Package ‘mpmsim’

June 5, 2025

Title Simulation of Matrix Population Models with Defined Life History Characteristics

Version 3.2.1

Description Allows users to simulate matrix population models with particular characteristics based on aspects of life history such as mortality trajectories and fertility trajectories. Also allows the exploration of sampling error due to small sample size.

URL `https://jonesor.github.io/mpmsim/`

BugReports `https://github.com/jonesor/mpmsim/issues/`

Imports dplyr, ggplot2, grDevices, popbio, popdemo, Rage, Rcompadre, reshape, stats

Suggests knitr, patchwork, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.2

NeedsCompilation no

Author Owen Jones [aut, cre] (ORCID: <`https://orcid.org/0000-0001-5720-4686`>)

Maintainer Owen Jones <`jones@biology.sdu.dk`>

Repository CRAN

Date/Publication 2025-06-05 04:20:02 UTC

Contents

`add_mpm_error` *Add sampling error to matrix population models (MPMs) based on expected values of transition rates and sample sizes*

Description

Produces a list of matrix population models based on expected values in the transition matrix and sample size. The expected values are provided in lists of two submatrices: `mat_U` for the growth/development and survival transitions and `mat_F` for the reproductive transitions. The output `mat_U` values are simulated based on expected probabilities, assuming a binomial process with a sample size defined by `sample_size`. The output `mat_F` values are simulated using a Poisson process with a sample size defined by `sample_size`.Thus users can expect that large sample sizes will result in simulated matrices that match closely with the expectations, while simulated matrices with small sample sizes will be more variable.

Usage

```
add_mpm_error(mat_U, mat_F, sample_size, split = TRUE, by_type = TRUE)
```

Arguments

`mat_U`	A list of U submatrices, or a single U submatrix.
`mat_F`	A list of F submatrices, or a single F submatrix.
`sample_size`	either (1) a single matrix of sample sizes for each element of every MPM, (2) a list of two named matrices ("`mat_F_ss`", "`mat_U_ss`") containing sample sizes for the survival and reproductive output submatrices of every MPM or (3) a single value applied to the every element of every matrix.
`split`	logical, whether to split the output into survival and reproductive output matrices or not. Defaults to `TRUE`.
`by_type`	A logical indicating whether the matrices should be returned in a list by type (A, U, F, C). If split is `FALSE`, then `by_type` must also be `FALSE`. Defaults to `TRUE`.

Details

if any `sample_size` input is 0, it is assumed that the estimate for the element(s) concerned is known without error.

Value

list of matrices of survival and reproductive output if `split = TRUE`, otherwise a single matrix of the sum of survival and reproductive output.

Author(s)

Owen Jones jones@biology.sdu.dk

See Also

Other errors: `calculate_errors()`, `compute_ci()`, `compute_ci_U()`

Other errors: `calculate_errors()`, `compute_ci()`, `compute_ci_U()`

Examples

```
set.seed(42) # set seed for repeatability

# First generate a set of MPMs
mpm_set <- rand_lefko_set(n = 5, n_stages = 5, fecundity = c(
  0, 0, 4, 8, 10
), archetype = 4, output = "Type4")

# Now apply sampling error to this set
add_mpm_error(
  mat_U = mpm_set$U_list, mat_F = mpm_set$F_list, sample_size =
    50
)

# Also works with a single matrix.
mats <- make_leslie_mpm(
  survival = c(0.1, 0.2, 0.5),
  fecundity = c(0, 1.2, 2.4),
  n_stages = 3, split = TRUE
)

# Sample size is a single value
add_mpm_error(mat_U = mats$mat_U, mat_F = mats$mat_F, sample_size = 20)

# Sample size is a list of two matrices
# here with a sample size of 20 for fecundity and 10 for growth/survival.
mpm_set <- rand_lefko_set(
  n = 5, n_stages = 3, fecundity = c(0, 2, 4),
  archetype = 4, output = "Type4"
)

ssMats <- list(
```

```
  "mat_F_ss" = matrix(20, nrow = 3, ncol = 3),
  "mat_U_ss" = matrix(10, nrow = 3, ncol = 3)
)

# Add sampling error to the matrix models
output <- add_mpm_error(
  mat_U = mpm_set$U_list, mat_F = mpm_set$F_list,
  sample_size = ssMats
)

# Examine the outputs
names(output)
output
```

`calculate_errors` *Calculate error (standard error or 95%CI) in elements of a matrix population model.*

Description

Given two submatrices of a matrix population model (`mat_U` and `mat_F`, the growth/survival matrix and the fecundity matrix respectively) and a sample size, or matrix/matrices of sample sizes, this function calculates the standard error or 95% confidence interval (95%CI) for each element of the matrix. These calculations assume that `mat_U` is the result of binomial processes (i.e., the survival (0/1) of a sample of n individuals), while `mat_F` is the result of Poisson processes (i.e., counts of offspring from n individuals), where n is the sample size.

Usage

```
calculate_errors(mat_U, mat_F, sample_size, type = "sem", calculate_A = TRUE)
```

Arguments

`mat_U`	matrix of mean survival probabilities
`mat_F`	matrix of mean fecundity values
`sample_size`	either (1) a single matrix of sample sizes for each element of the MPM, (2) a list of two named matrices ("`mat_F_ss`", "`mat_U_ss`") containing sample sizes for the survival and fecundity submatrices of the MPM or (3) a single value applied to the whole matrix
`type`	A character string indicating the type of error to calculate. Must be one of "`sem`" (standard error), or "`CI95`" (95% confidence interval).
`calculate_A`	A logical argument indicating whether the returned error information should include the A matrix and its error. Defaults to `TRUE`.

Details

The output is a list containing the original matrices and matrices showing error estimates or confidence intervals.

Value

A list containing the original matrices and the error estimates (or upper and lower confidence intervals) for the U, F and (optionally) A matrices.

Author(s)

Owen Jones jones@biology.sdu.dk

See Also

`add_mpm_error()` which simulates matrices with known values and sample sizes.

