Package ‘simRestore’

November 17, 2023

Type Package

Title Simulate the Effect of Management Policies on Restoration Efforts

Version 1.1.4

Description Simulation methods to study the effect of management policies on efforts to restore populations back to their original genetic composition. Allows for single-scenario simulation and for optimization of specific chosen scenarios. Further information can be found in Hernandez, Janzen and Lavretsky (2023) <doi:10.1111/1755-0998.13892>.

License GPL (>= 2)

Imports Rcpp, shiny, subplex, tibble

Suggests dplyr, ggplot2, knitr, magrittr, rmarkdown, shinyBS, shinythemes, shinyWidgets, testthat, tidyr, egg

LinkingTo Rcpp

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation yes

Author Thijs Janzen [aut, cre]

Maintainer Thijs Janzen <`thijsjanzen@gmail.com`>

Repository CRAN

Date/Publication 2023-11-17 09:50:02 UTC

R topics documented:

optimize_adaptive 2
optimize_static 5
run_shiny_app 8
simulate_policy 9

Index **13**

`optimize_adaptive` *Optimize a policy assuming a fixed total sum across all generations of individuals that can be put or pulled (e.g. a fixed effort). This fixed total sum is distributed across the generations following a beta distribution, and the parameters of this beta distribution are fitted.*

Description

Optimize a policy assuming a fixed total sum across all generations of individuals that can be put or pulled (e.g. a fixed effort). This fixed total sum is distributed across the generations following a beta distribution, and the parameters of this beta distribution are fitted.

Usage

```
optimize_adaptive(
  target_frequency = 0.99,
  initial_population_size = 400,
  reproduction_success_rate = 0.387,
  reproductive_risk = c(0.2, 0),
  K = 400,
  num_generations = 20,
  optimize_put = 100,
  optimize_pull = 0,
  starting_freq = 0.2,
  sd_starting_freq = 0.05,
  morgan = c(1),
  establishment_burnin = 30,
  num_replicates = 1,
  max_age = 6,
  mean_number_of_offspring = 6,
  sd_number_of_offspring = 1,
  genetic_model = "point",
  smin = 0.5,
  smax = 0.9,
  b = -2,
  p = 0.5,
  sex_ratio_put = 0.5,
  sex_ratio_pull = 0.5,
  sex_ratio_offspring = 0.5,
  ancestry_put = 1,
  ancestry_pull = 1,
  random_mating = FALSE,
  extra_pair_copulation = 0,
  verbose = FALSE,
  return_genetics = FALSE
)
```

Arguments

`target_frequency`	frequency to aim for
`initial_population_size`	population size at the start
`reproduction_success_rate`	frequency of females that yield offspring at the end of the breeding season (e.g. a fraction of 1 - reproduction_success_rate of females. This is a joint effect of breeding females getting killed (see `female_death_rate`) and other sources of failure to produce offspring. Other sources of failure are calculated from reproduction_success_rate and female_death_rate, such that the resulting reproduction failure rate = 1 - reproduction_success_rate / (1 - female breeding risk)
`reproductive_risk`	Additional death rate of males and females as a result of breeding (e.g. as a result of protecting the offspring against predators). Provide as a vector where the first index indicates the risk for females, the second the risk for males.
`K`	carrying capacity
`num_generations`	number of generations
`optimize_put`	optimization proceeds such that the sum of all addition over all generations is equal to this number. Switch off by setting to zero. The individuals are distributed over time following a beta distribution.
`optimize_pull`	Optimization proceeds such that the sum of all removal is equal to this number. Switch off by setting to zero.
`starting_freq`	initial frequency of target ancestry in the population.
`sd_starting_freq`	variation in initial frequency of target ancestry.
`morgan`	a vector with the size of each chromosome in morgan, e.g. if a single chromosome is to be simulated a single number will suffice, but for two chromosomes of a size of 1 Morgan, a vector like: c(1, 1) will work.
`establishment_burnin`	number of generations before establishment
`num_replicates`	number of replicates per parameter combination to be simulated. Fit of the parameter combination is chosen as the average frequency across replicates.
`max_age`	maximum age an individual can reach.
`mean_number_of_offspring`	mean number of offspring per female
`sd_number_of_offspring`	standard deviation of number of offspring per female (assuming the number of offspring is always 0 or larger).
`genetic_model`	The model can either use the point ancestry model ("point") of underlying genetics, which speeds up simulation considerably, but underestimates genetic variation. Alternatively, a more detailed genetic model is available, making use of the theory of junctions, this can be accessed using the option "junctions". Default is "point".

