AlleleRetain

A model for simulating allele retention, demography, and inbreeding accumulation

User Guide

AlleleRetain version 2.0

11 January 2018

Emily L. Weiser

Work was completed at: Department of Zoology, University of Otago, Dunedin, New Zealand
Present affiliation: U.S. Geological Survey, La Crosse, WI USA

with many thanks to
I. Jamieson, C. Grueber, M. Efford, J. Scrimgeour, M. Reynolds, and G. Humphries
for helpful discussions on model development and function.

Funding for this work was generously provided by
New Zealand Department of Conservation, Landcare Research, Marsden Fund, University of Otago, and Allan Wilson Centre for Molecular Ecology and Evolution.

DISCLAIMER: Although this model has been tested extensively and seems to perform appropriately, I offer no guarantee that it works properly, as expected, or exactly as described. My colleagues and I are not responsible for any errors or problems resulting from this program or its documentation. Please use caution when using the results for your own work, especially when there is some uncertainty in your input values.

Table of Contents

Introduction

AlleleRetain is an individual-based model implemented in R (R Development Core Team 2011) to simulate retention of selectively neutral alleles, demographics, and inbreeding accumulation in bottlenecked or newly established populations of animals with overlapping generations. **AlleleRetain** was originally intended to be an expansion of **mohuasim** (Efford 2010; see also Tracy et al. 2011), but ended up being dramatically expanded and more flexible.

The user of **AlleleRetain** describes the individuals used to establish a new population (number, age, sex, release period). Released individuals are assumed to be unrelated and are randomly assigned genotypes (0-2 copies of a hypothetical neutral allele of the specified initial frequency). Released individuals and their offspring mature, breed, and survive according to user-specified options. Immigration can be simulated at regular user-specified intervals; emigration can be simulated by reducing the survival rate of the appropriate age class, assuming emigrants and their descendants do not return to the simulated population. The simulation is run for a specified period of years and number of replicates.

The proportion of replicates in which the allele is retained to each year of the simulation is interpreted as the probability of retaining the allele in the population over that period of time. This is equivalent to the proportion of neutral alleles occurring at the given frequency in the source population predicted to be retained in the new population.

The summary functions included in the package output a census of the simulated population (adults, nonbreeders, breeding pairs, starting individuals, and migrants) each year, averaged across replicates; a summary of information (e.g. probability of breeding) for the individuals of different origin; and average simulated inbreeding coefficients (F).

Questions about the program or its use can be directed to AlleleRetainR@gmail.com; I will try to respond to as many questions as my schedule allows. (Please carefully read this manual first.)

Citation

If you use **AlleleRetain** in your work, please cite the following paper that describes the package:

> Weiser, E.L., Grueber, C. E., and I. G. Jamieson. 2012. **AlleleRetain**: A program to assess management options for conserving allelic diversity in small, isolated populations. Molecular Ecology Resources 12:1161-1167.

You might also be interested in these detailed examples using **AlleleRetain**:

> Weiser, E.L., Grueber, C. E., and I. G. Jamieson. 2013. Simulating retention of rare alleles in small populations: assessing management options for species with different life histories. Conservation Biology 27:335-344.

Changes for version 2.0

AlleleRetain version 2.0 was created on 21 Dec 2017 and replaces version 1.3.3 and all previous versions. Most of the changes involved adding additional parameters; running old code should still work, because the default values of the new parameters will not affect how your code runs. The exception is that the argument `exactSR` must now be called `exactSSR`, and sex ratios for supplementals and migrants must be specified separately as indicated below.

- Added option for harvesting individuals using new arguments `harvN`, `harvAge`, `harvyrs`.
- Allowed `inisurv` and sex ratios to vary among starters, supplementals, and migrants. `inisurv` is now specified as a vector of three values, while sex ratios are specified using `startSR` and `exactSSR` (starters), `addSR` and `exactASR` (supplementals), and `migrSR` and `exactMSR` (migrants).
- Added new summary functions to output information about reproductive success of individuals (`LRS.summary`) and age of reproductive individuals (`agerepro.summary`). These summaries might be of interest in situations such as a polygynous mating system with unequal male reproductive success.
- Changed fecundity to draw from a binomial distribution limited to the indicated maximum, rather than an artificially truncated Poisson. The new approach is cleaner but should not substantially change results.
- Fixed bug so that if no starters/supplementals survive initially, the simulation continues.

Definition of Terms Used in this Manual

Source population	The population from which individuals used to establish the new (modeled) population are taken. `q0` and `sourceN` (pg. 6) refer to this population; all other input and output parameters refer to the new (modeled) population.
Starters	Individuals initially released to establish the population
Supplementals	Individuals released to top up the newly established population (usually in the first few years after establishment)
Migrants (immigrants)	Individuals released at regular intervals to supplement the population
Nonbreeders	Individuals that have not yet recruited to breeding vacancies in the simulated population; includes subadults, helpers, and unpaired adults.
SD	Standard deviation
SE	Standard error

Input and output for R (arguments, objects) and functions are in `Courier New` font.
R package names are **bolded**.

Installation and Use

AlleleRetain and its supporting documentation can be freely downloaded from the Comprehensive R Archive Network (CRAN: cran.r-project.org) or from its website: https://sites.google.com/site/alleleretain/. You can access CRAN from R by choosing "Install package…" from the "Packages" menu; or you can download the zip file from CRAN or from the **AlleleRetain** website, then open R and choose "Install package from local zip file" in the "Packages" menu. Be sure to load the package (with the command `library(AlleleRetain)`) each time you want to use it.

Package **pedigree** is needed for one of the summary functions; it and its dependencies (**Matrix, lattice, HaploSim, reshape**) can be downloaded from CRAN. If you install **AlleleRetain** via CRAN directly from your R console, **pedigree** (and its dependencies) will install automatically.

Before running your simulations, be sure to gather all of the input values needed (see "Input to `aRetain`," pg. 6). This is a daunting list of parameters, but enables flexibility to model a variety of species. Some of these parameters strongly influence allele retention, so think carefully about the values you use.

Because it is an individual-based simulation, **AlleleRetain** can be time-consuming and memory-intensive to run. A 32-bit system may not be sufficient, but a 64-bit system and a modern processor should be able to handle most simulations. Run your simulation first with <100 replicates (which could take several minutes) to ensure there are no errors and the demographic output looks reasonable before running more lengthy simulations (which could take > 1 hr, perhaps several hours, depending on the size of your population, number of years, and number of replicates). A message will be displayed at regular intervals to enable you to gauge the progress of the simulation. If your system does not have enough memory for the simulations you are trying to run, you will get an error message from R to that effect; see "Troubleshooting," pg. 14.

How many replicates to run? This will depend on your purpose in running the model. To assess demographics, inbreeding coefficients, or individual data (e.g. probability of breeding for locals vs. migrants), 50-100 replicates may be sufficient. For assessing the probability of retaining a rare allele, 1000 replicates will usually give a reasonable confidence interval. More replicates will be decreasingly effective in reducing the confidence interval.