Other errors: `add_mpm_error()`, `compute_ci()`, `compute_ci_U()`

Examples

```
# Set up two submatrices
matU <- matrix(c(
  0.1, 0,
  0.2, 0.4
), byrow = TRUE, nrow = 2)
matF <- matrix(c(
  0, 4,
  0., 0.
), byrow = TRUE, nrow = 2)

# errors as 95% CI, with a sample size of 20 for all elements
calculate_errors(mat_U = matU, mat_F = matF, sample_size = 20, type = "CI95")

# errors as sem, with a sample size of 20 for all elements
calculate_errors(mat_U = matU, mat_F = matF, sample_size = 20, type = "sem")

# Sample size is a single matrix applied to both F and U matrices
ssMat <- matrix(10, nrow = 2, ncol = 2)

calculate_errors(
  mat_U = matU, mat_F = matF, sample_size = ssMat, type =
    "sem"
)

# Sample size is a list of two matrices, one for F and one for U.
ssMats <- list(
  "mat_F_ss" = matrix(10, nrow = 2, ncol = 2),
  "mat_U_ss" = matrix(10, nrow = 2, ncol = 2)
)
calculate_errors(
  mat_U = matU, mat_F = matF, sample_size = ssMats, type =
    "sem"
)
```

`compute_ci` *Compute 95% confidence intervals for derived estimates from a matrix population model*

Description

This function computes the 95% confidence interval for measures derived from a matrix population model using parametric bootstrapping. In this approach a sampling distribution of the matrix population model (MPM) is generated by taking a large number of random independent draws using the sampling distribution of each underlying transition rate. The approach rests on our assumption that survival-related processes are binomial, while fecundity is a Poisson process (see the function `add_mpm_error()` for details).

Usage

```
compute_ci(mat_U, mat_F, sample_size, FUN, ..., n_sim = 1000, dist.out = FALSE)
```

Arguments

`mat_U`	A matrix that describes the growth and survival process.
`mat_F`	A matrix that describes fecundity.
`sample_size`	either (1) a single matrix of sample sizes for each element of the MPM, (2) a list of two named matrices ("`mat_F_ss`", "`mat_U_ss`") containing sample sizes for the survival and fecundity submatrices of the MPM or (3) a single value applied to the whole matrix
`FUN`	A function to apply to each simulated matrix population model. This function must take, as input, a single matrix population model (i.e., the A matrix). For functions that require only the U matrix, use `compute_ci_U`.
`...`	Additional arguments to be passed to `FUN`.
`n_sim`	An integer indicating the number of simulations to run. Default is 1000.
`dist.out`	Logical. If TRUE, returns a list with both the quantiles and the simulated estimates. Default is FALSE.

Details

The inputs are the U matrix, which describes the survival-related processes, and the F matrix which describes fecundity. The underlying assumption is that the U matrix is the average of a binomial process while the F matrix is the average of a Poisson process . The confidence interval will depend largely on the sample size used.

Value

If dist.out is FALSE, a numeric vector of the 2.5th and 97.5th quantiles of the estimated measures. If `dist.out = TRUE`, a list with two elements: `quantiles` and `estimates`. `quantiles` is a numeric vector of the 2.5th and 97.5th quantiles of the estimated measures, and `estimates` is a numeric vector of the estimated measures.

Author(s)

Owen Jones jones@biology.sdu.dk

References

Chapter 12 in Caswell, H. (2001). Matrix Population Models. Sinauer Associates Incorporated.

See Also

Other errors: `add_mpm_error`(), `calculate_errors`(), `compute_ci_U`()

Examples

```
set.seed(42) # set seed for repeatability

# Data for use in example
matU <- matrix(c(
  0.1, 0.0,
  0.2, 0.4
), byrow = TRUE, nrow = 2)

matF <- matrix(c(
  0.0, 5.0,
  0.0, 0.0
), byrow = TRUE, nrow = 2)

set.seed(42)

# Example of use to calculate 95% CI of lambda
compute_ci(
  mat_U = matU, mat_F = matF, sample_size = 10, FUN =
    popbio::lambda
)

# Example of use to calculate 95% CI of generation time
compute_ci(
  mat_U = matU, mat_F = matF, sample_size = 40, FUN =
    popbio::generation.time
)

# Example of use to calculate 95% CI of generation time and show the
# distribution of those bootstrapped estimates
xx <- compute_ci(
  mat_U = matU, mat_F = matF, sample_size = 100, FUN =
    popbio::generation.time, dist.out = TRUE
)
summary(xx$quantiles)
hist(xx$estimates)
```

`compute_ci_U` *Compute 95% confidence intervals for derived estimates from the U submatrix of a matrix population model*

Description

This function computes the 95% confidence interval for measures derived from the U submatrix of a matrix population model using parametric bootstrapping. In this approach a sampling distribution of the U submatrix is generated by taking a large number of random independent draws using the sampling distribution of each underlying transition rate. The approach rests on our assumption that survival-related processes are binomial (see the function `add_mpm_error()` for details).

Usage

```
compute_ci_U(mat_U, sample_size, FUN, ..., n_sim = 1000, dist.out = FALSE)
```

Arguments

`mat_U`	A matrix that describes the growth and survival process.
`sample_size`	either (1) a single matrix of sample sizes for each element of the U matrix, (2) a single value applied to the whole matrix
`FUN`	A function to apply to each simulated matrix population model. This function must take, as input, a single U submatrix of a matrix population model (i.e., the U matrix). For functions that require the A matrix, use `compute_ci`.
`...`	Additional arguments to be passed to `FUN`.
`n_sim`	An integer indicating the number of simulations to run. Default is 1000.
`dist.out`	Logical. If TRUE, returns a list with both the quantiles and the simulated estimates. Default is FALSE.

Details

The main inputs is the U matrix, which describes the survival-related processes. The underlying assumption is that the U matrix is the average of a binomial process. The confidence interval will depend largely on the sample size used.

Value

If dist.out is FALSE, a numeric vector of the 2.5th and 97.5th quantiles of the estimated measures. If `dist.out = TRUE`, a list with two elements: `quantiles` and `estimates`. `quantiles` is a numeric vector of the 2.5th and 97.5th quantiles of the estimated measures, and `estimates` is a numeric vector of the estimated measures.