`smin`	minimum survival rate
`smax`	maximum survival rate
`b`	steepness of the survival rate. Negative values indicate a declining survival rate with increasing population size, positive values indicate an increasing survival rate with increasing population size.
`p`	Density at which the survival rate changes most relative. Expressed in N / K (e.g., for a value of 1.0, the survival rate changes most rapidly around N = K, for a value of 0.5, the survival rate changes most rapidly around N = 0.5K, etc).
`sex_ratio_put`	sex ratio of individuals that are added (if any) to the population. Sex ratio is expressed as males / (males + females), such that 0.5 indicates an even sex ratio, 0.9 indicates a male biased sex ratio and 0.1 indicates a female biased sex ratio.
`sex_ratio_pull`	sex ratio of individuals that are removed (if any) from the population. The sex ratio is expressed as males / (males + females), such that 0.5 indicates an even sex ratio, 0.9 indicates a male biased sex ratio and 0.1 indicates a female biased sex ratio.
`sex_ratio_offspring`	sex ratio of newly born offspring. The sex ratio is expressed as males / (males + females), such that 0.5 indicates an even sex ratio, 0.9 indicates a male biased sex ratio and 0.1 indicates a female biased sex ratio.
`ancestry_put`	Average ancestry of individuals being used for supplementation. If the target is high focal ancestry (e.g. aiming for focal ancestry of 1.0), ancestry put should reflect this and be set to 1.0 (which is the default value). When supplementing with non-pure individuals, this value can consequently be lowered.
`ancestry_pull`	Ancestry level below which individuals are allowed to be pulled - this can reduce the effective number of individuals pulled if none of the individuals in the population match this ancestry level. This can be used to selectively only remove those with low target ancestry.
`random_mating`	by default, simulations assume fixed pair bonding, e.g. each female mates with exactly one male (if available). Alternatively, if random_mating = TRUE, females will mate with a random male, introducing the possibility that some males mate multiple times.
`extra_pair_copulation`	probability of offspring to be fathered by another male. We assume that all offspring from one mother can have at most two fathers.
`verbose`	provides verbose output if TRUE.
`return_genetics`	returns a tibble containing all local ancestry information for all individuals. This tibble contains the following informative columns: 1) time (the last generation), 2) replicate, 3) individual, 4) sex (0 = male, 1 = female), 5) Linkage Group (if use_simplified_model == FALSE), 6) chromosome (1 or 2, returning phased results), 7) position (if use_simplified_model == FALSE) and 8) local ancestry (0 or 1).

Value

list with five elements: 1) put: optimal number of individuals to put (0 if not estimated), 2) pull: optimal number of individuals to pull (0 if not estimated), 3) results tibble (see `simulate_policy()`),

4) curve tibble with three columns, indicating the realized number of put/pull per generation, with column 1) time in generations, column 2) number of individuals to put in generation t and 3) number of individuals to pull in generation t. The last element of the list contains the final obtained frequency for the best fit.

Examples

```
opt_res <- optimize_adaptive(target_frequency = 0.99,
                              optimize_put = 1000,
                              num_generations = 20,
                              starting_freq = 0.2,
                              initial_population_size = 100)
opt_res$put
```

`optimize_static` *Optimize putting and/or pulling, where it is assumed that the same amount is applied per generation.*

Description

Optimize putting and/or pulling, where it is assumed that the same amount is applied per generation.