How many years to simulate? For management purposes, the useful length of the simulation will depend on how far ahead you can plan for the population. If feasible, think in terms of the generation interval of your species (which can be derived from output from **AlleleRetain**; see `indiv.summary`, pg. 12), and simulate at least 10 generations into the future.

The first thing the model does is to check your input values to make sure they follow certain rules. If your values violate one of the rules, the simulation will stop and an error message will be displayed. See "Troubleshooting," pg. 14.

Input to aRetain

aRetain is the main function in **AlleleRetain**. See Fig. 1 for an overview of the order in which events occur, and the user-specified parameters called during each event. Standard R conventions are required for input; for example, text input must be in quotes ("juvenile"), TRUE and FALSE must be in all caps, and vectors are provided as c(1:5) or c(2,4,6,8). Be very cautious about using the defaults, as these are unlikely to be appropriate for your species.

Source population

q0	Frequency of rare allele in the source population (range 0-1); defaults to 0.05.
sourceN	Size of source population; must be > startN; defaults to Inf (infinite).

Translocated individuals

startN	Number of starters (or size of bottleneck); not all will become genetic founders. Minimum 2.
startAge	Age class ("juvenile", "young adult", or "adult") of starters, supplementals, and migrants (see Model Flow 3.d.); defaults to "juvenile".
startSR	Sex ratio (proportion male) of starters, supplementals, and migrants; defaults to 0.5 (must be between 0 and 1).
exactSSR	Whether startSR gives the exact sex ratio of individuals released (TRUE) or sexes are assigned randomly based on the probability given by startSR (FALSE); defaults to FALSE.
inisurv	Initial survival rate, as a proportion (range 0-1), of individuals released. Given as a vector, where the first value is for starters, the second is for additional releases, and third is for migrants. Annual mortality applies after this value is used. Defaults to 1 for all three groups.
addN	Vector of numbers of individuals (supplementals) to release in years soon after population establishment, e.g. c(10, 15); defaults to 0.
addyrs	Vector of years in which to release supplemental, e.g. c(1, 2). Each year corresponds to the number of individuals in the same position in the addN list. Defaults to 0.
addSR	Sex ratio (proportion male) of supplementals; defaults to 0.5 (must be between 0 and 1). This can be either a single value, or a vector, with each element in the vector corresponding to each instance of supplementation (must be the same length as addN).
exactASR	Whether addSR gives the exact sex ratio of individuals released (TRUE) or sexes are assigned randomly based on the probability given by addSR (FALSE); defaults to FALSE. This can be either a single value, or a vector, with each element in the vector corresponding to each instance of supplementation (must be the same length as addN).

migrN	Number of migrants to add (must be a whole number) at each interval given by migrfreq; defaults to 0.
migrfreq	Interval (number of years) at which to add migrN migrants; must be between 1 and nyears (below); defaults to 1.
migrSR	Sex ratio (proportion male) of supplementals; defaults to 0.5 (must be between 0 and 1).
exactMSR	Whether addSR gives the exact sex ratio of individuals released (TRUE) or sexes are assigned randomly based on the probability given by migrSR (FALSE); defaults to FALSE.
mpriority	TRUE or FALSE: whether migrants are given priority over locally produced offspring to recruit into any available breeding vacancies; defaults to FALSE.
removeL	TRUE or FALSE: whether to remove the corresponding number of locally produced adults to make room for migrants in the population; only necessary if retainBreeders = "both"/"female"/"male"; will only come into play when population is at K. Defaults to FALSE.
harvN	Number to be removed in each harvest year
harvAge	Age of individuals to be harvested (as for startAge). If not enough individuals of this age are available, the harvest quota (harvN) will not be filled.
harvyrs	Vector of years in which harvest occurs

Characteristics of the established population

K	Carrying capacity (population ceiling); defaults to 100.
Klag	Number of years for which population is held at or below initial size (breeding still occurs); indicates a prolonged bottleneck. Defaults to 0.
KAdults	TRUE (K = number of adults) or FALSE (K = total individuals, including subadults, nonbreeders, and helpers). Defaults to FALSE.
reprolag	Number of years after establishment in which no reproduction occurs. Defaults to 0.

Life history traits of the simulated species

mature	Average age (in years) at sexual maturity (first breeding); defaults to 1.
matingSys	Mating system: "monogamy", "polygyny", or "polygynandry". In any case, pairs are formed and breed. With polygyny, each male can be part of more than one pair. With polygynandry, females mate and reproduce multiple times each year. To model a polyandrous system, set to "polygyny" and then input female values for the "male" parameters (and male values for the "female" parameters) in the model. Defaults to "monogamy".
matingLength	"seasonal" or "lifelong". Determines whether individuals retain the same mate from year to year or divorce. Note that if set to "lifelong" with

polygyny/polygynandry, males will not obtain any new mates until ALL of their previous mates have died (probably not realistic for most species).

`meanMLRS` Mean lifetime reproductive success (LRS), in terms of number of matings that produce young (NOT number of offspring) a male gets over his lifetime. This is a population average for all males, including those that never reproduce, and may be a fraction. Each male is assigned an individual average from a gamma distribution with this mean and SD given by `sdMLRS` (see `newinfo`, pg. 21). The SD:mean ratio is more important than the magnitude of the mean. The individual mean indicates the male's "quality" and will be used to assess his chance of mating, relative to other males present, each year (does not translate directly into actual LRS experienced by that male). Not used if `matingSys = "monogamy"`. Defaults to 1.

`sdMLRS` Among-male standard deviation in LRS. Used with `meanMLRS` as described above. Not used if `matingSys = "monogamy"`. Defaults to 0 (all males have the same chance of breeding each year).

`reproAgeM` List of ages at which males can breed. If males breed at all ages, use 0:maximum possible lifespan. The maximum cannot be set to `Inf`, but you can use a really high number if unsure of lifespan. Defaults to c(1:200).

`AgeOnMLRS` Expression describing the proportion of LRS achieved by a male at a particular age (for ages contained within `reproAgeM`). The user specifies the form of this expression; it must include `age` (the individual's current age) and no other undefined variables. Example: `"-5.4 + 1.5*age - 0.08*age^2"` describes a parabolic relationship between age and mating success (proportion of LRS achieved at each age). If there is no effect of age, use the default value of `"age/age"` (equals 1 so all ages will be assigned the same average, given by `meanMLRS`). If a given age is not included in `reproAgeM`, reproductive output at that age will be set to 0 regardless of the value calculated by this equation.

`nMatings` Average number of matings per female each year. Only used when `matingSys = "polygynandry"`. Each female mates and breeds the corresponding number of times each year. Repeat matings with the same male are not prohibited and may occur by chance. Defaults to 1; must be a whole number.

