Package ‘fwsim’

October 5, 2018

Version 0.3.4
Title Fisher-Wright Population Simulation
Author Mikkel Meyer Andersen and Poul Svante Eriksen
Maintainer Mikkel Meyer Andersen <mikl@math.aau.dk>
Description Simulates a population under the Fisher-Wright model (fixed or stochastic population size) with a one-step neutral mutation process (stepwise mutation model, logistic mutation model and exponential mutation model supported). The stochastic population sizes are random Poisson distributed and different kinds of population growth are supported. For the stepwise mutation model, it is possible to specify locus and direction specific mutation rate (in terms of upwards and downwards mutation rate). Intermediate generations can be saved in order to study e.g. drift.
License GPL-2
Depends R (>= 3.1.0)
LinkingTo Rcpp
Imports Rcpp (>= 0.11), methods
SystemRequirements C++11
NeedsCompilation yes
Repository CRAN
Date/Publication 2018-10-05 13:30:03 UTC

R topics documented:

  fwsim-package .................................................. 2
  fwsim .......................................................... 2
  init_mutmodel .................................................... 4
  mutmodel ....................................................... 6

Index 8
fwsim-package

Haplotype tools

Description
Tools for analysing haplotypes, e.g. population simulation.

Author(s)
Mikkel Meyer Andersen
Maintainer: Mikkel Meyer Andersen <mikl@math.aau.dk>

fwsim

Fisher-Wright Population Simulation

Description
This package provides tools to simulate a population under the Fisher-Wright model with a stepwise neutral mutation process on r loci, where mutations on loci happen independently. The population sizes are either fixed (traditional/original Fisher-Wright model) or random Poisson distributed with exponential growth supported. Intermediate generations can be saved in order to study e.g. drift.

For stochastic population sizes: Model described in detail at [http://arxiv.org/abs/1210.1773](http://arxiv.org/abs/1210.1773). Let $M$ be the population size at generation $i$ and $N$ the population size at generation $i + 1$. Then we assume that $N$ conditionally on $M$ is Poisson($\alpha M$) distributed for $\alpha > 0$ ($\alpha > 1$ gives expected growth and $0 < \alpha < 1$ gives expected decrease).

For each haplotype $x$ occurring $m$ times in the $i$'th generation, the number of children $n$ is Poisson($\alpha m$) distributed independently of other haplotypes. It then follows that the sum of the number of haplotypes follows a Poisson($\alpha M$) distribution (as just stated in the previous paragraph) and that $n$ conditionally on $N$ follows a Binomial($N, m/M$) as expected.

The mutation model can be e.g. the stepwise neutral mutation model. See `init_mutmodel` for details.

Usage

```r
fwsim(G, H0, N0, mutmodel, alpha = 1.0, SNP = FALSE,
       save_generations = NULL, progress = TRUE, trace = FALSE, ensure_children = FALSE, ...)

fwsim_fixed(G, H0, N0, mutmodel, SNP = FALSE,
             save_generations = NULL, progress = TRUE, trace = FALSE, ...)
```

## S3 method for class 'fwsim'
print(x, ...)

## S3 method for class 'fwsim'
summary(object, ...)

## S3 method for class 'fwsim'
plot(x, which = 1L, ...)
Arguments

- **G**: number of generations to evolve (integer, remember postfix L).
- **H0**: haplotypes of the initial population. Must be a vector or matrix (if more than one initial haplotype). The number of loci is the length or number of columns of H0.
- **N0**: count of the H0 haplotypes. The i’th element is the count of the haplotype H0[i, ]. sum(N0) is the size of initial population.
- **mutmodel**: a mutmodel object created with init_mutmodel. Alternatively, a numeric vector of length r of mutation probabilities (this will create a stepwise mutation model with r loci divide the mutation probabilities evenly between upwards and downwards mutation).
- **alpha**: vector of length 1 or G of growth factors (1 correspond to expected constant population size). If length 1, the value is reused in creating a vector of length G.
- **SNP**: to make alleles modulus 2 to immitate SNPs.
- **save_generations**: to save intermediate populations. NULL means that no intermediate population will be saved. Else, a vector of the generation numbers to save.
- **progress**: whether to print progress of the evolution.
- **trace**: whether to print trace of the evolution (more verbose than progress).
- **ensure_children**: Ensures that every generation has at least one child; implemented by getting Poisson(αM) + 1 children.
- **x**: A fwsim object.
- **object**: A fwsim object.
- **which**: A number specifying the plot (currently only 1: the actual population sizes vs the expected sizes).
- **...**: not used.