Usage

```
optimize_static(
  target_frequency = 0.99,
  initial_population_size = 400,
  reproduction_success_rate = 0.387,
  reproductive_risk = c(0.2, 0),
  K = 400,
  num_generations = 20,
  optimize_put = TRUE,
  optimize_pull = FALSE,
  starting_freq = 0.2,
  sd_starting_freq = 0.05,
  morgan = c(1),
  establishment_burnin = 30,
  num_replicates = 1,
  max_age = 6,
  mean_number_of_offspring = 6,
  sd_number_of_offspring = 1,
  genetic_model = "point",
  smin = 0.5,
  smax = 0.9,
  b = -2,
  p = 0.5,
  sex_ratio_put = 0.5,
  sex_ratio_pull = 0.5,
```

```
  sex_ratio_offspring = 0.5,
  ancestry_put = 1,
  ancestry_pull = 1,
  random_mating = FALSE,
  extra_pair_copulation = 0,
  verbose = FALSE,
  return_genetics = FALSE
)
```

Arguments

`target_frequency`	frequency to aim for
`initial_population_size`	population size at the start
`reproduction_success_rate`	frequency of females that yield offspring at the end of the breeding season (e.g. a fraction of 1 - reproduction_success_rate of females. This is a joint effect of breeding females getting killed (see `female_death_rate`) and other sources of failure to produce offspring. Other sources of failure are calculated from reproduction_success_rate and female_death_rate, such that the resulting reproduction failure rate = 1 - reproduction_success_rate / (1 - female breeding risk)
`reproductive_risk`	Additional death rate of males and females as a result of breeding (e.g. as a result of protecting the offspring against predators). Provide as a vector where the first index indicates the risk for females, the second the risk for males.
`K`	carrying capacity
`num_generations`	number of generations
`optimize_put`	When set to 0, FALSE or a negative number, it will not be optimized. When negative, the absolute value will be taken as a fixed contribution to each generation (but will not be optimized)
`optimize_pull`	When set to 0, FALSE or a negative number, it will not be optimized. When negative, the absolute value will be taken as a fixed contribution to each generation (but will not be optimized)
`starting_freq`	initial frequency of target ancestry in the population.
`sd_starting_freq`	variation in initial frequency of target ancestry.
`morgan`	a vector with the size of each chromosome in morgan, e.g. if a single chromosome is to be simulated a single number will suffice, but for two chromosomes of a size of 1 Morgan, a vector like: c(1, 1) will work.
`establishment_burnin`	number of generations before establishment
`num_replicates`	number of replicates
`max_age`	maximum age an individual can reach.

`mean_number_of_offspring`	mean number of offspring per female
`sd_number_of_offspring`	standard deviation of number of offspring per female (assuming the number of offspring is always 0 or larger).
`genetic_model`	The model can either use the point ancestry model ("point") of underlying genetics, which speeds up simulation considerably, but underestimates genetic variation. Alternatively, a more detailed genetic model is available, making use of the theory of junctions, this can be accessed using the option "junctions". Default is "point".
`smin`	minimum survival rate
`smax`	maximum survival rate
`b`	steepness of the survival rate. Negative values indicate a declining survival rate with increasing population size, positive values indicate an increasing survival rate with increasing population size.
`p`	Density at which the survival rate changes most relative. Expressed in N / K (e.g., for a value of 1.0, the survival rate changes most rapidly around N = K, for a value of 0.5, the survival rate changes most rapidly around N = 0.5K, etc).
`sex_ratio_put`	sex ratio of individuals that are added (if any) to the population. Sex ratio is expressed as males / (males + females), such that 0.5 indicates an even sex ratio, 0.9 indicates a male biased sex ratio and 0.1 indicates a female biased sex ratio.
`sex_ratio_pull`	sex ratio of individuals that are removed (if any) from the population. The sex ratio is expressed as males / (males + females), such that 0.5 indicates an even sex ratio, 0.9 indicates a male biased sex ratio and 0.1 indicates a female biased sex ratio.
`sex_ratio_offspring`	sex ratio of newly born offspring. The sex ratio is expressed as males / (males + females), such that 0.5 indicates an even sex ratio, 0.9 indicates a male biased sex ratio and 0.1 indicates a female biased sex ratio.
`ancestry_put`	Average ancestry of individuals being used for supplementation. If the target is high focal ancestry (e.g. aiming for focal ancestry of 1.0), ancestry put should reflect this and be set to 1.0 (which is the default value). When supplementing with non-pure individuals, this value can consequently be lowered.
`ancestry_pull`	Ancestry level below which individuals are allowed to be pulled - this can reduce the effective number of individuals pulled if none of the individuals in the population match this ancestry level. This can be used to selectively only remove those with low target ancestry.
`random_mating`	by default, simulations assume fixed pair bonding, e.g. each female mates with exactly one male (if available). Alternatively, if random_mating = TRUE, females will mate with a random male, introducing the possibility that some males mate multiple times.
`extra_pair_copulation`	probability of offspring to be fathered by another male. We assume that all offspring from one mother can have at most two fathers.
`verbose`	provides verbose output if TRUE.