`retainBreeders` Should established breeders retain their breeding status from year to year, and prevent young individuals from recruiting if the population is at `K`? Specify which sex should be retained: `"none"`, `"both"`, `"male"`, or `"female"`. Only used when `matingSys = "monogamy."` When `"none"`, all mature individuals recruit; then individuals are randomly removed from that pool to truncate the population at `K` (new recruits may randomly replace established breeders). When adults will likely survive and prevent new individuals from recruiting, e.g. with territorial species, set this at one of the other values as appropriate for your species. When pairing off widowed or divorced individuals, those of the retained sex(es) that bred previously will stay in the breeding population and obtain a new mate; non-retained adults will compete with new recruits to obtain a mate. When `retainBreeders` is set to a value other than `"none"`, if

the population is at `K`, new recruits will only fill vacancies left by adults that died (they will not replace any surviving adults, including females when `retainBreeders = "male"` and vice versa; i.e. `"both"` functions the same as `"male" and "female"` in this part of the model). Defaults to `"male"`.

`MaxAge` Maximum allowable lifespan (in years); can be `Inf`.

`SenesAge` Age (in years) after which annual survival will be reduced by senescence. Through this age, adult survival values are set according to `adsurvivalF` and `adsurvivalM` (below); after this age, annual survival decreases linearly to 0 at `MaxAge` (see Model Flow 4.d.i., pg. 16). Can be `Inf`.

Expected demography of the new population

`adsurvivalF` Annual survival rate of adult females. All survival rates are given as a proportion (e.g. 0.85).

`adsurvivalM` Annual survival rate of adult males.

`nonbrsurv` Annual survival rate of nonbreeders (subadults or adults that have not recruited to breed).

`nonbrsurvK` Annual survival rate of nonbreeders when population is at `K` (used instead of `nonbrsurv`). Survival probability in each year depends on density of the population at the beginning of that year (see Model Flow 4.d.ii., pg. 16).

`juvsurv` First year survival (from the stage described by `youngperF`, below, to the beginning of the next breeding season) when population is below `K`.

`juvsurvK` First year survival when population is at `K` (used instead of `juvsurv`). Can be equal to `juvsurv`. Otherwise, juvenile survival is density-dependent as for `nonbrsurvK` (see Model Flow 4.d.ii.-iii., pg. 16).

`youngperF` Average number of offspring produced per mating each year (averaged over all females in the population). For a polyandrous female, this is the average number of offspring produced each time she mates: `youngperF` * `nMatings` = total average offspring per year. `youngperF` can be calculated for any reproductive stage (eggs, chicks, independent juveniles) as long as `juvsurv` indicates the proportion of individuals that survive from this stage to the beginning of the following breeding season. Given as offspring per pair, e.g. 1.5 or 0.75.

`SDypF` Among-individual standard deviation of `youngperF`, e.g. 0.50 or 2.

`ypF1` Where younger breeders have reduced reproductive rates, this can be used to define the reproductive success for the first reproductive stage (length of that stage is determined by `ypF1yr`, below). Given as a proportion of `youngperF`; e.g. if `youngperF` = 2 and `ypF1` = 0.5, mean reproductive success during the first stage will be 1 offspring per female. Can be < 1 if inexperienced females experience lower reproductive success than older females; or > 1 to indicate higher reproductive success for younger females than for older females (`youngperF` always applies after the age indicated by `ypF1yr`, below). Defaults to 1.

`ypF1yr` Max age at which `ypF1` applies; e.g. 1 if `ypF1` applies to one-year-olds only, or 5 if `youngperF` applies from age 6 onward.

`MAXypF` Maximum annual number of offspring per individual (e.g. based on biological constraints such as clutch size/renesting). Note that if `ypF1` > 1, this value may be exceeded by some individuals.

`MAXypFK` Maximum annual number of offspring per individual when population is at `K` (if different from `MAXypF`).

`ypFsex` Which member of a pair limits the female's reproductive output for the year, based on the biology of the species of interest. Can be `"female"`, `"male"` (e.g. if the father cares for the young, his quality may be more important than the female's), or `"both"` (which will average the male's and female's values).

`youngSR` Proportion of offspring that are male (range 0-1, defaults to 0.5).

Simulation and output specifications

`trackall` Whether to track all individuals from the population through the whole simulation (`TRUE` or `FALSE`). Must be `TRUE` if you wish to use `indiv.summary` or `pedigree.summary` after running the simulation (see Model Output, next page). The simulations will be noticeably slower (and require more RAM) if this is set to `TRUE`, especially with larger carrying capacity, higher fecundity, more replicates, and longer simulation periods. Defaults to `TRUE`.

`GeneCount` Which individuals should be included (`"all"`, or only breeding `"adults"`) when the number of rare alleles is counted in the population each year.

`nyears` Number of years to run the simulation. Consider setting a value that corresponds to 10 generations of your study species.

`nrepl` Number of replicates to run. More replicates will increase running time, but will produce narrower confidence limits for estimated output. Defaults to 100, but 1000 may be more useful for estimating allele retention.

`nreplprint` Interval (number of replicates) at which to print a message with the current system time. Allows the user to gauge model progress and to estimate time to completion. Defaults to 10.

`printplots` Whether to plot the population growth (number of individuals, as defined by `KAdults`, present each year) and allele frequency (in the pool defined by `GeneCount`) as they change over time (`TRUE` or `FALSE`). If `TRUE`, one line will be plotted for each replicate immediately after it runs. Can be used to immediately gauge the demographics of the population (e.g. if it will grow as expected) during test runs; will slow down the simulation by ~ 10-20%. Defaults to `FALSE`.

Model Output

The following summary functions are included to deal with output from `aRetain`:

`aRetain.summary`

This function summarizes the output from `aRetain` by averaging the census estimates across replicates. It has four arguments:

1. `adata` Object output by `aRetain`.
2. `GeneCount` As specified for aRetain.
3. `alpha` Significance level for confidence limits, e.g. 0.05.
4. `dropextinct` Exclude replicates in which the population went extinct (`TRUE` or `FALSE`). Defaults to `TRUE`; a message displays how many replicates this included.

Output: a matrix with `nyears` rows and 16 columns providing the following summary statistics for each year:

1. `MeanN` Mean number of adults.
2. `SEN` SE of `MeanN` .
3. `MeanNNonbr` Mean # of nonbreeders.
4. `MeanBrF` Mean # of breeding females.
5. `SEBrF` SE of `MeanBrF`.
6. `MeanBrM` Mean # of breeding males.
7. `SEBrM` SE of `MeanBrM`.
8. `MeanNFound` Mean # of founders (starters and supplementals) alive.
9. `MeanNMigr` Mean # of immigrants alive.
10. `MeanAge` Mean age of breeding adults (those that bred that year).
11. `P.extant` Probability that the population is extant.
12. `P.xLCL` Lower bound of the confidence interval for `P.extant`.
13. `P.xUCL` Upper bound of the confidence interval for `P.extant`.
14. `P.retain` Probability that the rare allele is present in the population.
15. `P.LCL` Lower bound of the confidence interval for `P.retain`.
16. `P.UCL` Upper bound of the confidence interval for `P.retain`.
17. `A.Freq` Frequency of the rare allele in the simulated population (in adults or in all individuals, as per `GeneCount` in `aRetain`).
18. `A.SE` SE of `A.Freq`.

