Package ‘Rage’

January 17, 2023

Type Package

Title Life History Metrics from Matrix Population Models

Version 1.3.0

Description Functions for calculating life history metrics using matrix population models ('MPMs'). Described in Jones et al. (2021) <doi:10.1101/2021.04.26.441330>.

License GPL-3

URL https://github.com/jonesor/Rage

BugReports https://github.com/jonesor/Rage/issues

Depends R (>= 3.5.0)

Imports DiagrammeR, MASS, popdemo, stats

Suggests ggplot2, knitr, Rcompadre, rmarkdown, spelling, testthat

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.1.2

NeedsCompilation no

Author Patrick Barks [aut] (<https://orcid.org/0000-0002-5947-8151>),
Danny Buss [ctb],
Pol Capdevila [aut] (<https://orcid.org/0000-0002-2842-4302>),
Hal Caswell [aut] (<https://orcid.org/0000-0003-4394-6894>),
Judy P. Che-Castaldo [aut] (<https://orcid.org/0000-0002-9118-9202>),
John Jackson [aut] (<https://orcid.org/0000-0002-4563-2840>),
Tamora James [aut] (<https://orcid.org/0000-0003-1363-4742>),
Owen Jones [aut, cre] (<https://orcid.org/0000-0001-5720-4686>),
Sam Levin [aut] (<https://orcid.org/0000-0002-3289-9925>),
William K. Petry [aut] (<https://orcid.org/0000-0002-5230-5987>),

Roberto Salguero-Gomez [aut] (<https://orcid.org/0000-0002-6085-4433>),
Caroline Schuette [ctb] (<https://orcid.org/0000-0002-2063-8736>),
Iain Stott [aut] (<https://orcid.org/0000-0003-2724-7436>),
Chelsea C. Thomas [aut] (<https://orcid.org/0000-0002-8155-9353>)

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

Repository CRAN

Date/Publication 2023-01-17 17:50:06 UTC

R topics documented:

`age_from_stage` *Calculate age-specific traits from a matrix population model*

Description

These functions use age-from-stage decomposition methods to calculate age-specific survivorship (`lx`), survival probability (`px`), mortality hazard (`hx`), or reproduction (`mx`) from a matrix population model (MPM). A detailed description of these methods can be found in sections 5.3.1 and 5.3.2 of Caswell (2001). A separate function `mpm_to_table` uses the same methods to calculate a full life table.

Usage

```
mpm_to_mx(matU, matR, start = 1L, xmax = 1000, lx_crit = 0.01, tol = 1e-04)

mpm_to_lx(matU, start = 1L, xmax = 1000, lx_crit = 0.01, tol = 1e-04)

mpm_to_px(matU, start = 1L, xmax = 1000, lx_crit = 0.01, tol = 1e-04)

mpm_to_hx(matU, start = 1L, xmax = 1000, lx_crit = 0.01, tol = 1e-04)
```

Arguments

`matU`	The survival component of a MPM (i.e., a square projection matrix reflecting survival-related transitions; e.g., progression, stasis, and retrogression). Optionally with named rows and columns indicating the corresponding life stage names.
`matR`	The reproductive component of a MPM (i.e., a square projection matrix reflecting transitions due to reproduction; either sexual, clonal, or both). Optionally with named rows and columns indicating the corresponding life stage names.
`start`	The index (or stage name) of the first stage at which the author considers the beginning of life. Defaults to 1. Alternately, a numeric vector giving the starting population vector (in which case `length(start)` must match `ncol(matU)`). See section *Starting from multiple stages*.
`xmax`	Maximum age to which age-specific traits will be calculated (defaults to `1000`).
`lx_crit`	Minimum value of `lx` to which age-specific traits will be calculated (defaults to `0.01`).
`tol`	To account for floating point errors that occasionally lead to values of `lx` slightly greater than 1, values of `lx` within the open interval (`1`, `1 + tol`) are coerced to 1. Defaults to `0.0001`. To prevent coercion, set `tol` to `0`.

Value

A vector

Starting from multiple stages

Rather than specifying argument `start` as a single stage class from which all individuals start life, it may sometimes be desirable to allow for multiple starting stage classes. For example, if users want to start their calculation of age-specific traits from reproductive maturity (i.e., first reproduction), they should account for the possibility that there may be multiple stage classes in which an individual could first reproduce.

To specify multiple starting stage classes, users should specify argument `start` as the desired starting population vector (**n1**), giving the proportion of individuals starting in each stage class (the length of `start` should match the number of columns in the relevant MPM).

See function `mature_distrib` for calculating the proportion of individuals achieving reproductive maturity in each stage class.

Note

Note that the units of time for the returned vectors (i.e., `x`) are the same as the projection interval (`ProjectionInterval`) of the MPM.