Author(s)

Owen Jones jones@biology.sdu.dk

References

Chapter 12 in Caswell, H. (2001). Matrix Population Models. Sinauer Associates Incorporated.

See Also

Other errors: `add_mpm_error()`, `calculate_errors()`, `compute_ci()`

Examples

```
set.seed(42) # set seed for repeatability

# Data for use in example
matU <- matrix(c(
  0.1, 0.0,
  0.2, 0.4
), byrow = TRUE, nrow = 2)


# Example of use to calculate 95% CI of life expectancy
compute_ci_U(
  mat_U = matU, sample_size = 10, FUN =
    Rage::life_expect_mean
)

# Example of use to calculate 95% CI of generation time and show the
# distribution of those bootstrapped estimates
xx <- compute_ci_U(
  mat_U = matU, sample_size = 100, FUN =
    Rage::life_expect_mean, dist.out = TRUE
)

summary(xx$quantiles)
hist(xx$estimates)
```

`driven_vital_rate` *Calculate driven vital rates*

Description

This function calculates new values for a vital rate, such as survival or fecundity that is being influenced by a driver (e.g., weather). It does this by using a driver variable and a baseline value, along with a specified slope for the relationship between the driver variable and the vital rate. The function works on a linearised scale, using logit for survival and log for fecundity, and takes into account the error standard deviation.

Usage

```
driven_vital_rate(
  driver,
  baseline_value = NULL,
  slope = NULL,
  baseline_driver = NULL,
  error_sd = 0,
  link = "logit"
)
```

Arguments

`driver`	A vector of driver values.
`baseline_value`	A vector or matrix of baseline values for the vital rate (e.g., survival) that is being influenced ("driven") by another variable (e.g. a climatic variable).
`slope`	A vector or matrix of slopes for the relationship between the driver variable and the vital rate being driven.
`baseline_driver`	The `baseline_driver` parameter is a single value representing the baseline driver value. If the driver value is greater than this value and the slope is positive, then the resulting vital rate will be higher. Conversely, if the driver value is less than this variable and the slope is positive, then the resulting vital rate will be less than the baseline value.
`error_sd`	A vector or matrix of error standard deviations for random normal error to be added to the driven value of the vital rate being modelled. If set to 0 (the default), no error is added.
`link`	A character string indicating the type of link function to use. Valid values are "`logit`" (the default) and "`log`", which are appropriate for survival (U submatrix) and fecundity (F submatrix) respectively.

Details

The relationship between the driver variable and the vital rate is assumed to be linear:

$$V = a * (d - d_b) + x + E$$

Where $$V$$ is the new vital rate (on the scale of the linear predictor), $$a$$ is the slope, $$x$$ is the baseline vital rate, $$d$$ is the driver, $$d_b$$ is the baseline driver and $$E$$ is the error.

The input vital rate(s) (`baseline_value`) can be a single-element vector representing a single vital rate (e.g., survival probability or fecundity), a longer vector representing a series of vital rates (e.g., several survival probabilities or fecundity values), or a matrix of values (e.g., a U or F submatrix of a matrix population model). The `slope`s of the relationship between the vital rate (`baseline_value`) and the driver can be provided as a single value, which is applied to all elements of the input vital rates, or as a matrix of values that map onto the matrix of vital rates. This allows users to simulate cases where different vital rates in a matrix model are affected in different ways by the same weather driver. For example, juvenile survival might be more affected by the driver than adult survival. The `baseline_driver` value represents the "normal" state of the driver. If the driver is greater than the `baseline_driver` and the `slope` is positive, then the outcome vital rate will be higher. If the driver is less than the `baseline_driver` variable and the `slope` is positive, then the outcome vital rate will be less than the `baseline_value`. The `error_sd` represents the error in the linear relationship between the driver and the vital rate.

Value

Depending on the input types, either a single value, a vector or a list of matrices of driven values for the vital rate(s) being modelled. The list has a length equal to the length of the `driver` input parameter.

Author(s)

Owen Jones jones@biology.sdu.dk

Examples

```
set.seed(42) # set seed for repeatability

# A single vital rate and a single driver
driven_vital_rate(
  driver = 14,
  baseline_value = 0.5,
  slope = .4,
  baseline_driver = 10,
  error_sd = 0,
  link = "logit"
)

# A single vital rate and a time series of drivers
driven_vital_rate(
  driver = runif(10, 5, 15),
  baseline_value = 0.5,
  slope = .4,
  baseline_driver = 10,
  error_sd = 0,
  link = "logit"
)

# A matrix of survival values (U submatrix of a Leslie model)
# with a series of drivers, and matrices of slopes and errors

lt1 <- model_survival(params = c(b_0 = 0.4, b_1 = 0.5), model = "Gompertz")
lt1$fecundity <- model_fecundity(
  age = 0:max(lt1$x), params = c(A = 10),
  maturity = 3, model = "step"
)

mats <- make_leslie_mpm(
  survival = lt1$px, fecundity = lt1$fecundity, n_stages =
    nrow(lt1), split = TRUE
)
mats$mat_U
```

```
mat_dim <- nrow(mats$mat_U)

driven_vital_rate(
  driver = runif(5, 5, 15),
  baseline_value = mats$mat_U,
  slope = matrix(.4,
    nrow = mat_dim,
    ncol = mat_dim
  ),
  baseline_driver = 10,
  error_sd = matrix(1, nrow = mat_dim, ncol = mat_dim),
  link = "logit"
)
```

`make_leslie_mpm` *Create a Leslie matrix population model*

Description

The function creates a Leslie matrix from inputs of number of stages, fecundity (the top row of the matrix), and survival probability (the value in the sub-diagonal).

