

Package ‘dartR.sim’

July 22, 2025

Type Package

Title Computer Simulations of 'SNP' Data

Version 0.70

Description Allows to simulate SNP data using genlight objects. For example, it is straight forward to simulate a simple drift scenario with exchange of individuals between two populations or create a new genlight object based on allele frequencies of an existing genlight object.

Encoding UTF-8

Depends R (>= 3.5), adegenet (>= 2.0.0), dartR.base, dartR.data, ggplot2

Imports shiny, fields, utils, methods, stringi, stringr, data.table, Rcpp, shinyBS, shinyjs, shinythemes, shinyWidgets, hierfstat, reshape2

License GPL (>= 3)

RoxygenNote 7.2.3

NeedsCompilation no

Author Jose L. Mijangos [aut, cre],
Bernd Gruber [aut],
Arthur Georges [aut],
Carlo Pacioni [aut],
Peter J. Unmack [ctb],
Oliver Berry [ctb]

URL <https://green-striped-gecko.github.io/dartR/>,
<https://github.com/green-striped-gecko/dartR.sim>

BugReports <https://github.com/green-striped-gecko/dartR.sim/issues>

Maintainer Jose L. Mijangos <luis.mijangos@gmail.com>

Repository CRAN

Date/Publication 2023-11-20 19:30:02 UTC

Contents

gl.diagnostics.sim	2
gl.sim.create_dispersal	4
gl.sim.emigration	5
gl.sim.ind	6
gl.sim.mutate	7
gl.sim.offspring	8
gl.sim.WF.run	9
gl.sim.WF.table	11
interactive_reference	13
interactive_sim_run	14
Index	15

gl.diagnostics.sim	<i>Comparing simulations against theoretical expectations</i>
--------------------	---

Description

Comparing simulations against theoretical expectations

Usage

```
gl.diagnostics.sim(  
  x,  
  Ne,  
  iteration = 1,  
  pop_he = 1,  
  pops_fst = c(1, 2),  
  plot_theme = theme_dartR(),  
  plot.file = NULL,  
  plot.dir = NULL,  
  verbose = NULL  
)
```

Arguments

x	Output from function gl.sim.WF.run [required].
Ne	Effective population size to use as input to compare theoretical expectations [required].
iteration	Iteration number to analyse [default 1].
pop_he	Population name in which the rate of loss of heterozygosity is going to be compared against theoretical expectations [default 1].
pops_fst	Pair of populations in which FST is going to be compared against theoretical expectations [default c(1,2)].
plot_theme	User specified theme [default theme_dartR()].

plot.file	Name for the RDS binary file to save (base name only, exclude extension) [default NULL]
plot.dir	Directory in which to save files [default = working directory]
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log ; 3, progress and results summary; 5, full report [default NULL, unless specified using gl.set.verbosity].

Details

Two plots are presented comparing the simulations against theoretical expectations:

1. Expected heterozygosity under neutrality (Crow & Kimura, 1970, p. 329) is calculated as:

$$H_t = H_0(1 - (1/2N_e))^t$$
 where N_e is effective population size, H_0 is heterozygosity at generation 0 and t is the number of generations.
2. Expected F_{ST} under neutrality (Takahata, 1983) is calculated as:

$$F_{ST} = 1 / (4N_e m (n/(n-1))^2 + 1)$$
 where N_e is effective populations size of each individual subpopulation, m is dispersal rate and n the number of subpopulations (always 2).

Value

Returns plots comparing simulations against theoretical expectations

Author(s)

Custodian: Luis Mijangos – Post to <https://groups.google.com/d/forum/dartR>

References

- Crow JF, Kimura M. An introduction to population genetics theory. An introduction to population genetics theory. 1970.
- Takahata N. Gene identity and genetic differentiation of populations in the finite island model. Genetics. 1983;104(3):497-512.

See Also

[gl.filter.callrate](#)

Examples

```
ref_table <- gl.sim.WF.table(file_var=system.file('extdata',
'ref_variables.csv', package = 'dartR.data'),interactive_vars = FALSE)

res_sim <- gl.sim.WF.run(file_var = system.file('extdata',
'sim_variables.csv', package = 'dartR.data'),ref_table=ref_table,
interactive_vars = FALSE,number_pops_phase2=2,population_size_phase2="50 50")

res <- gl.diagnostics.sim(x=res_sim,Ne=50)
```

gl.sim.create_dispersal

Creates a dispersal file as input for the function gl.sim.WF.run

Description

This function writes a csv file called "dispersal_table.csv" which contains the dispersal variables for each pair of populations to be used as input for the function [gl.sim.WF.run](#).

