Package ‘population’

March 16, 2022

Type  Package
Title  Models for Simulating Populations
Version  0.3
Date  2022-03-15
Author  Guillaume Chapron
Maintainer  Guillaume Chapron <gchapron@carnivoreconservation.org>
Description  Run population simulations using an Individual-Based Model (IBM) compiled in C.
License  GPL-3
Depends  parallel, abind
NeedsCompilation  yes
Repository  CRAN
Date/Publication  2022-03-16 13:30:06 UTC

R topics documented:

population-package ............................................... 1
get_cores .......................................................... 3
plot_projection ..................................................... 4
project ............................................................. 5
project_parallel ................................................... 6

Index  9

population-package  Population models

1
Description

A package to run population simulations using an Individual-Based Model compiled in C. The population model is a discrete, age-structured model and follows the formalizing of a post-breeding Leslie matrix model.

Version 0.1 proposes functions to run and plot population projections and includes demographic and environmental stochasticities. There is also the option to parallelize simulations (except on Windows).

Version 0.2 fixes a bug that generated wrong results at very large population sizes.

Details

Package: population
Type: Package
Version: 0.2
Date: 2018-02-05
License: GPL-3

Author(s)

Guillaume Chapron <gchapron@carnivoreconservation.org>

Examples

# Initial number of individuals
n0 <- 10
n1 <- 20
n2 <- 15
n3 <- 10
n4 <- 5

# Age-specific survival rates
s0 <- 0.5
s1 <- 0.6
s2 <- 0.7
s3 <- 0.8
s4 <- 0.9

# Age-specific number of offspring
b1 <- 0.5
b2 <- 0.8
b3 <- 1.8
b4 <- 1.8
b5 <- 1.1
# Project 10 years ahead repeated 10000 times
years <- 10
runs <- 10000

results <- project(
  years = years,
  runs = runs,
  initial_population = c(n0, n1, n2, n3, n4),
  survival = cbind(c(s0, s1, s2, s3, s4), 0.0), # no environmental stochasticity
  litter = cbind(c(b1, b2, b3, b4, b5), 0.0) # no environmental stochasticity
)

# Plot projection
plot_projection(results, "mean")

# Equivalent model with a post-breeding Leslie matrix
postM <- matrix(nrow=5, ncol=5, byrow=TRUE, data = c(
  s0*b1, s1*b2, s2*b3, s3*b4, s4*b5,
  s0, 0, 0, 0, 0,
  0, s1, 0, 0, 0,
  0, 0, s2, 0, 0,
  0, 0, 0, s3, 0
))

popvector <- c(n0, n1, n2, n3, n4)
N <- numeric(years)
N[1] <- sum(popvector)

for (i in 2:years) {
  popvector <- postM
  N[i] <- sum(popvector)
}

# Check we get similar results
lines(1:years, N, col="blue", lwd=2)

---

get_cores  

Get number of available cores

Description
Get number of available cores for parallel simulations. Non-Windows systems only.

Usage
get_cores(runs)

Arguments
runs Number of times (or Monte Carlo runs) to repeat the simulation.
Details

This function detects the number of cores (see 'detectCores' in package 'parallel') available and returns the largest possible number of cores being an integer divider of the number of runs. On multi-core machines at least one core is not used for the simulation.

Value

get_cores()

Examples

get_cores(2)
get_cores(1000)

plot_projection

Plot population projections

Description

Plot population projections.

Usage

plot_projection(projection, kind)

Arguments

projection A list obtained after running functions 'project' or 'project_cores'.
kind (optional) A string indicating which quantity should be plotted ("median" or "mean"). If missing, all projections are shown.

Details

Plot all population projections or the median or mean with 95% confidence interval. Only total population sizes are displayed.

Value

No returned value, plot created
**Examples**

```r
years <- 10
runs <- 100

init.pop <- c(30, 20, 15, 12, 10, 9, 8, 7, 6, 5)

surv.md <- c(0.5, 0.7, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9)
surv.sd <- c(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)
surv.msd <- cbind(surv.md, surv.sd)

litter.md <- c(0.2, 1.1, 2.8, 2.8, 2.8, 2.8, 2.8, 2.8, 1.8, 0.2)
litter.sd <- c(0.1, 0.2, 0.15, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)
litter.msd <- cbind(litter.md, litter.sd)

nclass <- 4 # vary number of classes

projection <- project(
  years = years,
  runs = runs,
  initial_population = init.pop[1:nclass],
  survival = surv.msd[1:nclass,]
)

plot_projection(projection)
plot_projection(projection, kind="median")
```

---

**Description**

Run stochastic population projections.