Usage

```
make_leslie_mpm(
  survival = NULL,
  fecundity = NULL,
  n_stages = NULL,
  lifetable = NULL,
  split = FALSE
)
```

Arguments

`survival`	a numeric value representing the survival probability of each stage along the lower off-diagonal of the matrix, with the final value being in the lower-right corner of the matrix. If only one value is provided, this is applied to all survival elements.
`fecundity`	a numeric vector of length n_stages representing the reproductive output of each stage. If only one value is provided, this is applied to all fecundity elements.
`n_stages`	a numeric value representing the number of stages in the matrix
`lifetable`	a life table containing columns `px` (age-specific survival) and `fecundity` (age-specific fecundity).
`split`	a logical argument indicating whether the output matrix should be split into separate A, U and F matrices (where A = U + F).

Details

Note that the simulations assume a post-breeding census, thus avoiding the often overlooked issue of unaccounted survival to reproduction highlighted by Kendall et al. (2019). Furthermore, the simulations assume no covariance among matrix elements (e.g. between reproduction and survival), and therefore do not allow the users to capture trade offs directly. This capability is roadmapped for a future package release.

Value

A matrix of size n_stages x n_stages representing the Leslie matrix

Author(s)

Owen Jones jones@biology.sdu.dk

References

Caswell, H. (2001). Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer.

Leslie, P. H. (1945). On the use of matrices in certain population mathematics. Biometrika, 33 (3), 183–212.

Leslie, P. H. (1948). Some Further Notes on the Use of Matrices in Population Mathematics. Biometrika, 35(3-4), 213–245.

Kendall, B. E., Fujiwara, M., Diaz-Lopez, J., Schneider, S., Voigt, J., & Wiesner, S. (2019). Persistent problems in the construction of matrix population models. Ecological Modelling, 406, 33–43.

See Also

- `model_survival()` to model age-specific survival using mortality models.
- `model_fecundity()` to model age-specific reproductive output using various functions.

Other Leslie matrices: `rand_leslie_set()`, `reorganise_matrices()`

Examples

```
make_leslie_mpm(
  survival = 0.5, fecundity = c(0.1, 0.2, 0.3),
  n_stages = 3, split = FALSE
)
make_leslie_mpm(
  survival = c(0.5, 0.6, 0.7), fecundity = c(0.1, 0.2, 0.3),
  n_stages = 3
)
make_leslie_mpm(
  survival = seq(0.1, 0.7, length.out = 4), fecundity = 0.1,
  n_stages = 4
)
make_leslie_mpm(
  survival = c(0.8, 0.3, 0.2, 0.1, 0.05), fecundity = 0.2,
  n_stages = 5
)
```

`model_fecundity` *Model reproductive output with age using set functional forms*

Description

This function computes reproductive output (often referred to as fertility in human demography and fecundity in population biology) based on the logistic, step, von Bertalanffy, Hadwiger, and normal models. The logistic model assumes that reproductive output increases sigmoidally with age from maturity until a maximum is reached. The step model assumes that reproductive output is zero before the age of maturity and then remains constant. The von Bertalanffy model assumes that, after maturity, reproductive output increases asymptotically with age until a maximum is reached. In this formulation, the model is set up so that reproductive output is 0 at the 'age of maturity - 1', and increases from that point. The Hadwiger model, while originally intended to model human fertility with a characteristic hump-shaped curve, is applied here to model fecundity (actual offspring production). For all models, the output ensures that reproductive output is zero before the age at maturity.

Usage

```
model_fecundity(params, age = NULL, maturity = 0, model = "logistic")

model_reproduction(params, age = NULL, maturity = 0, model = "logistic")

model_fertility(params, age = NULL, maturity = 0, model = "logistic")
```

Arguments

`params`	A numeric vector of parameters for the selected model. The number and meaning of parameters depend on the selected model.
`age`	A numeric vector representing age. For use in creation of MPMs and life tables, these should be integers.
`maturity`	A non-negative numeric value indicating the age at maturity. Whatever model is used, the reproductive output is forced to be 0 below the age of maturity.
`model`	A character string specifying the model to use. Must be one of "logistic", "step", "vonbertalanffy","normal" or "hadwiger".

Details

The required parameters varies depending on the model used. The parameters are provided as a vector and the parameters must be provided in the order mentioned here.

- Logistic: $f_x = A/(1 + exp(-k(x - x_m)))$
- Step: $f_x = \begin{cases} A, x \geq m \\ A, x < m \end{cases}$

- von Bertalanffy: $f_x = A(1 - exp(-k(x - x_0)))$
- Normal: $f_x = A \times \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$
- Hadwiger: $f_x = \frac{ab}{C}\left(\frac{C}{x}\right)^{\frac{3}{2}} \exp\left\{-b^2\left(\frac{C}{x} + \frac{x}{C} - 2\right)\right\}$

Value

A numeric vector representing the computed reproductive output values.

Author(s)

Owen Jones jones@biology.sdu.dk

References

Bertalanffy, L. von (1938) A quantitative theory of organic growth (inquiries on growth laws. II). Human Biology 10:181–213.

Peristera, P. & Kostaki, A. (2007) Modeling fertility in modern populations. Demographic Research. 16. Article 6, 141-194 doi:10.4054/DemRes.2007.16.6

See Also

`model_mortality()` to model age-specific survival using mortality models.

Other trajectories: `model_survival()`

Examples

```
# Compute reproductive output using the step model
model_fecundity(age = 0:20, params = c(A = 10), maturity = 2, model = "step")

# Compute reproductive output using the logistic model
model_fecundity(
  age = 0:20, params = c(A = 10, k = 0.5, x_m = 8), maturity =
    0, model = "logistic"
)

# Compute reproductive output using the von Bertalanffy model
model_fecundity(
  age = 0:20, params = c(A = 10, k = .3), maturity = 2, model =
    "vonbertalanffy"
)

# Compute reproductive output using the normal model
model_fecundity(
  age = 0:20, params = c(A = 10, mu = 4, sd = 2), maturity = 0,
  model = "normal"
)

# Compute reproductive output using the Hadwiger model
model_fecundity(
```

```
  age = 0:50, params = c(a = 0.91, b = 3.85, C = 29.78),
  maturity = 0, model = "hadwiger"
)

model_reproduction(age = 0:20, params = c(A = 10), maturity = 2, model = "step")
model_fertility(age = 0:20, params = c(A = 10), maturity = 2, model = "step")
```

`model_survival` *Model mortality hazard, survivorship and age-specific survival probability using a mortality model*

Description

Generates an actuarial life table based on a defined mortality model.

