Package 'SharkDemography'

November 21, 2023

Type Package

Title Shark Demographic Analyses Using Leslie Matrix Models

Version 1.1.0

Maintainer Jonathan Smart <jonsmartphd@gmail.com>

Description Run Leslie Matrix models using Monte Carlo simulations for any specified shark species. This package was developed during the publication of Smart, JJ, White, WT, Baje, L, et al. (2020) ``Can multi-species shark longline fisheries be managed sustainably using size limits? Theoretically, yes. Realistically, no".J Appl Ecol. 2020; 57; 1847–1860. <doi:10.1111/1365-2664.13659>.

Encoding UTF-8

LazyData true

Depends R (>= 3.3)

Imports MASS, interp, dplyr, tidyr, readr, popbio, iterators, parallel, doFuture, doParallel, foreach, magrittr

RoxygenNote 7.2.3

License GPL (>= 3)

Suggests knitr, rmarkdown

URL https://github.com/jonathansmart/SharkDemography

BugReports https://github.com/jonathansmart/SharkDemography/issues

NeedsCompilation no

Author Jonathan Smart [aut, cre, ctb, cph] (<https://orcid.org/0000-0003-2070-3208>)

Repository CRAN

Date/Publication 2023-11-21 18:10:05 UTC

R topics documented:

Calculate_demography 2
create_data_input 4

Estimate_eigenvectors 5
Estimate_mortality_dists 6
SharkDemography. 7
Silky_data 7
Simulate_AAFC 8
Simulate_AALC 9
Simulate_demography 10
Simulate_F_critical 11
Simulate_harvest_slots 12

Index **14**

`Calculate_demography` *The base function for calculating demographic traits using a Leslie matrix method*

Description

Using provided life history estimates from various sources, a demographic analysis is performed using Leslie matrix methods. This function can be run as stand alond and will given the outputs of a stochastic matrix model given the provided life history parameters and their associated error. However, the main purpose of this function is to be used in other simulation functions provided in this package. In each use this function will calculate:

lambda The finite rate of population growth

R0 The reproductive value of the population

G The generation length of the population

elast.fecund The mean elasticity of the fecundity elements

elast.juv.survival The mean elasticity of the juvenile survivoship elements

juv.ratio The ratio of fecundity to juvenile elasticities

adult.ratio The ratio of fecundity to adult elasticities

M.estimator The natural mortality estimator randomly used in this analysis

Usage

```
Calculate_demography(
  data,
  AAFC = NULL,
  F. = 0,
  AALC = NULL,
  M.estimators = NULL
)
```

Arguments

data	A multi-level list of the class 'Demography.inputs' produced from the 'create_data_input' function and then manually completed.
AAFC	Age-at-first-capture which can be specified by the user or automated in the 'Simulate_AAFC' function. Must be an integer age which can include zero to indicate the availability of the population to capture from birth.
F.	The instantaneous rate of fishing mortality 'F'. This will be applied to all ages available to capture as defined by either the AALC or AAFC arguments.
AALC	Age-at-last-capture which can be specified by the user or automated in the 'Simulate_AALC' function. Must be an integer age which can include zero.
M.estimators	Any specific natural mortality estimators to be included in the analysis. Only one will be used in each run which is randomly selected. In the simulation functions these will be varied among simulations. Must be a single estimator or a vector of estimators. These can include: "Pet.Wro","Jensen.mat","Chen.Yuan","Then_hoenig","Then_pauly", " Jensen.mat","Charnov" or "Chen.Want". If none are specified then all applicable estimators could be chosen.