`indiv.summary`

This function summarizes individual data by origin (starter, supplemental, local, migrant) to compare fates across origins. Can only be run after running `aRetain` with `trackall = TRUE`. `indiv.summary` has three arguments:

1. `adata` Object output by `aRetain`.
2. `genlength` Mean age of breeding individuals, as returned by `aRetain.summary` (after recovery from any founder age effects).
3. `alpha` Significance level for confidence limits, e.g. 0.05

Output: a matrix with four rows, one for each origin: starters, supplementals, locals, and migrants. All of these exclude individuals that died immediately post-release (subject to `inisurv`) and those added in the last generation. The function provides the following summary statistics, averaged across replicates for individuals of each origin:

1. `n` Total number of individuals from each origin, summed within each replicate and then averaged across replicates.
2. `pbreed` Probability of an individual breeding (proportion that bred).
3. `pbreed.LCL` Lower binomial confidence limit for `pbreed`.
4. `pbreed.UCL` Upper binomial confidence limit for `pbreed`.
5. `YrsBred` Mean # of years bred per individual (including those that never bred).
6. `YrsBredBr` Mean # of years bred per individual (that bred at least once).
7. `lifespan` Mean lifespan in the population.
8. `effectivegen` Mean # of individuals that bred each generation.
9. `NMatings` Mean # of lifetime matings

`pedigree.summary`

This uses function `calcInbreeding` from package **pedigree** (Coster 2011) to calculate the inbreeding coefficient (F) of each individual, the average across individuals alive each year (including founders, who are assigned F = 0 because ancestry is not known) within each replicate, and the average across replicates for each year. Can only be run after running `aRetain` with `trackall = TRUE`. This function has one argument:

1. `adata` Object output by `aRetain`.

Output: a matrix with one row for each year and 4 columns:

1. `year` Year of simulation, not including founding year (0).
2. `meanF` Mean F, averaged across individuals and replicates.
3. `varF` Inter-replicate variance in F.
4. `indivVarF` Inter-individual variance of F, averaged across replicates.

`LRS.summary`

This function calculates the number of matings for each individual of the specified sex, over the individual's full lifetime during the simulation. Can be run only after running `aRetain` with `trackall = TRUE`. This function has two arguments:

1. `adata` Object output by `aRetain`.
2. `sex` Which sex to use in the calculations.

Output: a matrix with one row for each individual that lived during each replicate of the simulation and 2 columns:

1. `ID` ID number of the individual. ID numbers are unique within each replicate, but will be repeated across replicates.
2. `NMatings` Number of times the individual mated during its lifetime. The individual-level data is not expected to be useful, but can be used to calculate the mean or a histogram of mating success.

`agerepro.summary`

This function calculates the average reproductive success, by age and sex, in the final year of the simulation. Can only be run after running `aRetain` with `trackall = TRUE`. This function has three arguments:

1. `adata` Object output by `aRetain`.
2. `maxage` Value of `MaxAge` used in `aRetain`, or the maximum age of interest for the summary, whichever is smaller.
3. `sex` Which sex to use in the calculations.

Output: a matrix with one row for each age and 7 columns:

1. `age` Age of each individual, from 0 to `MaxAge`
2. `alive.mean` Mean number of individuals that lived to this age.
3. `alive.sd` SD across replicates in the number of individuals that lived to this age.
4. `matings.mean` Mean number of matings per individual at this age.
5. `matings.sd` SD among individuals in `matings.mean`.
6. `offspring.mean` Mean number of offspring produced by individuals of this age.
7. `offspring.sd` SD among individuals in `offspring.mean`.

Troubleshooting

If the model stops immediately after starting and outputs an error message:

> `aRetain` and its summary functions provide informative error messages in response to unexpected input. For example, if you try to input a decimal where a whole number is needed, the simulation stops with an error message (e.g. `"Error in FUN(1:30[[1L]], ...) : 'migrfreq' must be a whole number"`). Input values are checked one at a time, so after you fix one error, another may appear when you restart the simulation.

If the model stops and outputs an error message after running for a while:

> `"Error: cannot allocate vector of size X.X Mb"` indicates that you do not have enough memory to store the information generated and used by the model. Increase the memory allotted to R with `memory.limit(size=XXXX)` where XXXX = the maximum for your machine (4095 for a 32-bit system, 8000000000 for a 64-bit system). Be sure to clear objects that are no longer needed, e.g. between simulations, with `rm`. If your system does not provide enough memory for your simulation, you have a few options (some of which may limit the usefulness of the simulation for your purposes):
>
> 1) Use a 64-bit system rather than a 32-bit system.
> 2) Use a 64-bit system with more RAM.
> 3) Run fewer replicates or simulate a shorter time period.
> 4) Simulate a smaller population size.
> 5) Run with `trackall = FALSE` to simulate allele retention; then run a smaller simulation (e.g. 100 replicates instead of 1000) with `trackall = TRUE` to run the pedigree and individual summaries. Most of these statistics are easier to estimate than allele retention so fewer replicates are needed.

If the output values indicate unexpected model function:

> Check the demography of your population (from `aRetain.summary`). Does it grow as expected? Does it decline to extinction in many replicates? Is the nonbreeder:breeder ratio too high, perhaps due to a juvenile/nonbreeder survival rate that is too high? Any demographic parameter can affect allele retention.
>
> Remember that the `.summary` functions should only be run after `aRetain` was run with `trackall = TRUE`. Otherwise, these functions will give unexpected values (e.g. very small inbreeding coefficients or very high probabilities of breeding) because not all individuals will be included. Note that if your simulation did not include animals of a certain origin (e.g. supplement or migrant), you will see NA values in the corresponding row of the output from `indiv.summary`; this does not indicate a problem.