**Usage**

```r
project(years, runs, initial_population, survival, litter, seed)
```

**Arguments**

- `years` Number of years to project the population.
- `runs` Number of times (or Monte Carlo runs) to project the population.
- `initial_population` Vector of initial number of individuals for each class. This vector must contain only positive integers.
- `survival` Matrix of survival for each class, with nrow = number of classes and ncol = 2. The first column is the median value of the survival of each class and the second column is the standard deviation of the survival of each class.
litter Matrix of litter size for each class, with nrow = number of classes and ncol = 2. The first column is the median value of the litter size of each class. The second column is the standard deviation of the litter size of each class.

seed (optional) seed for the R random number generator. If missing, the seed is set to 1.

Details
Run stochastic population projections with an Individual-Based Model (IBM) compiled in C.

Value
runs a 3-dimensional array of numbers of individuals with dimension c(years, classes, runs)

Examples
years <- 10
runs <- 100

init.pop <- c(30, 20, 15, 12, 10, 9, 8, 7, 6, 5)
surv.md <- c(0.5, 0.7, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9)
surv.sd <- c(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)
surv.msd <- cbind(surv.md, surv.sd)
litter.md <- c(0.2, 1.1, 2.8, 2.8, 2.8, 2.8, 2.8, 2.8, 1.8, 0.2)
litter.sd <- c(0.1, 0.2, 0.15, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)
litter.msd <- cbind(litter.md, litter.sd)
nclass <- 4 # vary number of classes

projection <- project(
  years = years,
  runs = runs,
  initial_population = init.pop[1:nclass],
  survival = surv.msd[1:nclass,],
  litter = litter.msd[1:nclass,]
)

project_parallel Parallel population projections

Description
Run parallel stochastic population projections. Non-Windows systems only.

Usage
project_parallel(years, runs, initial_population, survival, litter, cores)
Arguments

years
Number of years to project the population.

runs
Number of times (or Monte Carlo runs) to project the population.

initial_population
Vector of initial number of individuals for each class. This vector must contain only positive integers.

survival
Matrix of survival for each class, with nrow = number of classes and ncol = 2. The first column is the median value of the survival of each class. The second column is the standard deviation of the survival of each class.

litter
Matrix of litter size for each class, with nrow = number of classes and ncol = 2. The first column is the median value of the litter size of each class. The second column is the standard deviation of the litter size of each class.

cores
(optional) number of cores to use for the parallel projections. If missing, it is set to the value returned by get_cores().

Details

Run parallel stochastic population projections with an Individual-Based Model (IBM) compiled in C.

Value

runs
a 3-dimensional array of numbers of individuals with dimension c(years, classes, runs)

Examples

years <- 10
runs <- 100
init.pop <- c(30, 20, 15, 12, 10, 9, 8, 7, 6, 5)
surv.md <- c(0.5, 0.7, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9)
surv.sd <- c(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)
surv.msd <- cbind(surv.md, surv.sd)
litter.md <- c(0.2, 1.1, 2.8, 2.8, 2.8, 2.8, 2.8, 1.8, 0.2)
litter.sd <- c(0.1, 0.2, 0.15, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)
litter.msd <- cbind(litter.md, litter.sd)
nclass <- 4 # vary number of classes

# with 2 cores
projection <- project_parallel(
years = years,
runs = runs,
initial_population = init.pop[1:nclass],
survival = surv.msd[1:nclass,],
litter = litter.msd[1:nclass],
cores = 2)

# with 2 cores
projection <- project_parallel(
years = years,
runs = runs,
initial_population = init.pop[1:nclass],
survival = surv.msd[1:nclass,],
litter = litter.msd[1:nclass,],
cores = 2)
cores = 2
)

# with all possible cores (not run)
# projection <- project_parallel(
#  years = years,
#  runs = runs,
#  initial_population = init.pop[1:nclass],
#  survival = surv.msd[1:nclass],
#  litter = litter.msd[1:nclass],
#  )
Index

C_montecarlo (project), 5
get_cores, 3
plot_projection, 4
population (population-package), 1
population-package, 1
project, 5
project_parallel, 6