Value

A fwsim object with elements

- **pars**: the parameters used for the simulation
- **saved_populations**: a list of haplotypes in the intermediate populations
- **population**: haplotypes in the end population after G generations
- **pop_sizes**: the population size for each generation
- **expected_pop_sizes**: the expected population size for each generation

Author(s)

Mikkel Meyer Andersen <mikl@math.aau.dk> and Poul Svante Eriksen
Examples

```r
# SMM (stepwise mutation model) example
set.seed(1)
fit <- fwsim(G = 100L, H0 = c(0L, 0L, 0L), N0 = 10000L,
             mutmodel = c(Loc1 = 0.001, Loc2 = 0.002, Loc3 = 0.003))
summary(fit)
fit

# SMM (stepwise mutation model) example
H0 <- matrix(c(0L, 0L, 0L), 1L, 3L, byrow = TRUE)
mutmodel <- init_mutmodel(modeltype = 1L,
                           mutpars = matrix(c(c(0.003, 0.001), rep(0.004, 2), rep(0.001, 2)),
                                             ncol = 3,
                                             dimnames = list(NULL, c("DYS19", "DYS389I", "DYS391"))))
mutmodel

set.seed(1)
fit <- fwsim(G = 100L, H0 = H0, N0 = 10000L, mutmodel = mutmodel)

xtabs(N ~ DYS19 + DYS389I, fit$population)
plot(fit$fpar$G, fit$pop_sizes, type = "l",
     ylim = range(range(fit$pop_sizes), range(fit$expected_pop_sizes)))
points(fit$fpar$G, fit$expected_pop_sizes, type = "l", col = "red")

set.seed(1)
fit_fixed <- fwsim_fixed(G = 100L, H0 = H0, N0 = 10000L, mutmodel = mutmodel)

# LMM (logistic mutation model) example
mutpars.locus1 <- c(0.149, 2.08, 18.3, 0.149, 0.374, 27.4) # DYS19
mutpars.locus2 <- c(0.500, 1.18, 18.0, 0.500, 0.0183, 349) # DYS389I
mutpars.locus3 <- c(0.0163, 17.7, 11.1, 0.0163, 0.592, 14.1) # DYS391
mutpars <- matrix(mutpars.locus1, mutpars.locus2, mutpars.locus3, ncol = 3)
colnames(mutpars) <- c("DYS19", "DYS389I", "DYS391")
mutmodel <- init_mutmodel(modeltype = 2L, mutpars = mutpars)
mutmodel

set.seed(1)
H0_LMM <- matrix(c(15L, 13L, 10L), 1L, 3L, byrow = TRUE)
fit_LMM <- fwsim(G = 100L, H0 = H0_LMM, N0 = 10000L, mutmodel = mutmodel)
xtabs(N ~ DYS19 + DYS389I, fit_LM$population)
```

Description

Method to initialise a mutation model.
Usage

`init_mutmodel(modeltype = 1, mutpars = NULL, ...)`

## S3 method for class 'mutmodel'

`print(x, ...)`

Arguments

- **modeltype**
  - 1: SMM (traditional single-step mutation model).
  - 3: Exponential mutation model (unpublished).

- **mutpars**
  - A matrix specifying the mutation parameters for each locus. Rows are parameters and columns are loci. If a vector, the same values are used for all loci.

- **x**
  - A mutmodel object.

Details

Mutation parameters for each locus.

**Mutmodel 1 (SMM):** 2 parameters per locus

\[
P(i \rightarrow i-1) = \mu_d
\]

\[
P(i \rightarrow i+1) = \mu_u
\]

\[
P(i \rightarrow i) = 1 - P(i \rightarrow i-1) - P(i \rightarrow i+1)
\]

\[= 1 - \mu_d - \mu_u\]

\[
\text{mutpars}[1, \text{locus}] = \mu_d
\]

\[
\text{mutpars}[2, \text{locus}] = \mu_u
\]

**Mutmodel 2 (LMM):** 6 parameters per locus

\[
P(i \rightarrow i-1) = \gamma_d / (1 + \exp(\alpha_d + \beta_d \times i))
\]

\[
P(i \rightarrow i+1) = \gamma_u / (1 + \exp(\alpha_u + \beta_u \times i))
\]

\[
P(i \rightarrow i) = 1 - P(i \rightarrow i-1) - P(i \rightarrow i+1)
\]

\[= 1 - \gamma_d - \gamma_u\]

\[
\text{mutpars}[1, \text{locus}] = \gamma_d
\]

\[
\text{mutpars}[2, \text{locus}] = \alpha_d
\]

\[
\text{mutpars}[3, \text{locus}] = \beta_d
\]

\[
\text{mutpars}[4, \text{locus}] = \gamma_u
\]

\[
\text{mutpars}[5, \text{locus}] = \alpha_u
\]

\[
\text{mutpars}[6, \text{locus}] = \beta_u
\]

**Mutmodel 3 (EMM):** 4 parameters per locus

\[
P(i \rightarrow i-1) = 1/(1 + \exp(a + b \times i))
\]

\[
P(i \rightarrow i+1) = \exp(a + b \times i)/(1 + \exp(a + b \times i))
\]

\[
P(i \rightarrow i) = 1 - P(i \rightarrow i-1) - P(i \rightarrow i+1)
\]

\[= \exp(a + b \times i)/(1 + \exp(a + b \times i))\]
mutpars[1, locus]: a
mutpars[2, locus]: b
mutpars[3, locus]: alpha
mutpars[4, locus]: beta

Value

A mutmodel object (a list with entries modeltype and mutpars).