The values of the variables can be modified using the columns "transfer_each_gen" and "number_transfers" of this file.

See documentation and tutorial for a complete description of the simulations. These documents can be accessed by typing in the R console: `browseVignettes(package="dartR")`

Usage

```
gl.sim.create_dispersal(
  number_pops,
  dispersal_type = "all_connected",
  number_transfers = 1,
  transfer_each_gen = 1,
  outpath = tempdir(),
  outfile = "dispersal_table.csv",
  verbose = NULL
)
```

Arguments

number_pops	Number of populations [required].
dispersal_type	One of: "all_connected", "circle" or "line" [default "all_connected"].
number_transfers	Number of dispersing individuals. This value can be . modified by hand after the file has been created [default 1].
transfer_each_gen	Interval of number of generations in which dispersal occur. This value can be modified by hand after the file has been created [default 1].
outpath	Path where to save the output file. Use outpath=getwd() or outpath='.' when calling this function to direct output files to your working directory [default tempdir(), mandated by CRAN].
outfile	File name of the output file [default 'dispersal_table.csv'].
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Value

A csv file containing the dispersal variables for each pair of populations to be used as input for the function [gl.sim.WF.run](#).

Author(s)

Custodian: Luis Mijangos – Post to <https://groups.google.com/d/forum/dartr>

See Also

[gl.sim.WF.run](#)

Other simulation functions: [gl.sim.WF.run\(\)](#), [gl.sim.WF.table\(\)](#)

Examples

```
gl.sim.create_dispersal(number_pops=10)
```

gl.sim.emigration	<i>Simulates emigration between populations</i>
-------------------	---

Description

A function that allows to exchange individuals of populations within a genlight object (=simulate emigration between populations).

There are two ways to specify emigration. If an emi.table is provided (a square matrix of dimension of the populations that specifies the emigration from column x to row y), then emigration is deterministic in terms of numbers of individuals as specified in the table. If perc.mig and emi.m are provided, then emigration is probabilistic. The number of emigrants is determined by the population size times the perc.mig and then the population where to migrate to is taken from the relative probability in the columns of the emi.m table.

Be aware if the diagonal is non zero then migration can occur into the same patch. So most often you want to set the diagonal of the emi.m matrix to zero. Which individuals is moved is random, but the order is in the order of populations. It is possible that an individual moves twice within an emigration call(as there is no check, so an individual moved from population 1 to 2 can move again from population 2 to 3).

Usage

```
gl.sim.emigration(x, perc.mig = NULL, emi.m = NULL, emi.table = NULL)
```

Arguments

x	A genlight or list of genlight objects [required].
perc.mig	Percentage of individuals that migrate (emigrates = nInd times perc.mig) [default NULL].
emi.m	Probabilistic emigration matrix (emigrate from=column to=row) [default NULL]
emi.table	If presented emi.m matrix is ignored. Deterministic emigration as specified in the matrix (a square matrix of dimension of the number of populations). e.g. an entry in the 'emi.table[2,1]<- 5' means that five individuals emigrate from population 1 to population 2 (from=columns and to=row) [default NULL].

Value

A list or a single [depends on the input] genlight object, where emigration between population has happened

Author(s)

Custodian: Bernd Gruber (Post to <https://groups.google.com/d/forum/dartr>)

Examples

```
x <- possums.gl
#one individual moves from every population to
#every other population
emi.tab <- matrix(1, nrow=nPop(x), ncol=nPop(x))
diag(emi.tab)<- 0
np <- gl.sim.emigration(x, emi.table=emi.tab)
np
```

gl.sim.ind

Simulates individuals based on allele frequencies

Description

This function simulates individuals based on the allele frequencies of a genlight object. The output is a genlight object with the same number of loci as the input genlight object.

Usage

```
gl.sim.ind(x, n = 50, popname = NULL)
```

Arguments

x	Name of the genlight object containing the SNP data [required].
n	Number of individuals that should be simulated [default 50].
popname	A population name for the simulated individuals [default NULL].