`return_genetics`
: returns a tibble containing all local ancestry information for all individuals. This tibble contains the following informative columns: 1) time (the last generation), 2) replicate, 3) individual, 4) sex (0 = male, 1 = female), 5) Linkage Group (if use_simplified_model == FALSE), 6) chromosome (1 or 2, returning phased results), 7) position (if use_simplified_model == FALSE) and 8) local ancestry (0 or 1).

Value

list with five elements: 1) put: optimal number of individuals to put (0 if not estimated), 2) pull: optimal number of individuals to pull (0 if not estimated), 3) results tibble (see `simulate_policy()`), 4) curve tibble with three columns, indicating the realized number of put/pull per generation, with column 1) time in generations, column 2) number of individuals to put in generation t and 3) number of individuals to pull in generation t. The last element of the list contains the final obtained frequency for the best fit.

Examples

```
opt_res <- optimize_static(target_frequency = 0.99,
                           optimize_put = TRUE,
                           num_generations = 10,
                           starting_freq = 0.2,
                           initial_population_size = 100)
opt_res$put
```

`run_shiny_app` *runs shiny app locally*

Description

This function allows for local execution of the shiny app. Alternatively, an online version of this app can be found here.

Usage

```
run_shiny_app()
```

Value

No return value

`simulate_policy` *Simulate the effect of a restoration policy over time.*

Description

Using this function, the user can simulate the effect of an intended management policy on the genetic composition of a focal population. The population is assumed to have overlapping generations, and the user can specify two genetic models, either using a simplified average ancestry representation (genetic_model = "point"), or a more detailed model tracking explicit recombination among chromosomes, using genetic_model = "junctions".

Usage

```
simulate_policy(
  initial_population_size = 400,
  reproduction_success_rate = 0.387,
  reproductive_risk = c(0.2, 0),
  K = 400,
  num_generations = 20,
  pull = 0,
  put = 0,
  starting_freq = 0.5,
  sd_starting_freq = 0.05,
  morgan = c(1),
  max_age = 6,
  mean_number_of_offspring = 6,
  sd_number_of_offspring = 1,
  genetic_model = "point",
  establishment_burnin = 30,
  num_replicates = 1,
  seed = NULL,
  smin = 0.5,
  smax = 0.9,
  b = -2,
  p = 0.5,
  sex_ratio_put = 0.5,
  sex_ratio_pull = 0.5,
  sex_ratio_offspring = 0.5,
  ancestry_put = 1,
  ancestry_pull = 1,
  random_mating = FALSE,
  extra_pair_copulation = 0,
  verbose = FALSE,
  return_genetics = FALSE
)
```

Arguments

`initial_population_size`	population size at the start
`reproduction_success_rate`	frequency of females that yield offspring at the end of the breeding season (e.g. a fraction of 1 - reproduction_success_rate of females. This is a joint effect of breeding females getting killed (see `female_death_rate`) and other sources of failure to produce offspring. Other sources of failure are calculated from reproduction_success_rate and female_death_rate, such that the resulting reproduction failure rate = 1 - reproduction_success_rate / (1 - female breeding risk)
`reproductive_risk`	Additional death rate of males and females as a result of breeding (e.g. as a result of protecting the offspring against predators). Provide as a vector where the first index indicates the risk for females, the second the risk for males.
`K`	carrying capacity
`num_generations`	number of generations
`pull`	vector of the number of individuals pulled per year
`put`	vector of the number of individuals added per year
`starting_freq`	initial frequency of target ancestry in the population.
`sd_starting_freq`	variation in initial frequency of target ancestry.
`morgan`	a vector with the size of each chromosome in morgan, e.g. if a single chromosome is to be simulated a single number will suffice, but for two chromosomes of a size of 1 Morgan, a vector like: c(1, 1) will work.
`max_age`	maximum age an individual can reach.
`mean_number_of_offspring`	mean number of offspring per female
`sd_number_of_offspring`	standard deviation of number of offspring per female (assuming the number of offspring is always 0 or larger).
`genetic_model`	The model can either use the point ancestry model ("point") of underlying genetics, which speeds up simulation considerably, but underestimates genetic variation. Alternatively, a more detailed genetic model is available, making use of the theory of junctions, this can be accessed using the option "junctions". Default is "point".
`establishment_burnin`	number of generations before establishment
`num_replicates`	number of replicates
`seed`	random number seed, if left open, current time is used.
`smin`	minimum survival rate
`smax`	maximum survival rate