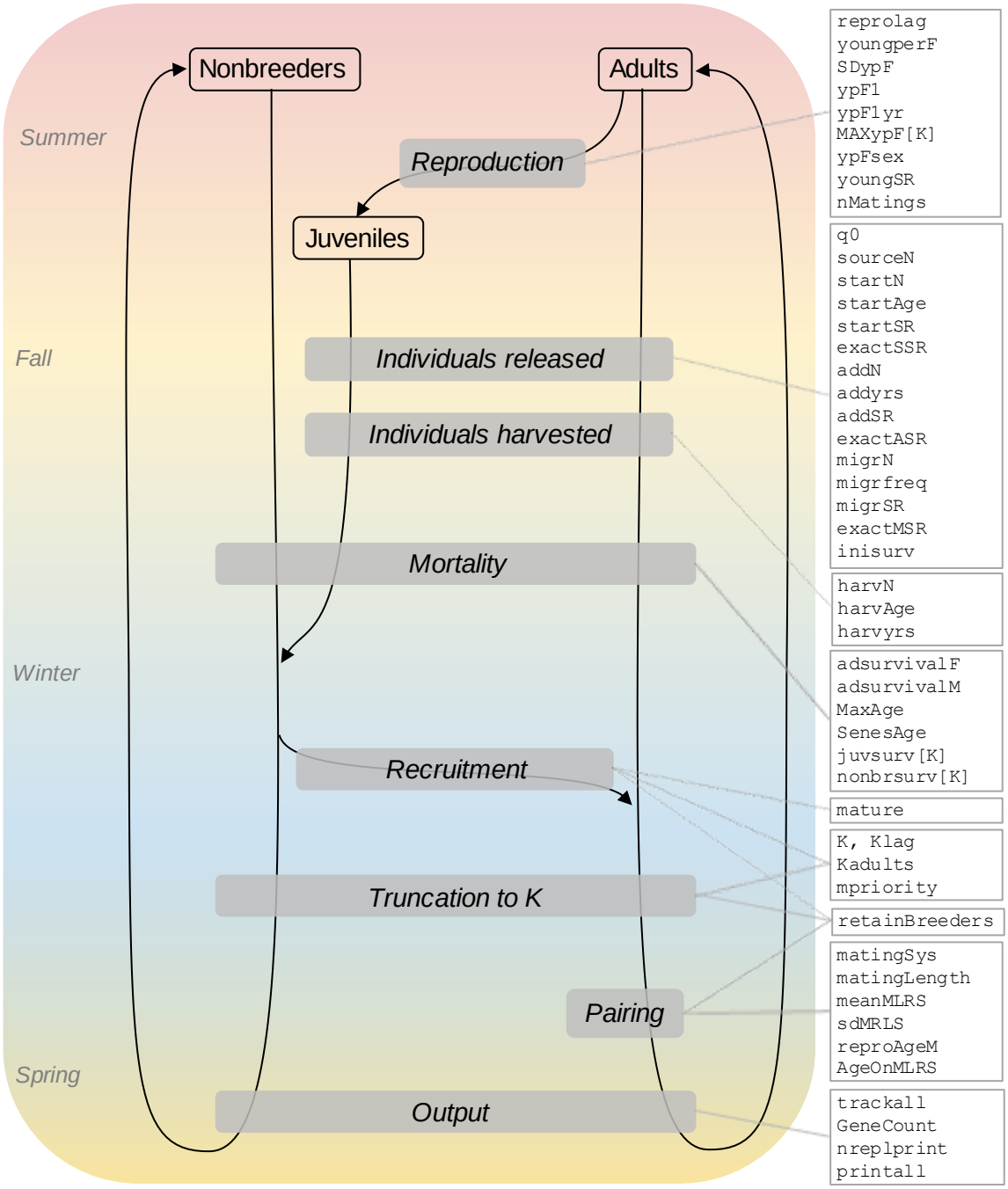

Figure 1. Flow diagram of the `aRetain` function of **AlleleRetain**, showing the age classes tracked (transparent boxes) and events (gray boxes) that influence each age class at the indicated points in the annual loop. User-specified parameters (white boxes) are connected to the events during which they are called; some parameters can change at carrying capacity, as indicated by "[K]" (e.g. `MAXypF` and `MAXypFK`). Each year loop (Summer through Spring) is repeated over the simulation period specified by `nyears`, and the simulation period is repeated over the number of replicates specified by `nrepl`. The seasons listed assume the simulated species breeds in summer (as an example only; does not affect program function). Note that individuals are released in Fall; the first half-year simulated after starters are released is year 0.

Model Flow

The following is a plain-English description of the procedures performed when a call is made to each of the functions provided in **AlleleRetain**. These details are provided to aid in understanding output, to troubleshoot any unexpected behavior of the program, and to assist advanced users in modifying the source code if desired. Input and output (arguments, objects) and functions are in `Courier New font`. R package names are **bolded**. Internal objects (which are used by, but not output from, **AlleleRetain** or its functions) are in *italics*.

`aRetain`

1. The model checks all input to ensure values are valid; if not, the simulation stops and outputs a warning message.

2. The model sets up empty matrices needed later (including *population*, which will contain adults that have recruited into the breeding population and individuals that have died if `trackall = TRUE`; *nonbreeders; juveniles; migrants*)

3. Initial population is formed with `addnew` and `newinfo`.

4. Each year (looped over 1:`nyears`):

    a. Starters are put into *nonbreeders* if they are adults (no breeding will occur this year), or into *juveniles* otherwise. This is year 1.

    b. *pairs* (if any) are also stored in *oldpairs* to later assess which ones were paired last year.

    c. The total number of adults (if `KAdults = TRUE`) or all individuals (if `KAdults = FALSE`) now present is recorded to be used later in density-dependent effects.

    d. Breeding occurs (unless in year 0 or within `reprolag` years of founding):
        i. # young/year for females in the first reproductive stage is adjusted by multiplying their individual means by `ypF1`.
        ii. Reproduction is simulated by running *pairs* through `breed`; offspring are put into *juveniles*.
        iii. If `nMatings` > 1, *pairs* are re-formed and breeding repeats until `nMatings` is achieved for females.
        iv. Information from `newinfo` and parent IDs are recorded for each offspring.

    e. Any supplementals are added with `addnew` and `newinfo`. The year in which they were added is recorded. If juvenile, they are added to *juveniles;* if adult, they are added to *nonbreeders.*

    f. Any immigrants are added, as for supplementals. If `removeL = TRUE`, the corresponding number of local adults is removed from *population*.

g. If harvest is indicated for this year, the specified numbers and ages of individuals are removed from the population. If harvest is limited to one age class, and there aren't enough individuals of that age to meet `harvN`, the harvest quota is not met.

h. Survival: individuals are randomly selected to survive or die, based on the appropriate survival probability:
   i. Adult: survival rate is adjusted for age (Fig. 2a):

$$S_a = S_x - \frac{S_x}{\texttt{MaxAge} - \texttt{SenesAge}} * (a - \texttt{SenesAge})$$

   where S_a = survival at age a and S_x = `adsurvivalF` or `adsurvivalM`.
   ii. Nonbreeder: survival rate is density-dependent (Fig. 2b):

$$S_{E_t} = \frac{S_0}{1 + \beta * E_t}$$

   where S_{E_t} is survival rate at population density E in year t, S_0 is survival when density is near 0, β is the decline in survival as density increases, and E_t is population density at time t (Morris & Doak 2002). "Density" is defined in this model as the proportion of `K` that has been filled. The model solves for β according to the user-specified values for `nonbrsurv` (S_0) and `nonbrsurvK` (S_1), then uses β and S_0 to calculate density-dependent survival probability in each year. Nonbreeder survival is applied to all individuals in *nonbreeders*, even those that are older than `mature` (they remain in *nonbreeders* until they recruit to breed).
   iii. Juvenile: density dependent as for nonbreeders, depending on `juvsurv`/`juvsurvK`.

i. *juveniles* are added to *nonbreeders*.

j. Maturing individuals are randomly selected to recruit into the breeding population (not all those selected may be able to pair, depending on the sex ratio). If `retainBreeders = "none"`, all maturing individuals recruit into *population*, which is later truncated to `K` by randomly removing individuals (new recruits have the same chance as established breeders to remain in the population). Otherwise, only the number of individuals needed to reach `K` (or `startN`, if within `Klag`) is recruited from nonbreeders that will be old enough to breed next year (depending on `mature`). If `mpriority = TRUE`, immigrants are selected first; then locals are randomly selected to fill any remaining spaces.