Examples

mutpars <- matrix(c(c(0.003, 0.001), rep(0.004, 2), rep(0.001, 2)), ncol = 3)
colnames(mutpars) <- c("DYS19", "DYS389I", "DYS391")
mutmodel <- init_mutmodel(modeltype = 1L, mutpars = mutpars)
mutmodel

mutpars.locus1 <- c(0.149, 2.08, 18.3, 0.149, 0.374, 27.4) # DYS19
mutpars.locus2 <- c(0.500, 1.18, 18.0, 0.500, 0.0183, 349) # DYS389I
mutpars.locus3 <- c(0.0163, 17.7, 11.1, 0.0163, 0.592, 14.1) # DYS391
mutpars <- matrix(c(mutpars.locus1, mutpars.locus2, mutpars.locus3), ncol = 3)
colnames(mutpars) <- c("DYS19", "DYS389I", "DYS391")
mutmodel <- init_mutmodel(modeltype = 2L, mutpars = mutpars)
mutmodel

Description

Functions for mutation model logic, e.g. probability of downwards and upwards mutations etc.

Usage

mutmodel_not_mut(mutmodel, locus, alleles)
mutmodel_dw_mut(mutmodel, locus, alleles)
mutmodel_up_mut(mutmodel, locus, alleles)
approx_stationary_dist(mutmodel, alleles)

Arguments

mutmodel a mutmodel object created with init_mutmodel.
locus the locus of interest (integer, remember postfix L).
alleles vector of integers (remember postfix L) of the alleles of interest.

Value

Mutation probabilities for locus locus in mutation model mutmodel at alleles alleles.
Examples

```r
mutpars.locus1 <- c(0.149, 2.08, 18.3, 0.149, 0.374, 27.4) # DYS19
mutpars.locus2 <- c(0.500, 1.18, 18.0, 0.500, 0.0183, 349) # DYS389I
mutpars.locus3 <- c(0.0163, 17.7, 11.1, 0.0163, 0.592, 14.1) # DYS391
mutpars <- matrix(c(mutpars.locus1, mutpars.locus2, mutpars.locus3), ncol = 3)
colnames(mutpars) <- c("DYS19", "DYS389I", "DYS391")
mutmodel <- init_mutmodel(modeltype = 2L, mutpars = mutpars)

mutmodel_not_mut(mutmodel, locus = 1L, alleles = 10L:20L)
mutmodel_dw_mut(mutmodel, locus = 1L, alleles = 10L:20L)
mutmodel_up_mut(mutmodel, locus = 1L, alleles = 10L:20L)

statdists <- approx_stationary_dist(mutmodel, alleles = 5L:20L)
bp <- barplot(statdists, beside = TRUE)
text(bp, 0.02, round(statdists, 1), cex = 1, pos = 3)
text(bp, 0, rep(rownames(statdists), ncol(mutmodel$mutpars)), cex = 1, pos = 3)

mutpars <- matrix(c(0.003, 0.001), rep(0.004, 2), rep(0.001, 2)), ncol = 3)
colnames(mutpars) <- c("DYS19", "DYS389I", "DYS391")
mutmodel <- init_mutmodel(modeltype = 1L, mutpars = mutpars)
mutmodel
statdists <- approx_stationary_dist(mutmodel, alleles = 5L:20L)
statdists

bp <- barplot(statdists, beside = TRUE)
text(bp, 0.02, round(statdists, 1), cex = 1, pos = 3)
text(bp, 0, rep(rownames(statdists), ncol(mutmodel$mutpars)), cex = 1, pos = 3)
```
Index

*Topic Fisher-Wright
  fwsim, 2
  mutmodel, 6
*Topic package
  fwsim-package, 2

approx_stationary_dist (mutmodel), 6
fwsim, 2
fwsim-package, 2
fwsim_fixed (fwsim), 2

init_mutmodel, 2, 3, 4, 6
mutmodel, 6
mutmodel_dw_mut (mutmodel), 6
mutmodel_not_mut (mutmodel), 6
mutmodel_up_mut (mutmodel), 6

plot.fwsim (fwsim), 2
print.fwsim (fwsim), 2
print.mutmodel (init_mutmodel), 4

summary.fwsim (fwsim), 2