam_out_small$fit_out$mean
     EBV N_AS_PARENT_PREV  RANK  equiGen       fPED
1 137.65                0 261.4 2.941667 0.01109201
> head(OCFam_out_small$fam_contbn)
            FAM N_INDIV_TOTAL N_AS_PAST_PARENT N_INDIV
1 G2_0043_G2_0949             2                0       2
2 G2_0070_G2_1549             3                0       3
3 G2_0132_G2_0788             4                0       4
4 G2_0185_G2_1148             1                0       1
5 G2_0194_G2_0579             2                0       2
6 G2_0264_G2_1741             3                0       3
> head(OCFam_out_small$parent_contbn)
    INDIV     SIRE      DAM BORN EBV             FAM N_AS_PARENT_PREV AVAIL_PARENT    SEX RANK AVAIL_OR_PAST_PARENT        FAM_SIRE
1 G3_0007 G2_0355 G2_0065     3 152 G2_0355_G2_0065                 0         TRUE   male   19                 TRUE G1_0698_G1_2519
2 G3_0035 G2_0355 G2_0065     3 145 G2_0355_G2_0065                 0         TRUE female   72                 TRUE G1_0698_G1_2519
3 G3_0058 G2_0070 G2_1549     3 138 G2_0070_G2_1549                 0         TRUE   male  207                 TRUE G0_0131_G0_2502
4 G3_0075 G2_0070 G2_1549     3 147 G2_0070_G2_1549                 0         TRUE   male   54                 TRUE G0_0131_G0_2502
5 G3_0088 G2_0070 G2_1549     3 130 G2_0070_G2_1549                 0         TRUE female  445                 TRUE G0_0131_G0_2502
6 G3_0124 G2_0043 G2_0949     3 122 G2_0043_G2_0949                 0         TRUE female  598                 TRUE G1_3486_G1_0699
         FAM_DAM         BREED N_TOTAL_AS_PARENT ADDED_IN_LAST_ITERATION
1 G0_0131_G0_2502 Breed_ignored                1                   FALSE
2 G0_0131_G0_2502 Breed_ignored                1                   FALSE
3 G1_4177_G1_1223 Breed_ignored                1                   FALSE
4 G1_4177_G1_1223 Breed_ignored                1                   FALSE
5 G1_4177_G1_1223 Breed_ignored                1                   FALSE
6 G1_5054_G1_4059 Breed_ignored                1                    TRUE
```

Minimum $\bar{k} = 0.0111$

21

Optimal contributions selection using OCFam

OCFam ($\bar{k}$ is constrained)

```
OCFam_out_big <- OCFam::OCFam(ped = ped_big,
                              sire_count_by_age = c(20, 100),
                              dam_count_by_age = c(10, 110),
                              kinship_constraint = 0.014,
```

ped_big from OCFam help in R

$\bar{k} \leq 0.014$ (i.e. constrained)

EBV is maximised while $\bar{k}$ is constrained

Run 1 ($\bar{k}$ minimised)

```
> OCFam_out_big$fit_out$mean
     EBV N_AS_PARENT_PREV     RANK  equiGen       fPED
1 384.6668                0 1779.995 5.882857 0.01166759
```

Run 2

```
> OCFam_out_big$fit_out$mean
     EBV N_AS_PARENT_PREV     RANK  equiGen       fPED
1 433.3779                0 1570.641 5.882662 0.0120041
```

Run 3

```
> OCFam_out_big$fit_out$mean
     EBV N_AS_PARENT_PREV     RANK  equiGen       fPED
1 494.5993                0 1266.432 5.882786 0.01301556
```

Run 4

```
> OCFam_out_big$fit_out$mean
     EBV N_AS_PARENT_PREV     RANK  equiGen       fPED
1 512.1629                0 1170.37 5.882829 0.01400649
```

Run 5

```
> OCFam_out_big$fit_out$mean
     EBV N_AS_PARENT_PREV     RANK  equiGen       fPED
1 515.2568                0 1154.137 5.882857 0.01471008
```

22

Optimal contributions selection using OCFam

OCFam ($\bar{k}$ is constrained)