k. The population is truncated to `K` (or to `startN`, if within `Klag`), if necessary. This is not necessary if `KAdults = TRUE` and `retainBreeders ≠ "none"` because recruitment was limited to only filling vacancies; there are no surplus individuals.
   i. If `KAdults = FALSE`, *nonbreeders* are included in carrying capacity (`K`). Otherwise, only adults in *population* are included.

ii. Individuals that have priority are retained through truncation:
    1. Retained breeding adults, if any (depending on `retainBreeders`). Only necessary if `KAdults = FALSE`, because if `KAdults = TRUE` and `retainBreeders ≠ "none"`, no surplus individuals are present (see 4.h.i).
    2. Immigrants, if `mpriority = TRUE`.
iii. Then other available individuals are randomly selected until the population is at `K`. Dead nonbreeders are moved into *population*.

l. *pairs* are re-formed for next year (by `pairoff` if monogamy or `polypair` otherwise):
    i. If `matingLength = "seasonal"`, *pairs* are split up and all individuals are put into the *singles* pool. Otherwise, *pairs* with both members still alive remain intact.
    ii. If `matingSys = "monogamy"`, widowed or divorced individuals of the sex indicated by `retainBreeders` are guaranteed a new mate (if the opposite sex is available). Individuals of the non-retained sex(es) are returned to the *singles* pool.
    iii. If `matingSys = "polygyny"` or `"polygynandry"` and `matingLength = "lifelong"`, males will not receive a new mate if at least one of their previous mates is still alive (probably not realistic for most species).
    iv. Any remaining *singles* are paired (as the sex ratio allows).

m. Numbers of years alive, breeding seasons, and mates is updated for all individuals.

n. The population is censused.

5. Steps 1-4 are repeated over the specified number of years. The censuses from all years are compiled into one matrix, and the list of individuals simulated in this replicate is saved.

6. Steps 1-5 are repeated over the specified number of replicates.

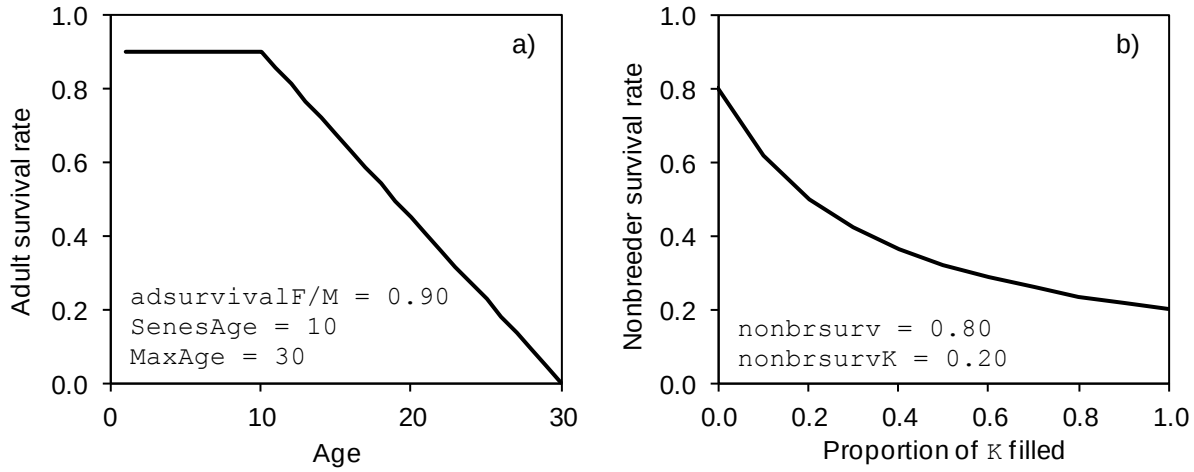

Figure 2. Adjustments to survival rates. a) Adult survival declines linearly with age, from `SenesAge` to 0 at `MaxAge`. b) Nonbreeder survival declines with density (proportion of `K` filled), from near `nonbrsurv` at very low densities to `nonbrsurvK` at `K` (juvenile survival declines the same way, according to `juvsurv`/`juvsurvK`).

Summary functions

See Model Output (pg. 11-13) for further details.

`aRetain.summary`: summarize output across replicates. Averages are produced with the base function `mean`, binomial confidence limits are produced with **AlleleRetain** internal functions `Bucl` and `Blcl`, and standard error of each list of values (x) is calculated as `sqrt(var(x)/sum(x))`.

`indiv.summary`: summarize individual data across replicates. For each origin (starters, supplementals, locals, migrants), summarize information across individuals that were present prior to the last generation. Does not include individuals that died immediately post-release (subject to `inisurv`). All values are rounded to 2 significant digits.

`pedigree.summary`: outputs the mean and variance of inbreeding coefficient for individuals alive in each year, and the inter-individual variance within pedigrees, averaged across replicates. Founders are included in this summary (their F-values default to 0 as their ancestry is not known).

`LRS.summary`: calculate the number of matings for each individual of the specified sex, over the individual's full lifetime during the simulation.

`agerepro.summary`: calculate the average reproductive success, by age and sex, in the final year of the simulation.

Internal functions created for use by `aRetain` or summary functions

`is.wholenumber`: returns `TRUE` or `FALSE` depending on whether the input value is a whole number. Used to check input values, to check whether the current year is an immigration year (`is.wholenumber (y/migrfreq)`), and to check whether the current replicate is one after which the message "`Replicate # [r] finished at [time]`" is printed.

`pad`: returns a vector padding the given data (a vector) with 0s to achieve the specified length (given 1:5, length = 10, will return 1,2,3,4,5,0,0,0,0,0). Used in `pairoff`.

`Blcl`, `Bucl`: calculate Wilson's binomial confidence interval (lower and upper limits, respectively). Used in `aRetain.summary` to summarize output across replicates.

`pairoff`: Monogamous pairs (to the maximum number specified) are formed based on sex. Individuals are split into "mated" (in pairs) and "unmated".

`polypair`: Polygynous pairs are formed based on sex and relative quality of males. If only one male with LRS > 0 is present, all females mate with him. Otherwise, each male's probability of mating is determined by dividing his LRS score by the sum of the scores of all other males present. One mate for each female is randomly drawn from the available males,

based on their probabilities of mating, with any male able to be selected more than once. All females are assigned a mate. Unpaired males are put in "unmated" pool; pairs are listed as "mated" (each male may be listed more than once). (`aRetain` will later put these groups into *singles* and *pairs*, respectively.)

`getsingles` and `dropsingles`: two internal functions used together to move widowed individuals from *pairs* into *singles.* `getsingles` adds widowed individuals to *singles*; `dropsingles` removes those individuals from *pairs*.

`breed`: Offspring are produced based on characteristics of the breeding pairs.