The output vector is calculated recursively until the age class (`x`) reaches `xmax` or survivorship (`lx`) falls below `lx_crit`, whichever comes first. To force calculation to `xmax`, set `lx_crit` to `0`. Conversely, to force calculation to `lx_crit`, set `xmax` to `Inf`.

Note that the units of time in returned values (i.e., `x`) are the same as the projection interval (‘ProjectionInterval‘) of the MPM.

Author(s)

Owen R. Jones <jones@biology.sdu.dk>

Roberto Salguero-Gómez <rob.salguero@zoo.ox.ac.uk>

Hal Caswell <h.caswell@uva.nl>

Patrick Barks <patrick.barks@gmail.com>

References

Caswell, H. 2001. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates; 2nd edition. ISBN: 978-0878930968

Jones O. R. 2021. Life tables: Construction and interpretation In: Demographic Methods Across the Tree of Life. Edited by Salguero-Gomez R & Gamelon M. Oxford University Press. Oxford, UK. ISBN: 9780198838609

Preston, S., Heuveline, P., & Guillot, M. 2000. Demography: Measuring and Modeling Population Processes. Wiley. ISBN: 9781557864512

See Also

`lifetable_convert`

Other life tables: `lifetable_convert`, `mpm_to_table()`, `qsd_converge()`

Examples

```
data(mpm1)

# age-specific survivorship
mpm_to_lx(mpm1$matU)
mpm_to_lx(mpm1$matU, start = 2) # starting from stage 2
mpm_to_lx(mpm1$matU, start = "small") # equivalent using named life stages
mpm_to_lx(mpm1$matU, xmax = 10) # to a maximum age of 10
mpm_to_lx(mpm1$matU, lx_crit = 0.05) # to a minimum lx of 0.05

# age-specific survival probability
mpm_to_px(mpm1$matU)

# age-specific mortality hazard
mpm_to_hx(mpm1$matU)

# age-specific fecundity
mpm_to_mx(mpm1$matU, mpm1$matF)


### starting from first reproduction
repstages <- repro_stages(mpm1$matF)
n1 <- mature_distrib(mpm1$matU, start = 2, repro_stages = repstages)

mpm_to_lx(mpm1$matU, start = n1)
mpm_to_px(mpm1$matU, start = n1)
mpm_to_hx(mpm1$matU, start = n1)
mpm_to_mx(mpm1$matU, mpm1$matF, start = n1)
```

`entropy_d` *Calculate Demetrius' entropy from trajectories of age-specific survivorship and fecundity*

Description

This function calculates Demetrius' entropy from vectors of age-specific survivorship (`lx`) and fecundity (`mx`).

Usage

```
entropy_d(lx, mx)
```

Arguments

`lx`	Age-specific survivorship trajectory (a vector of monotonically-declining values in the interval [0,1]).
`mx`	Age-specific fecundity trajectory (a vector of non-negative values).

Value

Demetrius' entropy.

Warning

Note that this function may produce unexpected results if used on partial survivorship and fecundity trajectories. In addition, it is sensitive to the length of the these vectors. We direct users to the functions '`shape_surv`' and '`shape_rep`' which are relatively robust to these issues.

Author(s)

Roberto Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

Patrick Barks <patrick.barks@gmail.com>

References

Demetrius, L., & Gundlach, V. M. 2014. Directionality theory and the entropic principle of natural selection. Entropy 16: 5428-5522.

See Also

Other life history traits: `entropy_k()`, `gen_time()`, `life_expect_mean()`, `longevity()`, `net_repro_rate()`, `repro_maturity`, `shape_rep()`, `shape_surv()`

Examples

```
data(mpm1)

# derive trajectories of lx and mx, starting from stage 2
lx <- mpm_to_lx(mpm1$matU, start = 2)
mx <- mpm_to_mx(mpm1$matU, mpm1$matF, start = 2)

# calculate Demetrius' entropy
entropy_d(lx, mx)
```

`entropy_k` *Calculate Keyfitz's entropy from a trajectory of age-specific survivorship*

Description

Calculate Keyfitz's entropy from a vector of age-specific survivorship (`lx`).

Usage

```
entropy_k(lx, trapeze = FALSE)
```

Arguments

`lx`	Survivorship trajectory (a vector of monotonically-declining values in the interval [0,1]).
`trapeze`	A logical argument indicating whether the composite trapezoid approximation should be used for approximating the definite integral.

Value

Keyfitz's life table entropy.

Warning

Note that this function may produce unexpected results if used on partial survivorship trajectories. In addition, it is sensitive to the length of the survivorship vector. We direct users to the function '`shape_surv`' which is relatively robust to these issues.