```
OCFam_out_big <- OCFam::OCFam(ped = ped_big,
                              sire_count_by_age = c(20, 100),
                              dam_count_by_age = c(10, 110),
                              kinship_constraint = 0.014,
```

ped_big from OCFam help in R

$\bar{k} \leq 0.014$ (i.e. constrained)

EBV is maximised while $\bar{k}$ is constrained

	ΔEBV		$\Delta \bar{k}$
Run 1 ($\bar{k}$ minimised)		`> OCFam_out_big$fit_out$mean` EBV N_AS_PARENT_PREV RANK equiGen fPED: 1 384.6668 0 1779.995 5.882857 0.01166759	
	48.7		0.0003
Run 2		`> OCFam_out_big$fit_out$mean` EBV N_AS_PARENT_PREV RANK equiGen fPED: 1 433.3779 0 1570.641 5.882662 0.0120041	
	61.2		0.001
Run 3		`> OCFam_out_big$fit_out$mean` EBV N_AS_PARENT_PREV RANK equiGen fPED: 1 494.5993 0 1266.432 5.882786 0.01301556	
	17.6		0.001
Run 4		`> OCFam_out_big$fit_out$mean` EBV N_AS_PARENT_PREV RANK equiGen fPED: 1 512.1629 0 1170.37 5.882829 0.01400649	
	3.1		0.001
Run 5		`> OCFam_out_big$fit_out$mean` EBV N_AS_PARENT_PREV RANK equiGen fPED: 1 515.2568 0 1154.137 5.882857 0.01471008	

Change in EBV declines

Change in $\bar{k}$ is fixed

23

Thank You

WorldFish

Annex 2. OCFamPrep function R help

OCFamPrep

Description
This function provides tables and a plot that allows analysis of historic trends in average coancestry (k) and inbreeding (Wrights inbreeding coefficient, F) across age classes.

Usage
OCFamPrep(ped, age_class_names = NULL, gene_flow_vector = NULL)

Arguments

ped	is a data frame with the following columns (class in parentheses): • 'INDIV' is the individual identifier (character). • 'SIRE' is the male parent identifier (character). Unknown sires should be entered as '0' or NA. • 'DAM' is the female parent identifier (character). Unknown sires should be entered as '0' or NA. • 'FAM' is a full-sibling family identifier. May be NA if parents are unknown (character). • 'BORN' integer indicating age class. May be the year of birth if one age class per year or an integer indicating the sequence of age classes (integer). • 'EBV' is the estimated breeding value. This field may contain missing values or NA (numeric).
age_class_names	is a data frame specifying names for age classes (class in parentheses): • 'AGE_CLASS_NAME' is a name for the age class (character). • 'BORN' integer indicating age class. May be the year of birth if one age class per year or an integer indicating the sequence of age classes (integer).
gene_flow_vector	is a applicable to overlapping generations (if NA discrete generations is assumed). It is vector representing the assumed parental contributions by age class to the next age class. For example, gene_flow_vector = c(0.2, 0.8, 0, 0) assumes 20 percent of parents are four age classes old and 80 percent are three age classes old.

Value
'plot' is plot of trends in coancestry and inbreeding coefficients
'age_class_means' is a dataframe listing inbreeding coefficients for individuals within families.

- 'BORN' is age class (integer).
- 'INBREEDING' is the average Wright's inbreeding coefficient of families in the age class (numeric).
- 'COANCESTRY' is the average coancestry between families in the age class (numeric).

- 'OVERLAPPING_COANCESTRY' is the average coancestry between families weighted by the age class contributions specified in the gene_flow_vector (numeric).

'age_class_K_mat' is a matrix containing average coancestries within age classes on the diagonals and between age classes on the off-diagonals. Row and column names are the FAM identifiers from 'ped'.
'fam_K_matrix' is a data frame containing the between family coancestry matrix (half of the Between-Family Relationship Matrix described in Hamilton (2020) 'Optimal Contribution Selection in Highly Fecund Species With Overlapping Generations'). Row and column names are the FAM identifiers from 'ped'.
'family_inbreeding' is a dataframe listing inbreeding coefficients for individuals within families.'

- 'FAM' is the family identifier (character).
- 'F' is the Wright's inbreeding coefficient for individuals in the family (numeric).

'ped' is a data frame with the following columns (class in parentheses):

- 'INDIV' is the individual identifier (character).
- 'SIRE' is the male parent identifier (character).
- 'DAM' is the female parent identifier (character).
- 'FAM' is a full-sibling family identifier (character).
- 'BORN' integer indicating age class. May be the year of birth if one age class per year or an integer indicating the sequence of age classes (integer).
- 'EBV' is the estimated breeding value (numeric).
- 'FAM_SIRE' is a full-sibling family identifier of the SIRE (character).
- 'EBV_SIRE' is the estimated breeding value of the SIRE (numeric).
- 'FAM_DAM' is a full-sibling family identifier of the DAM (character).
- 'EBV_DAM' is the estimated breeding value of the DAM (numeric).
- 'EBV_FAM' is the mean EBV of the FAM's SIRE and DAM

'family_ped' is a data frame with the following columns (class in parentheses):

- 'FAM' is a full-sibling family identifier (character).
- 'SIRE' is the male parent identifier (character).
- 'DAM' is the female parent identifier (character).
- 'BORN' integer indicating age class. May be the year of birth if one age class per year or an integer indicating the sequence of age classes (integer).
- 'FAM_SIRE' is a full-sibling family identifier of the SIRE (character).
- 'EBV_SIRE' is the estimated breeding value of the SIRE (numeric).
- 'FAM_DAM' is a full-sibling family identifier of the DAM (character).
- 'EBV_DAM' is the estimated breeding value of the DAM (numeric).
- 'EBV_FAM' is the mean EBV of the FAM's SIRE and DAM

'gene_flow_vector' is a applicable to overlapping generations (if NA discrete generations is assumed). It is vector representing the assumed parental contributions by age class to the next age class. For example, gene_flow_vector = c(0.2, 0.8, 0, 0) assumes 20 percent of parents are four age classes old and 80 percent are three age classes old.
'r' is a weight vector of age classes; r[i] denotes the long-term contribution of age class i (until lifetime is reached). See Meuwissen, T. H. E. and A. K. Sonesson. 1998. Maximizing the response of selection with a predefined rate of inbreeding: overlapping generations.
'gen_interval' is the generation interval.

Examples