Value

A list of parameters including:

lambda The finite rate of population growth

R0 The reproductive value of the population

G The generation length of the population

elast.fecund The mean elasticity of the fecundity elements

elast.juv.survival The mean elasticity of the juvenile survivoship elements

juv.ratio The ratio of fecundity to juvenile elasticities

adult.ratio The ratio of fecundity to adult elasticities

M.estimator The natural mortality estimator randomly used in this analysis

Examples

```
# load Silky shark data produced by create_data_input()
# Type `?create_data_input()` for details
data("Silky_data")

# run a single Leslie Matrix analysis with random draws from biological data
# distributions. Use `Simulate_demography()` to run Monte Carlo Simulations
# using this function

Calculate_demography(Silky_data)
```

`create_data_input` *Create empty template for demography life history data*

Description

The demography functions in this package require a large amount of detailed life history information. This is provided to these functions as a multi-level list of various life history parameters with the class 'Demography.inputs'. This function creates the template for this input data which can be filled in after it is created.

Usage

```
create_data_input(maturity.type, t0 = FALSE)
```

Arguments

`maturity.type`	The type of maturity estimates from the source life history study. must be one of 'logistic - int/slope', 'logistic - a50/a95', 'normal' or 'uniform'.
`t0`	Logical argument regarding whether the growth models included "t0" or "L0" as as a parameter. Default is 'FALSE'

Value

A multi-level list of the class 'Demography.inputs'

Examples

```
######-----------
# Example code for Silky sharks
######-----------

silky_data <- create_data_input("logistic - int/slope", t0 = FALSE)
# Add growth data
silky_data$`growth`$model.type <- "logistic"
silky_data$growth$pars$Linf <- 268
silky_data$growth$pars$k <- 0.14
silky_data$growth$pars$L0 <- 82.7
silky_data$growth$se$Linf.se <- 5.8
silky_data$growth$se$k.se <- 0.006
silky_data$growth$se$L0.se <- 1.6
silky_data$growth$corr.matrix <- matrix(ncol = 3, nrow = 3,
                              dimnames = list(c("Linf", "k", "L0"),c("Linf", "k", "L0")),
                                        data = c(1.0000000, -0.907188, 0.6233407,
                                                -0.9071881,  1.0000000, -0.8572509,
                                                0.6233407,-0.857250, 1.0000000))
# Add maturity data
silky_data$maturity$pars$intercept <- -15.90
silky_data$maturity$pars$slope <- 1.14
silky_data$maturity$se$intercept.se <- 2.78258
```

```
silky_data$maturity$se$slope.se <- 0.1971363
silky_data$maturity$corr.matrix <- matrix(ncol = 2, nrow = 2,
                                          dimnames = list(c("Intercept", "slope")
                                          ,c("Intercept", "slope")),
                                          data = c(1.0000000, -0.9922574,
                                                   -0.9922574, 1.0000000))
# max age lower bound
silky_data$max.age$min <- 28

# Add fecundity info
silky_data$litter.size$mean <- 10
silky_data$litter.size$se <- 3
silky_data$gest.period <- 1
silky_data$repro.cycle <- 2

# Add TL conversions (if available and required)
silky_data$Lt.type <- "TL"
silky_data$Lt.to.Wt$model.type <- "PCL"
silky_data$Lt.to.Wt$pars$a <- 2.73e-5
silky_data$Lt.to.Wt$pars$b <- 2.86

silky_data$convert.TL$model.type <- "PCL"
silky_data$convert.TL$pars$a <- 2.08
silky_data$convert.TL$pars$b <- 1.32
```

`Estimate_eigenvectors` *Estimate the left and right eignvectors (Reproductive value: v and Stable age distribution: w)*

Description

This function performs a Monte Carlo simulation analysis which determines the distributions of the left and right eigenvector. To facilitate this, maximum age is fixed as the minimum value + 20% for the species.

Usage

```
Estimate_eigenvectors(n = 1000, data, M.estimators = NULL)
```

Arguments

`n`	The number of simulations to be run
`data`	A multi-level list of the class 'Demography.inputs' produced from the 'create_data_input' function and then manually completed.
`M.estimators`	Any specific natural mortality estimators to be included in the analysis. Must be a single estimator or a vector of estimators. These can include: "Pet.Wro", "Jensen.mat","Chen.Yuan","Then_hoenig","Then_pauly", "Jensen.mat","Charnov" or "Chen.Want". If none are specified then all applicable estimators could be chosen.