`b`	steepness of the survival rate. Negative values indicate a declining survival rate with increasing population size, positive values indicate an increasing survival rate with increasing population size.
`p`	Density at which the survival rate changes most relative. Expressed in N / K (e.g., for a value of 1.0, the survival rate changes most rapidly around N = K, for a value of 0.5, the survival rate changes most rapidly around N = 0.5K, etc).
`sex_ratio_put`	sex ratio of individuals that are added (if any) to the population. Sex ratio is expressed as males / (males + females), such that 0.5 indicates an even sex ratio, 0.9 indicates a male biased sex ratio and 0.1 indicates a female biased sex ratio.
`sex_ratio_pull`	sex ratio of individuals that are removed (if any) from the population. The sex ratio is expressed as males / (males + females), such that 0.5 indicates an even sex ratio, 0.9 indicates a male biased sex ratio and 0.1 indicates a female biased sex ratio.
`sex_ratio_offspring`	sex ratio of newly born offspring. The sex ratio is expressed as males / (males + females), such that 0.5 indicates an even sex ratio, 0.9 indicates a male biased sex ratio and 0.1 indicates a female biased sex ratio.
`ancestry_put`	Average ancestry of individuals being used for supplementation. If the target is high focal ancestry (e.g. aiming for focal ancestry of 1.0), ancestry put should reflect this and be set to 1.0 (which is the default value). When supplementing with non-pure individuals, this value can consequently be lowered.
`ancestry_pull`	Ancestry level below which individuals are allowed to be pulled - this can reduce the effective number of individuals pulled if none of the individuals in the population match this ancestry level. This can be used to selectively only remove those with low target ancestry.
`random_mating`	by default, simulations assume fixed pair bonding, e.g. each female mates with exactly one male (if available). Alternatively, if random_mating = TRUE, females will mate with a random male, introducing the possibility that some males mate multiple times.
`extra_pair_copulation`	probability of offspring to be fathered by another male. We assume that all offspring from one mother can have at most two fathers.
`verbose`	provides verbose output if TRUE.
`return_genetics`	returns a tibble containing all local ancestry information for all individuals. This tibble contains the following informative columns: 1) time (the last generation), 2) replicate, 3) individual, 4) sex (0 = male, 1 = female), 5) Linkage Group (if use_simplified_model == FALSE), 6) chromosome (1 or 2, returning phased results), 7) position (if use_simplified_model == FALSE) and 8) local ancestry (0 or 1).

Value

tibble with 8 columns: 1) replicate, 2) time (in generations), 3) average frequency of ancestry across all individuals 4) average frequency of ancestry across all males, 5) average frequency of ancestry across all females, 6) number of individuals, 7) number of males and 8) number of females if return_genetics = TRUE, the output is a list containing the above mentioned tibble, called 'results', and a second tibble called 'genetics', with the local ancestry in long format, split out per generation, replicate, individual, sex, linkage group and chromosome (1 or 2). Here, linkage group indicates the focal chromosome (linkage group), and 'chromosome' indicates which of the diploid pair of chromosomes is measured, allowing for phased output if required.

Examples

```
sim_pop <- simulate_policy(initial_population_size = 100,
                           num_generations = 20,
                           starting_freq = 0.2,
                           K = 400)
plot(sim_pop$results$num_individuals ~ sim_pop$results$t)
```

Index

optimize_adaptive, 2
optimize_static, 5

run_shiny_app, 8

simulate_policy, *4*, *8*, 9