1. Each pair is assigned a mean number of juveniles produced annually, based on the male's mean, the female's mean, or an average of the two (depending on `ypFsex`)
2. The number of offspring actually produced by that pair in that year is drawn from a Poisson distribution with the pair's mean. This number is constrained to `MAXypF` if necessary.
3. Each offspring for that pair is assigned a sex (based on `youngSR`), it inherits one allele from the mother and one from the father, and the unique IDs of its parents are recorded.
4. Loop through all pairs.
5. All offspring are combined into one matrix and output.

`addnew`: Add new individuals (from additional translocations) to the population.

1. The number of individuals that survive translocation is drawn from a binomial distribution with size = `startN` and probability = `inisurv`. Each individual is assigned a sex, randomly (by drawing from a binomial distribution) based on `startSR` if `exactSSR` = FALSE, or to exactly meet `startSR` if `exactSSR` = `TRUE`. If `startSR` cannot be met exactly (e.g. if `startN` is odd but `startSR` = 0.50), the remainder is assigned as female.
2. Genotypes are assigned to each individual (0, 1, or 2 copies of the rare allele described by `q0`), randomly if `sourceN` = `Inf`, or after randomly permuting the genes in the source population according to Hardy-Weinberg Equilibrium and `sourceN`.

`newinfo`: Describe individual information for individuals added to the population (via either translocation or reproduction).

1. The "origin" of each individual is assigned (1 = starter, 2 = supplemental, 3 = local, 4 = migrant).
2. A unique ID is assigned to each individual.
3. Year of birth is assigned: Local offspring and individuals translocated as juveniles are assigned the current year. Individuals translocated as young adults are assumed to have just reached sexual maturity and are assigned birth years accordingly (e.g. if released in year 10 and `mature` = 4, they are assigned year 6 as their birth year). For individuals released as adults, estimated age of each individual is randomly drawn based on the expected proportion of individuals that survive to each age:

$$P_a = P_{a\text{-}1} * S_{a\text{-}1}$$

where P_a is the proportion that survive to age a and S_a is survival rate at age a. The proportion at each age = the probability that any given starter is that age. The birth year for each adult is then assigned by subtracting their age from the current year.

4. Each individual is assigned a mean number of young produced per year, drawn from a normal distribution with mean = `youngperF` and SD = `SDypF`. Any negatives are converted to 0, and anything above `MAXypF` is constrained to `MAXypF`.
5. Each individual is assigned a value for lifetime reproductive success (LRS) by drawing from a gamma distribution with mean = `meanMLRS` and SD = `sdMLRS`. The shape of the gamma distribution = $\texttt{meanMLRS}^2/\texttt{sdMLRS}^2$; scale = $\texttt{sdMLRS}^2/\texttt{meanMLRS}$. The gamma distribution was chosen because of its flexibility in shape appropriate to polygynous mating systems (from strongly right-skewed to nearly symmetrical). These values are rounded to two decimal places to ensure that very tiny numbers are rounded to 0 (those individuals will never breed). Any negatives are set to zero. This value is only used for males (not females).
6. Each individual is assigned a 0 for number of years alive in the population and number of years bred (because this information is recorded when the individual is just entering the population).
7. Parents are recorded as NA (unknown) for translocated (assumed wild-caught) individuals.

References

Coster, A. 2011. **pedigree**: Pedigree functions. R package version 1.3.2. http://CRAN.R-project.org/package=pedigree

Efford, M. G. 2010. **mohuasim**: a stochastic model for the loss of rare alleles from re-introduced populations. R package version 1.2.

Morris, W. F., and D. F. Doak. 2002. Quantitative conservation biology. Sinauer Associates, Sunderland, Massachusetts.

R Development Core Team. 2011. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org.

Tracy, L. N., G. P. Wallis, M. G. Efford, and I. G. Jamieson. 2011. Preserving genetic diversity in threatened species reintroductions: how many individuals should be released? Animal Conservation **14**:439-446.

Weiser, E.L., Grueber, C. E., and I. G. Jamieson. 2012. **AlleleRetain**: A program to assess management options for conserving allelic diversity in small, isolated populations. Molecular Ecology Resources 12:1161-1167.

Weiser, E.L., Grueber, C. E., and I. G. Jamieson. 2013. Simulating retention of rare alleles in small populations: assessing management options for species with different life histories. Conservation Biology 27:335-344.

Appendix 1: Example Code

Here's an example of code to run several scenarios through `aRetain`; save output from `aRetain.summary`, `indiv.summary`, and `pedigree.summary` into .csv files; and plot certain output. Four scenarios are run using this code: the parameters that vary between scenarios (starting population size and migration) are specified first, with the remaining parameter values assigned explicitly within the code. This code produces plots as the simulation progresses by calling from the R base `plot` function.

Parameter values used here were estimated for North Island robins *Petroica longipes* (Weiser et al. 2013) with `K` applied only to adults and adjusted to achieve 50 breeding pairs. This example runs for 5 robin generations (25 years) over 100 replicates (so the confidence intervals are wide).

Total run time for the 4 scenarios may be 5 minutes or so, depending on your system. A message will display after every 10 replicates to inform you of the progress of the simulation. Hit the Escape key at any time to abort the simulation (no data will be saved for scenarios that have not finished running).