Value

A dataframe with the mean and 95% quantiles for each eigenvector for each age class

Examples

```
# load Silky shark data produced by create_data_input()
# Type `?create_data_input()` for details
data("Silky_data")

# Run function to get Eigenvectors from  Monte Carlo simulations for
# all available natural mortality estimators. Set n = at least 1000 for full
# analysis but use n = 100 for testing

Estimate_eigenvectors(n = 100, Silky_data)
```

Estimate_mortality_dists

Estimate the range of mortality estimates produced by the different estimators

---

Description

This function performs a Monte Carlo simulation analysis which determines the distributions of natural mortality (M) produced by each estimator.

Usage

```
Estimate_mortality_dists(n = 1000, data, M.estimators = NULL)
```

Arguments

n	The number of simulations to be run
data	A multi-level list of the class 'Demography.inputs' produced from the 'create_data_input' function and then manually completed.
M.estimators	Any specific natural mortality estimators to be included in the analysis. Must be a single estimator or a vector of estimators. These can include: "Pet.Wro", "Jensen.mat","Chen.Yuan","Then_hoenig","Then_pauly", "Jensen.mat","Charnov" or "Chen.Want". If none are specified then all applicable estimators could be chosen.

Value

A list which includes a data.frame of the distributions age invariant estimators summarised as the mean and 95% quantiles and dataframes of any age dependent estimators with the mean and 95% quantiles for each age class

Examples

```
# load Silky shark data produced by create_data_input()
# Type `?create_data_input()` for details
data("Silky_data")

# Run function to get natural mortality distributions from  Monte Carlo
# simulations for all available natural mortality estimators.
# Set n = at least 1000 for full analysis but use n = 100 for testing

Estimate_mortality_dists(n = 100, Silky_data)
```

SharkDemography. *SharkDemography.*

Description

A package to conduct Shark demographic analyses using Leslie matrix models.

Author(s)

Jonathan Smart

Silky_data *Example life history data for Silky Sharks*

Description

An example dataset of a completed set of life history data for Silky Sharks in a 'create_data_input()' template.

Usage

```
data(Silky_data)
```

Format

A list of 10 with class "Demography.inputs"

`Simulate_AAFC` *Monte Carlo simulations of Leslie matrix models under varying levels of F and AAFC*

Description

This is a wrapper function for 'Calculate_demography' which runs this function the specified number of times using a range of F and Age-at-first-capture (AAFC) values

Usage

```
Simulate_AAFC(n = 1000, .data, M.estimators = NULL, max.AAFC = 15, n_cores = 1)
```

Arguments

`n`	The number of simulations to be run. 1000 is recomended but smaller numbers should be run when testing to avoid long run times.
`.data`	A multi-level list of the class 'Demography.inputs' produced from the 'create_data_input' function and then manually completed.
`M.estimators`	Any specific natural mortality estimators to be included in the analysis. Only one will be used in each run which is randomly selected. Must be a single estimator or a vector of estimators. These can include: "Pet.Wro","Jensen.mat","Chen.Yuan", "Then_hoenig","Then_pauly", "Jensen.mat","Charnov" or "Chen.Want". If none are specified then all applicable estimators could be chosen.
`max.AAFC`	The maximum Age class to be run in the analyses. This does not need to be the maximum age for the population and keeping this number reasonable reduces run-time.
`n_cores`	The number of cores to be used for parallel processing. It should be 1 core less than the maximum number available.