Usage

```
model_survival(params, age = NULL, model, truncate = 0.01)

model_mortality(params, age = NULL, model, truncate = 0.01)
```

Arguments

`params`	Numeric vector representing the parameters of the mortality model.
`age`	Numeric vector representing age. The default is `NULL`, whereby the survival trajectory is modelled from age 0 to the age at which the survivorship of the synthetic cohort declines to a threshold defined by the `truncate` argument, which has a default of `0.01` (i.e. 1% of the cohort remaining alive).
`model`	A character string specifying the name of the mortality model to be used. Options are `gompertz`, `gompertzmakeham`, `exponential`, `siler`, `weibull`, and `weibullmakeham`. These names are not case-sensitive.
`truncate`	a value defining how the life table output should be truncated. The default is `0.01`, indicating that the life table is truncated so that survivorship (`lx`) > 0.01 (i.e. the age at which 1% of the cohort remains alive).

Details

The required parameters varies depending on the mortality model. The parameters are provided as a vector.

*For `gompertz` and `weibull`, the parameters are `b0`, `b1`. *For `gompertzmakeham` and `weibullmakeham` the parameters are `b0`, `b1` and `C`. *For `exponential`, the parameter is `C`. *For `siler`, the parameters are `a0`, `a1`, `C`, `b0` and `b1`.

Note that the parameters must be provided in the order mentioned here. `x` represents age.

- Gompertz: $h_x = b_0 \mathrm{e}^{b_1 x}$
- Gompertz-Makeham: $h_x = b_0 \mathrm{e}^{b_1 x} + c$

- Exponential: $h_x = c$
- Siler: $h_x = a_0 \mathrm{e}^{-a_1 x} + c + b_0 \mathrm{e}^{b_1 x}$
- Weibull: $h_x = b_0 b_1 (b_1 x)^{(b_0 - 1)}$
- Weibull-Makeham: $h_x = b_0 b_1 (b_1 x)^{(b_0 - 1)} + c$

In the output, the probability of survival (`px`) (and death (`qx`)) represent the probability of individuals that enter the age interval $[x, x+1]$ survive until the end of the interval (or die before the end of the interval). It is not possible to estimate a value for this in the final row of the life table (because there is no $x+1$ value) and therefore the input values of `age` (`x`) may need to be extended to capture this final interval.

Value

A dataframe in the form of a lifetable with columns for age (`x`), hazard (`hx`), survivorship (`lx`) and mortality (`qx`) and survival probability within interval (`px`).

Author(s)

Owen Jones jones@biology.sdu.dk

References

Cox, D.R. & Oakes, D. (1984) Analysis of Survival Data. Chapman and Hall, London, UK.

Pinder III, J.E., Wiener, J.G. & Smith, M.H. (1978) The Weibull distribution: a method of summarizing survivorship data. Ecology, 59, 175–179.

Pletcher, S. (1999) Model fitting and hypothesis testing for age-specific mortality data. Journal of Evolutionary Biology, 12, 430–439.

Siler, W. (1979) A competing-risk model for animal mortality. Ecology, 60, 750–757.

Vaupel, J., Manton, K. & Stallard, E. (1979) The impact of heterogeneity in individual frailty on the dynamics of mortality. Demography, 16, 439–454.

See Also

`model_fecundity()` to model age-specific reproductive output using various functions.

Other trajectories: `model_fecundity()`

Examples

```
model_mortality(params = c(b_0 = 0.1, b_1 = 0.2), model = "Gompertz")

model_mortality(
  params = c(b_0 = 0.1, b_1 = 0.2, C = 0.1),
  model = "GompertzMakeham",
  truncate = 0.1
)

model_mortality(params = c(c = 0.2), model = "Exponential", age = 0:10)
```

```
model_mortality(
  params = c(a_0 = 0.1, a_1 = 0.2, C = 0.1, b_0 = 0.1, b_1 = 0.2),
  model = "Siler",
  age = 0:10
)

model_mortality(
  params = c(b_0 = 1.4, b_1 = 0.18),
  model = "Weibull"
)

model_mortality(
  params = c(b_0 = 1.1, b_1 = 0.05, c = 0.2),
  model = "WeibullMakeham"
)

model_survival(params = c(b_0 = 0.1, b_1 = 0.2), model = "Gompertz")
```

`plot_matrix` *Plot a matrix as a heatmap*

Description

Visualise a matrix, such as a matrix population model (MPM), as a heatmap.

Usage

```
plot_matrix(mat, zero_na = FALSE, legend = FALSE, na_colour = NA, ...)
```

Arguments

`mat`	A matrix, such as the A matrix of a matrix population model
`zero_na`	Logical indicating whether zero values should be treated as NA
`legend`	Logical indicating whether to include a legend
`na_colour`	Colour for NA values
`...`	Additional arguments to be passed to ggplot

Value

A ggplot object

Author(s)

Owen Jones jones@biology.sdu.dk

See Also

Other utility: `summarise_mpms()`

Examples

```
matDim <- 10
A1 <- make_leslie_mpm(
  survival = seq(0.1, 0.7, length.out = matDim),
  fecundity = seq(0.1, 0.7, length.out = matDim),
  n_stages = matDim
)
plot_matrix(A1, zero_na = TRUE, na_colour = "black")
plot_matrix(A1, zero_na = TRUE, na_colour = NA)
```

`rand_lefko_mpm` *Generate random Lefkovitch matrix population models (MPMs) based on life history archetypes*

Description

Generates a random matrix population model (MPM) with element values based on defined life history archetypes. Survival and transition/growth probabilities from any particular stage are restricted to be less than or equal to 1 by drawing from a Dirichlet distribution. The user can specify archetypes (from Takada et al. 2018) to restrict the MPMs in other ways:

- Archetype 1: all elements are positive, although they may be very small. Therefore, transition from/to any stage is possible. This model describes a life history where individuals can progress and retrogress rapidly.
- Archetype 2: has the same form as archetype 1 (transition from/to any stage is possible), but the survival probability (column sums of the survival matrix) increases monotonously as the individuals advance to later stages. This model, as the one in the first archetype, also allows for rapid progression and retrogression, but is more realistic in that stage-specific survival probability increases with stage advancement.
- Archetype 3: positive non-zero elements for survival are only allowed on the diagonal and lower sub-diagonal of the matrix This model represents the life cycle of a species where retrogression is not allowed, and progression can only happen to the immediately larger/more developed stage (slow progression, e.g., trees).
- Archetype 4: This archetype has the same general form as archetype 3, but with the further assumption that stage-specific survival increases as individuals increase in size/developmental stage. In this respect it is similar to archetype 2.