```
#Retrieve example data
ped_small <- OCFam::ped_small
age_class_names <-  data.frame(AGE_CLASS_NAME = c("Founders",
"Gen_0", "Gen_1", "Gen_2"), BORN = c(0, 1, 2, 3))
gene_flow_vector <- c(0.2,0.8)

#Show example data
tail(ped_small)
age_class_names
gene_flow_vector

#Run OCFamPrep function
OCFamPrep_output <- OCFam::OCFamPrep(ped = ped_small,
                              age_class_names = age_class_names,
                              gene_flow_vector = gene_flow_vector
)

OCFamPrep_output$age_class_means
OCFamPrep_output$age_class_K_mat
OCFamPrep_output$fam_K_matrix[1:5,1:5]
head(OCFamPrep_output$family_inbreeding)
```

Annex 3. OCFamPrep function R help

OCFam

Description

This function addresses the rounding issue associated with standard optimal contributions, particularly in highly fecund species in which relatively few families are generated (or parents are used).

Usage

```
OCFam(
  ped,
  sire_count_by_age,
  dam_count_by_age,
  kinship_constraint = NA,
  iterations = 10,
  min_prop_fams = 0,
  max_parents_per_fam = 10000,
  fast = TRUE
)
```

Arguments

ped	is a data frame with the following columns (class in parentheses): • 'INDIV' is the individual identifier (character). • 'SIRE' is the male parent identifier (character). • 'DAM' is the female parent identifier (character). • 'FAM' is a full-sibling family identifier (character). • 'SEX' is the sex of the individual. "male" (or 1 = male), "female" (or 2 = female), NA if unknown. If one animal is of unknown sex, sex will be ignored (integer or character). • 'BORN' integer indicating age class. May be the year of birth if one age class per year or an integer indicating the sequence of age classes (integer). • 'EBV' is the estimated breeding value (numeric). • 'N_AS_PARENT_PREV' is the number of families in the next age class previously contributed to (integer). • 'AVAIL_PARENT' is TRUE if the individual is candidate parent of next age class (logical).
sire_count_by_age	is scalar (discrete generations) or a vector (overlapping generations) representing required sires by age class. For example (overlapping generations), sire_count_by_age = c(20, 80, 0, 0) - oldest parental age class to youngest age class.
dam_count_by_age	is scalar (discrete generations) or a vector (overlapping generations) representing required dams by age class. For

	example (overlapping generations), dam_count_by_age = c(20, 80, 0, 0) - oldest parental age class to youngest age class.
kinship_constraint	is the maximum value of the mean kinship among families in the next age class, while mean EBV is maximised (max.EBV). If NA there is no constraint placed on kinship and the average kinship is minimised while EBV is not considered (min.fPED) (numeric between 0 and 1; default = NA)
iterations	specifies the number of iterations. The greater the number of iterations specified, the closer the solution will be to the optimum but the longer it will take to run (positive integer; default = 10)
min_prop_fams	is the proportion of families to be retained (i.e. to contribute at least one parent) from the oldest age class with parental candidates (numeric between 0 and 1; default = 0)
max_parents_per_fam	is the maximum number of parents to contribute from each family (integer; default = 10000)
fast	when TRUE, only candidate parents with the highest estimated breeding values (EBVs) within each family are retained as candidate parents, up to the limit defined by max_parents_per_fam (logical; default = TRUE)

Value

'fam_contbn' is a data frame containing details of contributions by family:

- 'FAM' is a full-sibling family identifier (character).
- 'N_INDIV_TOTAL' is the total count of parental contributions to the next age class (integer)