Value

A list with two data.frames. The first provides Fcritical values for each age class. This is the value of F where the population growth rate is stable (lambda = 1). The second dataframe is the mean lambda produced for each combination of AAFC and F

Examples

```
# load Silky shark data produced by create_data_input()
# Type `?create_data_input()` for details
data("Silky_data")

# Run function to get conduct an age-at-first-capture (AAFC) analysis using
# Monte Carlo Simulations using for all available natural mortality estimators.
# Set n = at least 1000 for full analysis but use n = 10 for testing given long run times
```

```
Simulate_AAFC(n = 10, Silky_data, n_cores = 1)
```

`Simulate_AALC` *Monte Carlo simulations of Leslie matrix models under varying levels of F and AALC*

Description

This is a wrapper function for 'Calculate_demography' which runs this function the specified number of times using a range of F and Age-at-last-capture (AALC) values

Usage

```
Simulate_AALC(n = 1000, data, M.estimators = NULL, min.AALC = 15, n_cores = 1)
```

Arguments

`n`	The number of simulations to be run. 1000 is recomended but smaller numbers should be run when testing to avoid long run times.
`data`	A multi-level list of the class 'Demography.inputs' produced from the 'create_data_input' function and then manually completed.
`M.estimators`	Any specific natural mortality estimators to be included in the analysis. Only one will be used in each run which is randomly selected. Must be a single estimator or a vector of estimators. These can include: "Pet.Wro","Jensen.mat","Chen.Yuan", "Then_hoenig","Then_pauly", "Jensen.mat","Charnov" or "Chen.Want". If none are specified then all applicable estimators could be chosen.
`min.AALC`	The last Age class to be run in the analyses. This does not need to be the maximum age for the population and keeping this number reasonable reduces runtime.
`n_cores`	The number of cores to be used for parallel processing. It should be 1 core less than the maximum number available.

Value

A list with two data.frames. The first provides Fcritical values for each age class. This is the value of F where the population growth rate is stable (lambda = 1). The second dataframe is the mean lambda produced for each combination of AALC and F

Examples

```
# load Silky shark data produced by create_data_input()
# Type `?create_data_input()` for details
data("Silky_data")

# Run function to get conduct an age-at-last-capture (AALC) analysis using
```

```
# Monte Carlo Simulations using for all available natural mortality estimators.
# Set n = at least 1000 for full analysis but use n = 10 for testing given long run times

Simulate_AALC(n = 10, Silky_data, n_cores = 1)
```

`Simulate_demography` *Monte Carlo simulations of Leslie matrix models*

Description

This is a wrapper function for 'Calculate_demography' which runs this function the specified number of times.

Usage

```
Simulate_demography(
  n,
  data,
  AALC = NULL,
  AAFC = NULL,
  F. = 0,
  M.estimators = NULL,
  Verbatim = TRUE
)
```

Arguments

`n`	The number of specified Monte Carlo simulations to run
`data`	A multi-level list of the class 'Demography.inputs' produced from the 'create_data_input' function and then manually completed.
`AALC`	Age-at-last-capture which can be specified by the user. Must be an integer age which can include zero.
`AAFC`	Age-at-first-capture which can be specified by the user.Must be an integer age which can include zero to indicate the availability of the population to capture from birth.
`F.`	The instantaneous rate of fishing mortality 'F'. This will be applied to all ages available to capture as defined by either the AALC or AAFC arguments.
`M.estimators`	Any specific natural mortality estimators to be included in the analysis. Only one will be used in each run which is randomly selected. Must be a single estimator or a vector of estimators. These can include: "Pet.Wro","Jensen.mat","Chen.Yuan", "Then_hoenig","Then_pauly", "Jensen.mat","Charnov" or "Chen.Want". If none are specified then all applicable estimators could be chosen.
`Verbatim`	Print summary results to screen if TRUE. When FALSE, the progress bar is also disabled.

Value

A list with two data.frames. The first is the summary of the Monte Carlo simulations for all parameters calculated by the 'Calculate_demography' function with mean and 95% quantiles. The second is all of the results for each parameter from individual simulations so that their distributions can be interrogated further.

Examples

```
# load Silky shark data produced by create_data_input()
# Type `?create_data_input()` for details
data("Silky_data")

# Run function to get conduct Monte Carlo Simulations using
# `Calculate_demography()`  for all available natural mortality estimators.
# Set n = at least 1000 for full analysis but use n = 100 for testing

Simulate_demography(n = 100, Silky_data)
```

Simulate_F_critical *Estimate F critical through simulations*

Description

This is a wrapper function for 'Calculate_demography' which runs this function the specified number of times using a range of F values accross the enture age range of the population. This determines the rate of population increase at each increment of F.