Details

The function can be used to simulate populations for sampling designs or for power analysis. Check the example below where the effect of drift is explored, by simply simulating several generation a genlight object and putting in the allele frequencies of the previous generation. The beauty of the function is, that it is lightning fast. Be aware this is a simulation and to avoid lengthy error checking the function crashes if there are loci that have just NAs. If such a case can occur during your simulation, those loci need to be removed, before the function is called.

Value

A genlight object with n individuals.

Author(s)

Bernd Gruber (bernd.gruber@canberra.edu.au)

Examples

```
glsim <- gl.sim.ind(testset.gl, n=10, popname='sims')
glsim
###Simulate drift over 10 generation
# assuming a bottleneck of only 10 individuals
# [ignoring effect of mating and mutation]
# Simulate 20 individuals with no structure and 50 SNP loci
founder <- glSim(n.ind = 20, n.snp.nonstruc = 50, ploidy=2)
#number of fixed loci in the first generation

res <- sum(colMeans(as.matrix(founder), na.rm=TRUE) %%2 ==0)
simgl <- founder
#49 generations of only 10 individuals
for (i in 2:50)
{
  simgl <- gl.sim.ind(simgl, n=10, popname='sims')
  res[i]<- sum(colMeans(as.matrix(simgl), na.rm=TRUE) %%2 ==0)
}
plot(1:50, res, type='b', xlab='generation', ylab='# fixed loci')
```

gl.sim.mutate

Simulates mutations within a genlight object

Description

This script is intended to be used within the simulation framework of dartR. It adds the ability to add a constant mutation rate across all loci. Only works currently for biallelic data sets (SNPs). Mutation rate is checking for all alleles position and mutations at loci with missing values are ignored and in principle 'double mutations' at the same loci can occur, but should be rare.

Usage

```
gl.sim.mutate(x, mut.rate = 1e-06)
```

Arguments

x	Name of the genlight object containing the SNP data [required].
mut.rate	Constant mutation rate over nInd*nLoc*2 possible locations [default 1e-6]

Value

Returns a genlight object with the applied mutations

Author(s)

Bernd Gruber (Post to <https://groups.google.com/d/forum/dartr>)

Examples

```
b2 <- gl.sim.mutate(bandicoot.gl, mut.rate=1e-4 )
#check the mutations that have occurred
table(as.matrix(bandicoot.gl), as.matrix(b2))
```

gl.sim.offspring

Simulates offspring based on alleles provided by parents

Description

This takes a population (or a single individual) of fathers (provided as a genlight object) and mother(s) and simulates offspring based on 'random' mating. It can be used to simulate population dynamics and check the effect of those dynamics and allele frequencies, number of alleles. Another application is to simulate relatedness of siblings and compare it to actual relatedness found in the population to determine kinship.

Usage

```
gl.sim.offspring(
  fathers,
  mothers,
  noffpermother,
  sexratio = 0.5,
  popname = "offspring",
  verbose = NULL
)
```

Arguments

fathers	Genlight object of potential fathers [required].
mothers	Genlight object of potential mothers simulated [required].
noffpermother	Number of offspring per mother [required].
sexratio	The sex ratio of simulated offspring (females / females +males, 1 equals 100 percent females) [default 0.5].
popname	population name of the returned genlight object [default offspring]
verbose	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using gl.set.verbosity].

Value

A genlight object with n individuals.

Author(s)

Bernd Gruber (Post to <https://groups.google.com/d/forum/dartR>)

Examples

```
#Simulate 10 potential fathers
gl.fathers <- glSim(10, 20, ploidy=2)
#Simulate 10 potential mothers
gl.mothers <- glSim(10, 20, ploidy=2)
res <- gl.sim.offspring(gl.fathers, gl.mothers, 2, sexratio=0.5)
```

gl.sim.WF.run

Runs Wright-Fisher simulations

Description

This function simulates populations made up of diploid organisms that reproduce in non-overlapping generations. Each individual has a pair of homologous chromosomes that contains interspersed selected and neutral loci. For the initial generation, the genotype for each individual's chromosomes is randomly drawn from distributions at linkage equilibrium and in Hardy-Weinberg equilibrium.

See documentation and tutorial for a complete description of the simulations. These documents can be accessed at <http://georges.biomatix.org/dartR>

Take into account that the simulations will take a little bit longer the first time you use the function gl.sim.WF.run() because C++ functions must be compiled.