- N_AS_PAST_PARENT is the count of past parental contributions to the next age class (integer).
- N_INDIV is the count of parental contributions to the next age class from the current OCFam run. Equals N_INDIV_TOTAL minus N_AS_PAST_PARENT (integer)

'parent_contbn' is a data frame containing details of contributions by individual:

- 'INDIV' is the individual identifier (character).
- 'SIRE' is the male parent identifier (character).
- 'DAM' is the female parent identifier (character).
- 'FAM' is a full-sibling family identifier (character).
- 'BORN' integer indicating age class. May be the year of birth if one age class per year or an integer indicating the sequence of age classes (integer).
- 'EBV' is the estimated breeding value (numeric).
- 'N_AS_PARENT_PREV' is the number of families in the next age class previously contributed to (integer).
- 'AVAIL_PARENT' is TRUE if the individual is candidate parent of next age class (logical).
- 'RANK' individual rank based on EBV
- 'AVAIL_OR_PAST_PARENT' is TRUE if AVAIL_PARENT is TRUE or the individual has previously been used as a parent in the next age class.
- 'FAM_SIRE' is a full-sibling family identifier of the SIRE (character).
- 'FAM_DAM' is a full-sibling family identifier of the DAM (character).
- 'BREED' is the species.
- 'MATURE' is sexually mature

- 'IS_CANDIDATE' was considered a candidate parent in OCFam
- 'N_TOTAL_AS_PARENT' is the total count of parental contributions to the next age class
- 'ADDED_IN_LAST_ITERATION' individual was addes in the last iteration of OCFam. Would be the first individuals to be removed in necessary.

'candidate_parents' is a data frame containing details of candidate parents:

- 'INDIV' is the individual identifier (character).
- 'N_AS_PARENT_PREV' is the number of families contributed to in the next age class prior to running optimal contributions (integer).
- 'LB' is the low bound of possilbe contribution to the next age class
- 'UB' is the upper bound of possilbe contribution to the next age class
- EXCLUDE_MAX_PARENTS_PER_FAM' If TRUE individual is excluded on the basis that there are better parents available from the family to meet the max_parents_per_fam constraint.

'fit_out' is a vector containing details of the constraints applied in the implementation of optiSel.

Examples

```
#Retrieve small example data
ped_small <- OCFam::ped_small
tail(ped_small)

#Run OCFam function
OCFam_out_small <- OCFam::OCFam(ped = ped_small,
                                sire_count_by_age = 60,
                                dam_count_by_age = 60,
                                kinship_constraint = NA, #This is a
small example.  Cannot constrain kinship_constraint.
                                iterations = 10,
                                min_prop_fams = 0.9,
                                max_parents_per_fam = 4,
                                fast = TRUE
)

OCFam_out_small$fit_out$mean
head(OCFam_out_small$fam_contbn)
head(OCFam_out_small$parent_contbn)

#Retrieve larger example (sex unknown)
ped_big <- OCFam::ped_big
tail(ped_big)

#Run OCFam function
OCFam_out_big <- OCFam::OCFam(ped = ped_big,
                                sire_count_by_age = c(20, 100),
                                dam_count_by_age = c(10, 110),
                                kinship_constraint = 0.014,
                                iterations = 10,
```

```
                                min_prop_fams = 0,
                                max_parents_per_fam = 4,
                                fast = TRUE
)

OCFam_out_big$fit_out$mean
head(OCFam_out_big$fam_contbn)
head(OCFam_out_big$parent_contbn)
```