Usage

```
Simulate_F_critical(
  n = 1000,
  .data,
  M.estimators = NULL,
  max.F = 0.3,
  n_cores = 1
)
```

Arguments

n	The number of simulations to be run. 1000 is recomended but smaller numbers should be run when testing to avoid long run times.
.data	A multi-level list of the class 'Demography.inputs' produced from the 'create_data_input' function and then manually completed.
M.estimators	Any specific natural mortality estimators to be included in the analysis. Only one will be used in each run which is randomly selected. Must be a single estimator or a vector of estimators. These can include: "Pet.Wro","Jensen.mat","Chen.Yuan", "Then_hoenig","Then_pauly", "Jensen.mat","Charnov" or "Chen.Want". If none are specified then all applicable estimators could be chosen.

max.F	The maximum value of F for simulations
n_cores	The number of cores to be used for parallel processing. It should be 1 core less than the maximum number available.

Value

A list with two data.frames. The first provides the mean F critical with 95% confidence intervals. The second dataframe provides the rate of increase for each increment of F.

Examples

```
# load Silky shark data produced by create_data_input()
# Type `?create_data_input()` for details
data("Silky_data")

# Run function to get conduct an F critical analysis using
# Monte Carlo Simulations using for all available natural mortality estimators.
# Set n = at least 1000 for full analysis but use n = 10 for testing given long run times

Simulate_F_critical(n = 10, Silky_data, n_cores = 1)
```

Simulate_harvest_slots

Simulate_harvest_slots

Description

This is a wrapper function for 'Calculate_demography' which runs this function using different values of AAFC, AALC and F to simulate Havest Slot options. The Fcritical is retuned for each simulation to show the max level of F needed to sustain a stable population.

Usage

```
Simulate_harvest_slots(
  n,
  data,
  M.estimators = NULL,
  Age.mid.point = NULL,
  HS.width = NULL,
  max.F = 1
)
```

Arguments

n	number of iterations for each combination of Age.mid.point, HS.width and F.
data	A multi-level list of the class 'Demography.inputs' produced from the 'create_data_input' function and then manually completed.

M.estimators	Any specific natural mortality estimators to be included in the analysis. Only one will be used in each run which is randomly selected. Must be a single estimator or a vector of estimators. These can include: "Pet.Wro","Jensen.mat","Chen.Yuan", "Then_hoenig","Then_pauly", "Jensen.mat","Charnov" or "Chen.Want". If none are specified then all applicable estimators could be chosen.
Age.mid.point	A vector of ages to be used in the simulation. Each age is used as a mid point and will have HS.Width subtracted and added to it to determine AAFC and AALC, respectively.
HS.width	A vector of widths for the Harvest slots in years. Widths are subtracted and added to mid points to determine the AAFC and AALC in each sim.
max.F	The maximum value of F for simulations

Value

A data.frame with three columns: MinAge, MaxAge and 'F.'. These represent the age at the start of a harvest slot, the age at the end of the harvest slot and the F for that harvest slot.

Examples

```
# load Silky shark data produced by create_data_input()
# Type `?create_data_input()` for details
data("Silky_data")

# Run function to get conduct an F critical analysis for different harvest slots using
# Monte Carlo Simulations using for all available natural mortality estimators.
# Set n = at least 1000 for full analysis but use n = 10 for testing given long run times

Simulate_harvest_slots(n = 10, Silky_data,Age.mid.point = 0:28, HS.width = 0:8)
```

Index

∗ **datasets**
Silky_data, 7

Calculate_demography, 2
create_data_input, 4

Estimate_eigenvectors, 5
Estimate_mortality_dists, 6

SharkDemography., 7
Silky_data, 7
Simulate_AAFC, 8
Simulate_AALC, 9
Simulate_demography, 10
Simulate_F_critical, 11
Simulate_harvest_slots, 12