Author(s)

Owen R. Jones <jones@biology.sdu.dk>

Roberto Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

References

Keyfitz, N. 1977. Applied Mathematical Demography. New York: Wiley.

Demetrius, L., & Gundlach, V. M. 2014. Directionality theory and the entropic principle of natural selection. Entropy 16: 5428-5522.

See Also

Other life history traits: `entropy_d()`, `gen_time()`, `life_expect_mean()`, `longevity()`, `net_repro_rate()`, `repro_maturity`, `shape_rep()`, `shape_surv()`

Examples

```
data(mpm1)

# derive lx trajectory, starting from stage 2
lx <- mpm_to_lx(mpm1$matU, start = 2)

# calculate Keyfitz' entropy
entropy_k(lx)

# use trapezoid approximation for definite integral
entropy_k(lx, trapeze = TRUE)
```

`gen_time` *Calculate generation time from a matrix population model*

Description

Calculate generation time from a matrix population model. Multiple definitions of the generation time are supported: the time required for a population to increase by a factor of R0 (the net reproductive rate; Caswell (2001), section 5.3.5), the average parent-offspring age difference (Bienvenu & Legendre (2015)), or the expected age at reproduction for a cohort (Coale (1972), p. 18-19).

Usage

```
gen_time(matU, matR, method = c("R0", "age_diff", "cohort"), ...)
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix reflecting survival-related transitions; e.g., progression, stasis, and retrogression).
`matR`	The reproductive component of a matrix population model (i.e., a square projection matrix only reflecting transitions due to reproduction; either sexual, clonal, or both).
`method`	The method used to calculate generation time. Defaults to "R0". See Details for explanation of calculations.
`...`	Additional arguments passed to `net_repro_rate` when `method = "R0"` or `mpm_to_*` when `method = "cohort"`. Ignored when `method = "age_diff"`

Details

There are multiple definitions of generation time, three of which are implemented by this function:

1. `"R0"` (default): This is the number of time steps required for the population to grow by a factor of its net reproductive rate, equal to `log(R0) / log(lambda)`. Here, `R0` is the net reproductive rate (the per-generation population growth rate; Caswell 2001, Sec. 5.3.4), and `lambda` is the population growth rate per unit time (the dominant eigenvalue of `matU + matR`).

2. `"age_diff"`: This is the average age difference between parents and offspring, equal to `(lambda v w) / (v matR w)` (Bienvenu & Legendre (2015)). Here, `lambda` is the population growth rate per unit time (the dominant eigenvalue of `matU + matR`), `v` is a row vector of stage-specific reproductive values (the left eigenvector corresponding to `lambda`), and `w` is a column vector of the stable stage distribution (the right eigenvector corresponding to `lambda`).

3. `"cohort"`: This is the age at which members of a cohort are expected to reproduce, equal to `sum(x lx mx) / sum(lx mx)` (Coale (1972), p. 18-19). Here, `x` is age, `lx` is age-specific survivorship, and `mx` is age-specific fertility. See functions `mpm_to_lx` and `mpm_to_mx` for details about the conversion of matrix population models to life tables.

Value

Returns generation time. If `matU` is singular (often indicating infinite life expectancy), returns `NA`.

Note

Note that the units of time in returned values are the same as the projection interval ('ProjectionInterval') of the MPM.

Author(s)

Patrick Barks <patrick.barks@gmail.com>

William Petry <wpetry@ncsu.edu>

References

Bienvenu, F. & Legendre, S. 2015. A New Approach to the Generation Time in Matrix Population Models. The American Naturalist 185 (6): 834–843. doi:10.1086/681104.

Caswell, H. 2001. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates; 2nd edition. ISBN: 978-0878930968

Coale, A.J. 1972. The Growth and Structure of Human Populations. Princeton University Press. ISBN: 978-0691093574

See Also

Other life history traits: `entropy_d()`, `entropy_k()`, `life_expect_mean()`, `longevity()`, `net_repro_rate()`, `repro_maturity`, `shape_rep()`, `shape_surv()`

Examples

```
data(mpm1)

# calculate generation time
gen_time(matU = mpm1$matU, matR = mpm1$matF) # defaults to "R0" method
gen_time(matU = mpm1$matU, matR = mpm1$matF, method = "age_diff")
gen_time(matU = mpm1$matU, matR = mpm1$matF, method = "cohort", lx_crit = 0.001)
```

`lifetable_convert` *Convert between age-specific survivorship, survival, or mortality hazard*

Description

Convert between vectors of age-specific survivorship (`lx`), survival probability (`px`), or mortality hazard (`hx`). Input vectors must be arranged in order of increasing age, starting with age 0.

Usage