Usage

```
rand_lefko_mpm(n_stages, fecundity, archetype = 1, split = TRUE)
```

Arguments

`n_stages`	An integer defining the number of stages for the MPM.
`fecundity`	A measure of reproductive output. The average number of offspring produced per projection interval from each stage. Values can be provided in 4 ways:

- An numeric vector of length 1 to provide a single fecundity measure to the top right corner of the matrix model only.
- A numeric vector of integers of length equal to `n_stages` to provide fecundity estimates for the whole top row of the matrix model. Use 0 for cases with no fecundity.
- A matrix of numeric values of the same dimension as `n_stages` to provide fecundity estimates for the entire matrix model. Use 0 for cases with no fecundity.
- A list of two matrices of numeric values, both with the same dimension as `n_stages`, to provide lower and upper limits of mean fecundity for the entire matrix model. Use 0 for both lower and upper limits in cases with no fecundity.

In the latter case, a fecundity value will be drawn from a uniform distribution for the defined range. If there is no fecundity in a particular age class, use a value of 0 for both the lower and upper limit.

`archetype`	Indication of which life history archetype should be used, based on Takada et al. 2018. An integer between 1 and 4.
`split`	`TRUE`/`FALSE`, indicating whether the matrix produced should be split into a survival matrix and a reproductive output matrix. If true, then the output becomes a list with a matrix in each element. Otherwise, the output is a single matrix. Default is `TRUE`.

Details

In all 4 of these Archetypes, reproductive output is placed as a single element on the top right of the matrix, if it is a single value. If it is a vector of length `n_stages` then the fecundity vector spans the entire top row of the matrix.

The function is constrained to only output ergodic matrices.

Note that the simulations assume a post-breeding census, thus avoiding the often overlooked issue of unaccounted survival to reproduction highlighted by Kendall et al. (2019). Furthermore, the simulations assume no covariance among matrix elements (e.g. between reproduction and survival), and therefore do not allow the users to capture trade offs directly. This capability is roadmapped for a future package release.

Value

Returns a random matrix population model with characteristics determined by the archetype selected and fecundity vector. If split = TRUE, the matrix is split into separate reproductive output and growth/survival matrices, returned as a list.

Author(s)

Owen Jones jones@biology.sdu.dk

References

Caswell, H. (2001). Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer.

Lefkovitch, L. P. (1965). The study of population growth in organisms grouped by stages. Biometrics, 21(1), 1.

Takada, T., Kawai, Y., & Salguero-Gómez, R. (2018). A cautionary note on elasticity analyses in a ternary plot using randomly generated population matrices. Population Ecology, 60(1), 37–47.

Kendall, B. E., Fujiwara, M., Diaz-Lopez, J., Schneider, S., Voigt, J., & Wiesner, S. (2019). Persistent problems in the construction of matrix population models. Ecological Modelling, 406, 33–43.

See Also

`rand_lefko_set()` which is a wrapper for this function allowing the generation of large numbers of random matrices of this type.

Other Lefkovitch matrices: `rand_lefko_set()`

Examples

```
set.seed(42) # set seed for repeatability

rand_lefko_mpm(n_stages = 2, fecundity = 20, archetype = 1, split = FALSE)
rand_lefko_mpm(n_stages = 2, fecundity = 20, archetype = 2, split = TRUE)
rand_lefko_mpm(n_stages = 3, fecundity = 20, archetype = 3, split = FALSE)
rand_lefko_mpm(n_stages = 4, fecundity = 20, archetype = 4, split = TRUE)
rand_lefko_mpm(
  n_stages = 5, fecundity = c(0, 0, 4, 8, 10), archetype = 4,
  split = TRUE
)
# Using a range of values for fecundity
rand_lefko_mpm(n_stages = 2, fecundity = 20, archetype = 1, split = TRUE)
```

`rand_lefko_set` *Generate lists of Lefkovitch matrix population models (MPMs) based on life history archetypes*

Description

This function generates a list of `n` MPMs according to the specified criteria. Criteria include the `archetype`, and the acceptable constraining criteria, which could include lambda, generation time or any other metric derived from an A matrix. The function attempts to find matrices that fulfil the criteria, discarding unacceptable matrices. By default, if it takes more than 1000 attempts to find a suitable matrix model, then an error is produced. However, the number of attempts can be altered with the `attempts` parameter.

Usage

```
rand_lefko_set(
  n_models = 5,
  n_stages = 3,
```

```
  archetype = 1,
  fecundity = 1.5,
  output = "Type1",
  max_surv = 0.99,
  constraint = NULL,
  attempts = 1000
)
```