Usage

```
gl.sim.WF.run(
  file_var,
  ref_table,
  x = NULL,
  file_dispersal = NULL,
  number_iterations = 1,
  every_gen = 10,
  sample_percent = 50,
  store_phase1 = FALSE,
  interactive_vars = TRUE,
  seed = NULL,
  verbose = NULL,
  ...
)
```

Arguments

file_var Path of the variables file 'sim_variables.csv' (see details) [required if interactive_vars = FALSE].

<code>ref_table</code>	Reference table created by the function <code>gl.sim.WF.table</code> [required].
<code>x</code>	Name of the <code>genlight</code> object containing the SNP data to extract values for some simulation variables (see details) [default NULL].
<code>file_dispersal</code>	Path of the file with the dispersal table created with the function <code>gl.sim.create_dispersal</code> [default NULL].
<code>number_iterations</code>	Number of iterations of the simulations [default 1].
<code>every_gen</code>	Generation interval at which simulations should be stored in a <code>genlight</code> object [default 10].
<code>sample_percent</code>	Percentage of individuals, from the total population, to sample and save in the <code>genlight</code> object every generation [default 50].
<code>store_phase1</code>	Whether to store simulations of phase 1 in <code>genlight</code> objects [default FALSE].
<code>interactive_vars</code>	Run a shiny app to input interactively the values of simulations variables [default TRUE].
<code>seed</code>	Set the seed for the simulations [default NULL].
<code>verbose</code>	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using <code>gl.set.verbosity</code>].
<code>...</code>	Any variable and its value can be added separately within the function, will be changed over the input value supplied by the csv file. See tutorial.

Details

Values for simulation variables can be submitted into the function interactively through a shiny app if `interactive_vars = TRUE`. Optionally, if `interactive_vars = FALSE`, values for variables can be submitted by using the csv file `'sim_variables.csv'` which can be found by typing in the R console: `system.file('extdata', 'sim_variables.csv', package = 'dartR.data')`.

The values of the variables can be modified using the third column ("value") of this file.

The output of the simulations can be analysed seemingly with other `dartR` functions.

If a `genlight` object is used as input for some of the simulation variables, this function access the information stored in the slots `x$position` and `x$chromosome`.

To show further information of the variables in interactive mode, it might be necessary to call first: `'library(shinyBS)'` for the information to be displayed.

The main characteristics of the simulations are:

- Simulations can be parameterised with real-life genetic characteristics such as the number, location, allele frequency and the distribution of fitness effects (selection coefficients and dominance) of loci under selection.
- Simulations can recreate specific life histories and demographics, such as source populations, dispersal rate, number of generations, founder individuals, effective population size and census population size.
- Each allele in each individual is an agent (i.e., each allele is explicitly simulated).
- Each locus can be customisable regarding its allele frequencies, selection coefficients, and dominance.

- The number of loci, individuals, and populations to be simulated is only limited by computing resources.
- Recombination is accurately modeled, and it is possible to use real recombination maps as input.
- The ratio between effective population size and census population size can be easily controlled.
- The output of the simulations are genlight objects for each generation or a subset of generations.
- Genlight objects can be used as input for some simulation variables.

Value

Returns genlight objects with simulated data.

Author(s)

Custodian: Luis Mijangos – Post to <https://groups.google.com/d/forum/dartr>

See Also

[gl.sim.WF.table](#)

Other simulation functions: [gl.sim.WF.table\(\)](#), [gl.sim.create_dispersal\(\)](#)

Examples

```
ref_table <- gl.sim.WF.table(file_var=system.file("extdata",
"ref_variables.csv", package = "dartR.data"),interactive_vars = FALSE)

res_sim <- gl.sim.WF.run(file_var = system.file("extdata",
"sim_variables.csv", package = "dartR.data"),ref_table=ref_table,
interactive_vars = FALSE)
```

gl.sim.WF.table

Creates the reference table for running gl.sim.WF.run

Description

This function creates a reference table to be used as input for the function [gl.sim.WF.run](#). The created table has eight columns with the following information for each locus to be simulated:

- q - initial frequency.
- h - dominance coefficient.
- s - selection coefficient.
- c - recombination rate.
- loc_bp - chromosome location in base pairs.

- loc_cM - chromosome location in centiMorgans.
- chr_name - chromosome name.
- type - SNP type.

The reference table can be further modified as required.