NOTE: If you try to copy and paste this code from this PDF document, it might not run due to font translation issues. Instead, download the example code from the AlleleRetain website: http://sites.google.com/site/alleleretain/download; choose the file called "AlleleRetain Appendix 1 Example Code.txt". Alternatively, you can type your own code into your own .txt or R script file, working from this example (all lines that start with a "#" are comments to help guide you, not something that you need to put into R for the model to run).

```
## Load AlleleRetain
library(AlleleRetain)

## Specify the location where you want to save output files, e.g.:
loc <- "C:/Program Files/R/"

## Increase the memory limit to the max allowed on your machine (4095 for a
32-bit system; 8000000000 on a 64-bit system):
memory.limit(size=8000000000)

## Specify the values for any parameters for which you'd like to try several
scenarios:
startN.set <- c(20, 40)
migrN.set <- c(0, 15)

## Define the number of values to try for each parameter (must correspond to
the lengths of the value lists specified above):
nN <- length(startN.set)
nM <- length(migrN.set)

## Set up the plotting frame to compare scenarios
par(mfrow=c(nN*nM, 4), mar=c(4.5,4.5,2,1))

## Display the current time so you know when you started the simulation:
```

```
(Start_time <- Sys.time())

## Loop over each combination of the parameter values listed above (each
scenario is simulated sequentially):
for(i in 1:nN){ for(j in 1:nM){

## Run each aRetain for scenario:

robin <- aRetain(

# Source population:
q0 = 0.05,
sourceN = Inf,

# Translocated individuals:
startN = startN.set[i],       # Pulled from the set listed above
startAge = "adult",
startSR = 0.5,
exactSSR = FALSE,
inisurv = c(1,1,1),
addN = 0,
addyrs = 0,
addSR = 0.5,
exactASR = FALSE,
migrN = migrN.set[j],         # Pulled from the set listed above
migrSR = 0.5,
exactMSR = FALSE,
migrfreq = 5,
mpriority = TRUE,
removeL = FALSE,

# Characteristics of the new population:
K = 108,
Klag = 0,
KAdults = TRUE,
reprolag = 0,

# Life history of the species:
mature = 1,
matingSys = "monogamy",
matingLength = "lifelong",
meanMLRS = 1,                 # meanMLRS, sdMLRS, reproAgeM, AgeOnMLRS,
nMatings are not used in monogamous systems
sdMLRS = 0,
reproAgeM = c(1:16),
AgeOnMLRS = "age/age",
nMatings = 1,
retainBreeders = "both",
MaxAge = 16,
SenesAge = 5,

# Expected demography in the new population:
adsurvivalF = 0.77,
adsurvivalM = 0.77,
nonbrsurv = 0.60,
nonbrsurvK = 0.30,
juvsurv = 0.60,
```

```
juvsurvK = 0.30,
youngperF = 3.19,
ypF1 = 1,
ypF1yr = 1,
SDypF = 1.23,
MAXypF = 6,
MAXypFK = 6,
ypFsex = "female",
youngSR = 0.5,

# Simulation and output specifications:
trackall = TRUE,
GeneCount = "adult",
nyears = 25,
nrepl = 100,
nreplprint = 10,
printplots = FALSE)

## Name this scenario based on which values were used:
fname <- paste("ROBIN_startN", startN.set[i], "_migr", migrN.set[j], sep="")

## Generate summary output for this scenario:

## Run the census summary:
robinsum <- aRetain.summary(robin, GeneCount = "adult", alpha=0.05,
dropextinct=TRUE)
## Name the census summary file:
sfn <- paste(loc, fname, "_Sum.csv", sep="")
## Save the main summary file:
write.csv(robinsum, file=sfn)

## Run, name and save the individual summary:
indiv <- indiv.summary(robin, genlength=5)
ifn <- paste(loc, fname, "_Indiv.csv", sep="")
write.csv(indiv, file=ifn)

## Run, name, and save the pedigree summary:
pedsum <- pedigree.summary(robin)
pfn<-paste(loc, fname, "_Pedigree.csv", sep="")
write.csv(pedsum, file=pfn)

## Display each summary file for immediate viewing, to 2 decimal places:
print(robinsum[20:25,], digits=2)   # only show the last 5 years
print(indiv, digits=2)
print(pedsum[20:25,], digits=2)           # only show the last 5 years

## Plot particular output (one row for each scenario):
if(i==1) if(j==1){
      Nmain = "Population growth"
      Amain = "Allele retention"
      Fmain = "Inbreeding (F)"
      Imain = "Prop. that bred"
}
else {
      Nmain = NULL
      Amain = NULL
      Fmain = NULL
```

```
      Imain = NULL
}
# Plot population growth:
      plot(robinsum[,1], main=Nmain, ylab="# adults", ylim=c(0, 110),
xlab="year", type="l", lwd=2, cex.main=1.5, cex.lab=1.4, cex.axis=1.25,
font.main=1, las=1)
# Add labels for the scenario modeled
      mtext(paste("startN =", startN.set[i], sep=" "), side=1, line=-3,
cex=0.9, at=16)
      mtext(paste("migrN =", migrN.set[j], sep=" "), side=1, line=-1.5,
cex=0.9, at=15)
# Plot allele retention
      plot(robinsum[,14], main=Amain, ylab="prob. retain allele",
xlab="year", ylim=c(0,1), type="l", lwd=2, cex.main=1.5, cex.lab=1.4,
cex.axis=1.25, font.main=1, las=1)
      lines(robinsum[,15])
      lines(robinsum[,16])
# Plot inbreeding accumulation
      plot(pedsum[,2], main=Fmain, ylab = "mean F", xlab = "year",
ylim=c(0,0.2), type="l", lwd=2, cex.main=1.5, cex.lab=1.4, cex.axis=1.25,
font.main=1, las=1, yaxp=c(0, 0.2, 2))
# Plot probability of breeding for each type
      plot(indiv[,2 ], main=Imain, ylab="proportion that bred", xlab=NULL,
ylim=c(0,1), xaxt="n", cex=2, pch=18, cex.main=1.5, cex.lab=1.4,
cex.axis=1.25, font.main=1, las=1)
      axis(side=1, at=c(1:4), labels=c("start", "suppl", "local", "migr"),
cex.axis=1.25)

# Remove stored objects to free up memory to run the next scenario:
rm(robin, robinsum, pedsum, indiv)

} }   # close the loop and run the simulation

# Time it took the whole simulation to run:
Sys.time() - Start_time
```

Appendix 2: Example Output

Some example output from code in Appendix 1, with `startN` = 20, `migrN` = 15, is shown below. Your numbers might be slightly different due to the probabilistic nature of the model.

Excerpt for last 5 years (20-25) from `aRetain.summary`:

	MeanNAd	SEN	MeanNNonbr	MeanBrF	SEBrF	MeanBrM	SEBrM	MeanNFound	MeanNMigr
20	108	0	33	50	0.32	50	0.32	0	6.0
21	108	0	30	50	0.33	50	0.33	0	4.4
22	108	0	29	50	0.34	50	0.34	0	3.5
23	108	0	30	50	0.31	50	0.31	0	2.7
24	108	0	31	49	0.29	49	0.29	0	2.1
25	108	0	34	50	0.32	50	0.32	0	6.3

MeanAge	P.extant	P.xLCL	P.xUCL	P.retain	P.LCL	P.UCL	A.Freq	A.SE
4.8	1	0.95	1	0.87	0.78	0.93	0.054	0.0051
4.7	1	0.95	1	0.87	0.78	0.93	0.053	0.0053
4.7	1	0.95	1	0.87	0.78	0.93	0.051	0.0054
4.6	1	0.95	1	0.86	0.77	0.92	0.052	0.0054
4.6	1	0.95	1	0.84	0.75	0.90	0.053	0.0054
4.7	1	0.95	1	0.89	0.81	0.94	0.054	0.0053

Output from `indiv.summary`; values for supplementals are `NaN` (NA) because none were added:

	n	pbreed	pbreed .LCL	pbreed .UCL	Yrs Bred	Yrs BredBr	lifespan	effectivegen
starters	20	0.58	0.35	0.78	2.21	3.8	2.4	NaN
supplement	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
locals	2116	0.24	0.22	0.26	0.89	3.7	1.1	102
migrants	54	0.46	0.33	0.60	1.58	3.4	1.5	5

Excerpt for the last 5 years from `pedigree.summary`:

year	meanF	varF	indivVarF
20	0.073	0.00025	0.0032
21	0.073	0.00024	0.0031
22	0.073	0.00023	0.0031
23	0.074	0.00023	0.0030
24	0.076	0.00026	0.0030
25	0.077	0.00028	0.0028

Plots output by example code given in Appendix 1 are shown below. Each row represents a different scenario (`startN` and `migrN` as indicated on the left-hand panels). Fine lines on the allele retention graphs indicate 95% confidence limits; the jumps in allele retention occur when migrants are added every 5 years. Migrants were given priority to recruit into breeding vacancies, but they were still subject to low density-dependent annual survival as nonbreeders, so only a fraction was able to breed.