Arguments

`n_models`	An integer indicating the number of MPMs to generate.
`n_stages`	The number of stages for the MPMs. Default is `3`.
`archetype`	The archetype of the MPMs. Default is `1`.
`fecundity`	The average number of offspring produced (fecundity). Values can be provided in 4 ways: • An numeric vector of length 1 providing a single measure of fecundity to the top right corner of the matrix model only. • A numeric vector of integers of length equal to `n_stages` to provide fecundity estimates for the whole top row of the matrix model. Use 0 for cases with no fecundity. • A matrix of numeric values of the same dimension as `n_stages` to provide fecundity estimates for the entire matrix model. Use 0 for cases with no fecundity. • A list of two matrices of numeric values, both with the same dimension as `n_stages`, to provide lower and upper limits of mean fecundity for the entire matrix model. In the latter case, a fecundity value will be drawn from a uniform distribution for the defined range. If there is no fecundity in a particular age class, use a value of 0 for both the lower and upper limit.
`output`	Character string indicating the type of output. • `Type1` or `cdb_split`: A `compadreDB` Object containing MPMs split into the submatrices (i.e. A, U, F and C). • `Type2` or `cdb_A`: A `compadreDB` Object containing MPMs that are not split into submatrices (i.e. only the A matrix is included). • `Type3`: A `list_split1` of MPMs arranged so that each element of the list contains a model and associated submatrices (i.e. the nth element contains the nth A matrix alongside the nth U and F matrices). • `Type4`: A `list_split2` of MPMs arranged so that the list contains 3 lists for the A matrix and the U and F submatrices respectively. • `Type5` or `list_A`: A `list` of MPMs, including only the A matrix.
`max_surv`	The maximum acceptable survival value, calculated across all transitions from a stage. Defaults to `0.99`. This is only used the output splits a matrix into the submatrices.
`constraint`	An optional data frame with 4 columns named `fun`, `arg`, `lower` and `upper`. These columns specify (1) a function that outputs a metric derived from an A

	matrix and (2) an argument for the function (`NA`, if no argument supplied) (3) the lower acceptable bound for the metric and (4) upper acceptable bound for the metric.
`attempts`	An integer indicating the number of attempts To be made when simulating matrix model. The default is 1000. If it takes more than 1000 attempts to make a matrix that satisfies the conditions set by the other arguments, then a warning is produced.

Value

A `compadreDB` object or list of MPMs that meet the specified criteria.

Author(s)

Owen Jones jones@biology.sdu.dk

References

Caswell, H. (2001). Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer.

Lefkovitch, L. P. (1965). The study of population growth in organisms grouped by stages. Biometrics, 21(1), 1.

Takada, T., Kawai, Y., & Salguero-Gómez, R. (2018). A cautionary note on elasticity analyses in a ternary plot using randomly generated population matrices. Population Ecology, 60(1), 37–47.

See Also

`rand_lefko_mpm()` which this function is essentially a wrapper for.

Other Lefkovitch matrices: `rand_lefko_mpm()`

Examples

```
set.seed(42) # set seed for repeatability

# Basic operation, without splitting matrices and with no constraints
rand_lefko_set(
  n_models = 3, n_stages = 5, fecundity = c(0, 0, 4, 8, 10),
  archetype = 4, output = "Type5"
)

# Constrain outputs to A matrices with lambda between 0.9 and 1.1
library(popbio)
constrain_df <- data.frame(
  fun = "lambda", arg = NA, lower = 0.9, upper =
    1.1
)
rand_lefko_set(
  n_models = 10, n_stages = 5, fecundity = c(0, 0, 4, 8, 10),
  archetype = 4, constraint = constrain_df, output = "Type5"
)
```

```
# As above, but using popdemo::eigs function instead of popbio::lambda
# to illustrate use of argument
library(popdemo)
constrain_df <- data.frame(
  fun = "eigs", arg = "lambda", lower = 0.9, upper = 1.1
)

rand_lefko_set(
  n_models = 10, n_stages = 5, fecundity = c(0, 0, 4, 8, 10),
  archetype = 4, constraint = constrain_df, output = "Type5"
)

# Multiple constraints
# Constrain outputs to A matrices with lambda between 0.9 and 1.1, generation
# time between 3 and 5 and damping ratio between 1 and 7.
library(popbio)
constrain_df <- data.frame(
  fun = c("lambda", "generation.time", "damping.ratio"),
  arg = c(NA, NA, NA),
  lower = c(0.9, 3.0, 1.0),
  upper = c(1.1, 5.0, 7.0)
)
rand_lefko_set(
  n_models = 10, n_stages = 5, fecundity = c(0, 0, 4, 8, 10),
  archetype = 4, constraint = constrain_df, output = "Type5"
)

# Constraints based on user-defined functions are also possible...
# User-defined function to calculate survival from the first stage
simpleFun <- function(x) {
  sum(x[, 1])
}

# Define the constraint, based on the user-defined function
constrain_df <- data.frame(
  fun = "simpleFun", arg = NA, lower = 0.75, upper = 1
)

rand_lefko_set(
  n_models = 10, n_stages = 5, fecundity = c(0, 0, 4, 8, 10),
  archetype = 4, output = "Type5", constraint = constrain_df
)
```

`rand_leslie_set` *Generate a set of random Leslie Matrix Population Models*

Description

Generates a set of Leslie matrix population models (MPMs) based on defined mortality and reproductive output models, and using model parameters randomly drawn from specified distributions.

Usage

```
rand_leslie_set(
  n_models = 5,
  mortality_model = "gompertz",
  fecundity_model = "step",
  mortality_params,
  fecundity_params,
  fecundity_maturity_params,
  dist_type = "uniform",
  output = "type1",
  scale_output = FALSE
)
```

Arguments

`n_models`	An integer indicating the number of MPMs to generate.
`mortality_model`	A character string specifying the name of the mortality model to be used. Options are `gompertz`, `gompertzmakeham`, `exponential`, `siler`, `weibull`, and `weibullmakeham`. See `model_mortality`. These names are not case-sensitive.
`fecundity_model`	A character string specifying the name of the model to be used for reproductive output. Options are `logistic`, `step`, `vonBertalanffy`, `normal` and `hadwiger`. See `?model_fecundity`.
`mortality_params`	A two-column dataframe with a number of rows equal to the number of parameters in the mortality model. The required order of the parameters depends on the selected `mortality_model` (see `?model_mortality`): • For `gompertz` and `weibull`: `b_0`, `b_1` • For `gompertzmakeham` and `weibullmakeham`: `b_0`, `b_1`, `C` • For `exponential`: `C` • For `siler`: `a_0`, `a_1`, `C`, `b_0`, `b_1` If `dist_type` is `uniform` these rows represent the lower and upper limits of the random uniform distribution from which the parameters are drawn. If `dist_type` is `normal`, the columns represent the mean and standard deviation of a random normal distribution from which the parameter values are drawn.
`fecundity_params`	A two-column dataframe with a number of rows equal to the number of parameters in the fecundity model. The required order of the parameters depends on the selected `fecundity_model` (see `?model_fecundity`): • For `logistic`: `A`, `k`, `x_m` • For `step`: `A` • For `vonBertalanffy`: `A`, `k` • For `normal`: `A`, `mu`, `sd` • For `hadwiger`: `a`, `b`, `C` If `dist_type` is `uniform` these rows represent the lower and upper limits of the random uniform distribution from which the