See documentation and tutorial for a complete description of the simulations. These documents can be accessed at <http://georges.biomatix.org/dartR>

Usage

```
gl.sim.WF.table(
  file_var,
  x = NULL,
  file_targets_sel = NULL,
  file_r_map = NULL,
  interactive_vars = TRUE,
  seed = NULL,
  verbose = NULL,
  ...
)
```

Arguments

<code>file_var</code>	Path of the variables file 'ref_variables.csv' (see details) [required if interactive_vars = FALSE].
<code>x</code>	Name of the genlight object containing the SNP data to extract values for some simulation variables (see details) [default NULL].
<code>file_targets_sel</code>	Path of the file with the targets for selection (see details) [default NULL].
<code>file_r_map</code>	Path of the file with the recombination map (see details) [default NULL].
<code>interactive_vars</code>	Run a shiny app to input interactively the values of simulation variables [default TRUE].
<code>seed</code>	Set the seed for the simulations [default NULL].
<code>verbose</code>	Verbosity: 0, silent or fatal errors; 1, begin and end; 2, progress log; 3, progress and results summary; 5, full report [default 2, unless specified using <code>gl.set.verbosity</code>].
<code>...</code>	Any variable and its value can be added separately within the function, will be changed over the input value supplied by the csv file. See tutorial.

Details

Values for the variables to create the reference table can be submitted into the function interactively through a Shiny app if `interactive_vars = TRUE`. Optionally, if `interactive_vars = FALSE`, values for variables can be submitted by using the csv file 'ref_variables.csv' which can be found by typing in the R console: `system.file('extdata', 'ref_variables.csv', package = 'dartR.data')`.

The values of the variables can be modified using the third column ("value") of this file.

If a `genlight` object is used as input for some of the simulation variables, this function access the information stored in the slots `x$position` and `x$chromosome`.

Examples of the format required for the recombination map file and the targets for selection file can be found by typing in the R console:

- `system.file('extdata', 'fly_recom_map.csv', package = 'dartR.data')`
- `system.file('extdata', 'fly_targets_of_selection.csv', package = 'dartR.data')`

To show further information of the variables in interactive mode, it might be necessary to call first: `'library(shinyBS)'` for the information to be displayed.

Value

Returns a list with the reference table used as input for the function `gl.sim.WF.run` and a table with the values variables used to create the reference table.

Author(s)

Custodian: Luis Mijangos – Post to <https://groups.google.com/d/forum/dartr>

See Also

`gl.sim.WF.run`

Other simulation functions: `gl.sim.WF.run()`, `gl.sim.create_dispersal()`

Examples

```
ref_table <- gl.sim.WF.table(file_var=system.file("extdata",
"ref_variables.csv", package = "dartR.data"),interactive_vars = FALSE)

res_sim <- gl.sim.WF.run(file_var = system.file("extdata",
"sim_variables.csv", package = "dartR.data"),ref_table=ref_table,
interactive_vars = FALSE)
```

interactive_reference *Shiny app for the input of the reference table for the simulations*

Description

Shiny app for the input of the reference table for the simulations

Usage

```
interactive_reference()
```

Author(s)

Custodian: Luis Mijangos – Post to <https://groups.google.com/d/forum/dartr>

interactive_sim_run	<i>Shiny app for the input of the simulations variables</i>
---------------------	---

Description

Shiny app for the input of the simulations variables

Usage

```
interactive_sim_run()
```

Author(s)

Custodian: Luis Mijangos – Post to <https://groups.google.com/d/forum/dartr>

Index

* simulation functions

- `gl.sim.create_dispersal`, [4](#)
 - `gl.sim.WF.run`, [9](#)
 - `gl.sim.WF.table`, [11](#)
- `gl.diagnostics.sim`, [2](#)
- `gl.filter.callrate`, [3](#)
- `gl.sim.create_dispersal`, [4](#), [10](#), [11](#), [13](#)
- `gl.sim.emigration`, [5](#)
- `gl.sim.ind`, [6](#)
- `gl.sim.mutate`, [7](#)
- `gl.sim.offspring`, [8](#)
- `gl.sim.WF.run`, [2](#), [4](#), [5](#), [9](#), [11](#), [13](#)
- `gl.sim.WF.table`, [5](#), [10](#), [11](#), [11](#)
- `interactive_reference`, [13](#)
- `interactive_sim_run`, [14](#)