	parameters are drawn. If `dist_type` is `normal`, the columns represent the mean and standard deviation of a random normal distribution from which the parameter values are drawn.
`fecundity_maturity_params`	A vector with two elements defining the distribution from which age at maturity is drawn for the models. The models will coerce reproductive output to be zero before this point. If `dist_type` is `uniform` these values represent the lower and upper limits of the random uniform distribution from which the parameters are drawn. If `dist_type` is `normal`, the values represent the mean and standard deviation of a random normal distribution from which the parameter values are drawn.
`dist_type`	A character string specifying the type of distribution to draw parameters from. Default is `uniform`. Supported types are `uniform` and `normal`.
`output`	Character string indicating the type of output. • `Type1` or `cdb_split`: A `compadreDB` Object containing MPMs split into the submatrices (i.e. A, U, F and C). • `Type2` or `cdb_A`: A `compadreDB` Object containing MPMs that are not split into submatrices (i.e. only the A matrix is included). • `Type3` or `list_split1`: A list of MPMs arranged so that each element of the list contains a model and associated submatrices (i.e. the nth element contains the nth A matrix alongside the nth U and F matrices). • `Type4` or `list_split2`: A list of MPMs arranged so that the list contains 3 lists for the A matrix and the U and F submatrices respectively. • `Type5` or `list_A`: A `list` of MPMs, including only the A matrix. • `Type6` or `lifetable`: A `list` of life tables. Default is `Type1`.
`scale_output`	A logical argument. If `TRUE` the resulting MPMs or life tables are scaled by adjusting reproductive output so that the population growth rate (lambda) is 1. Default is `FALSE`.

Value

Returns a `CompadreDB` object or `list` containing MPMs or life tables generated using the specified model with parameters drawn from random uniform or normal distributions. The format of the output MPMs depends on the arguments `output`. Outputs may optionally be scaled using the argument `scale_output` to ensure a population growth rate (lambda) of 1.

If the output is a `CompadreDB` object, the parameters of the models used to produce the MPM are included in the metadata.

Author(s)

Owen Jones jones@biology.sdu.dk

See Also

Other Leslie matrices: `make_leslie_mpm()`, `reorganise_matrices()`

Examples

```
mortParams <- data.frame(
  minVal = c(0, 0.01, 0.1),
  maxVal = c(0.14, 0.15, 0.1)
)

fecundityParams <- data.frame(
  minVal = c(10, 0.5, 8),
  maxVal = c(11, 0.9, 10)
)

maturityParam <- c(0, 0)

rand_leslie_set(
  n_models = 5,
  mortality_model = "gompertzmakeham",
  fecundity_model = "normal",
  mortality_params = mortParams,
  fecundity_params = fecundityParams,
  fecundity_maturity_params = maturityParam,
  dist_type = "uniform",
  output = "Type1"
)
```

`reorganise_matrices` *Reorganise Matrix Population Models*

Description

This function reorganises a list of matrix population models, which are split into `mat_A`, `mat_U`, `mat_F`, and optionally `mat_C` sub-matrices. It prepares the matrices for easy conversion into a `compadreDB` object.

Usage

```
reorganise_matrices(matrix_list)
```

Arguments

`matrix_list`	A list of lists, where each sub-list contains the matrices `mat_A`, `mat_U`, `mat_F`, and optionally `mat_C`.

Details

This function processes a list of matrix population models, extracting and grouping the sub-matrices (`mat_A`, `mat_U`, `mat_F`, and optionally `mat_C`) into separate lists. If a `mat_C` matrix is not present in a model, an `NA` matrix of the same size as `mat_U` is used as a placeholder.

Value

A list containing four elements: `mat_A`, `mat_U`, `mat_F`, and `mat_C`. Each element is a list of matrices corresponding to the respective matrix type from the input. If `mat_C` does not exist in a sub-list, it is replaced with an `NA` matrix of the same dimensions as `mat_U`.

Author(s)

Owen Jones jones@biology.sdu.dk

See Also

Other Leslie matrices: `make_leslie_mpm()`, `rand_leslie_set()`

Examples

```
# Example usage
matrix_list <- list(
  list(
    mat_A = matrix(1, 2, 2),
    mat_U = matrix(2, 2, 2),
    mat_F = matrix(3, 2, 2),
    mat_C = matrix(4, 2, 2)
  ),
  list(
    mat_A = matrix(5, 2, 2),
    mat_U = matrix(6, 2, 2),
    mat_F = matrix(7, 2, 2)
  )
)
reorganised_matrices <- reorganise_matrices(matrix_list)
reorganised_matrices$mat_A
```

`summarise_mpms` *Summarise Matrix Population Models*

Description

Calculates and summarises various metrics from matrix population models (MPMs) including dimension (= age in Leslie matrices), lambda values, maximum fecundity values, maximum growth/survival transition probabilities, and minimum non-zero growth/survival transition probabilities

Usage

```
summarise_mpms(x)
```

Arguments

`x`	A `compadreDB` object containing matrix population models, typically in a format compatible with `matA`, `matU`, and `matF` functions.

Value

This function prints summaries of the following metrics:

- **lambda values:** The lambda values (dominant eigenvalues) of the A matrices.
- **max F values:** The maximum values from the F matrices.
- **max U values:** The maximum values from the U matrices.
- **minimum non-zero U values:** The minimum non-zero values from the U matrices.

See Also

Other utility: `plot_matrix()`

Examples

```
mats <- rand_lefko_set(
  n = 10, n_stages = 5, fecundity = c(0, 0, 4, 8, 10),
  archetype = 4, output = "Type1"
)

summarise_mpms(mats)
```

Index