```
lx_to_px(lx)

lx_to_hx(lx)

px_to_lx(px)

px_to_hx(px)

hx_to_lx(hx)

hx_to_px(hx)
```

Arguments

`lx`	Vector of age-specific survivorship.
`px`	Vector of age-specific survival probabilities.
`hx`	Vector of age-specific mortality hazards.

Details

`lx` gives the proportional survivorship to the start of age class `x` (where survivorship at first age class is defined as 1), `px` gives the probability of survival between age `x` and `x+1`, and `hx` gives the time-averaged mortality hazard (also called force of mortality) between age `x` and `x+1`.

Value

A vector.

Note

Note that the units of time for the returned vectors (i.e., `x`) are the same as the (`ProjectionInterval`) of the MPM.

Author(s)

Patrick Barks <patrick.barks@gmail.com>

References

Caswell, H. 2001. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates; 2nd edition. ISBN: 978-0878930968

Caswell, H. 2006. Applications of Markov chains in demography. pp. 319-334 in A.N. Langville and W.J. Stewart (editors) MAM2006: Markov Anniversary Meeting. Boson Books, Raleigh, North Caroline, USA

Ergon, T., Borgan, Ø., Nater, C. R., & Vindenes, Y. 2018. The utility of mortality hazard rates in population analyses. Methods in Ecology and Evolution, 9, 2046-2056. <doi:10.1111/2041-210X.13059>

Horvitz, C. & Tuljapurkar, S. 2008. Stage dynamics, period survival, and mortality plateaus. The American Naturalist 172: 203-2015. <doi:10.1086/589453>

Jones, O. R., Scheuerlein, A., Salguero-Gomez, R., Camarda, C. G., Schaible, R., Casper, B. B., Dahlgren, J. P., Ehrlén, J., García, M. B., Menges, E., Quintana-Ascencio, P. F., Caswell, H., Baudisch, A. & Vaupel, J. 2014. Diversity of ageing across the tree of life. Nature 505, 169-173. <doi:10.1038/nature12789>

Jones O. R. 2021. Life tables: Construction and interpretation In: Demographic Methods Across the Tree of Life. Edited by Salguero-Gomez R & Gamelon M. Oxford University Press. Oxford, UK. ISBN: 9780198838609

Preston, S., Heuveline, P., & Guillot, M. 2000. Demography: Measuring and Modeling Population Processes. Wiley. ISBN: 9781557864512

See Also

Other life tables: `age_from_stage`, `mpm_to_table()`, `qsd_converge()`

Examples

```
lx <- c(1, 0.8, 0.7, 0.5, 0.3, 0.1)

# convert from lx
px <- lx_to_px(lx)
hx <- lx_to_hx(lx)

# convert from px
lx <- px_to_lx(px)
hx <- px_to_hx(px)

# convert from hx
lx <- hx_to_lx(hx)
px <- hx_to_px(hx)
```

`life_expect_mean` *Calculate mean and variance of life expectancy from a matrix population model*

Description

Applies Markov chain approaches to obtain mean and variance of life expectancy from a matrix population model (MPM).

Usage

```
life_expect_mean(matU, start = 1L)

life_expect_var(matU, start = 1L)

life_expect(matU, start = 1L)
```

Arguments

`matU`	The survival component of a MPM (i.e., a square projection matrix reflecting survival-related transitions; e.g., progression, stasis, and retrogression). Optionally with named rows and columns indicating the corresponding life stage names.
`start`	The index (or stage name) of the first stage of the life cycle which the user considers to be the beginning of life. Defaults to `1`. Alternately, a numeric vector giving the starting population vector (in which case `length(start)` must match `ncol(matU)`). See section *Starting from multiple stages*.

Value

Returns life expectancy. If `matU` is singular (often indicating infinite life expectancy), returns `NA`.

Starting from multiple stages

Rather than specifying argument `start` as a single stage class from which all individuals start life, it may sometimes be desirable to allow for multiple starting stage classes. For example, if the user wants to start their calculation of life expectancy from reproductive maturity (i.e., first reproduction), they should account for the possibility that there may be multiple stage classes in which an individual could first reproduce.

To specify multiple starting stage classes, specify argument `start` as the desired starting population vector (**n1**), giving the proportion of individuals starting in each stage class (the length of `start` should match the number of columns in the relevant MPM).

See function `mature_distrib` for calculating the proportion of individuals achieving reproductive maturity in each stage class.

Note

Note that the units of time in returned values are the same as the projection interval (‘ProjectionInterval‘) of the MPM.

Author(s)

Roberto Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

Hal Caswell <hcaswell@whoi.edu>

References

Caswell, H. 2001. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates; 2nd edition. ISBN: 978-0878930968

See Also

Other life history traits: `entropy_d()`, `entropy_k()`, `gen_time()`, `longevity()`, `net_repro_rate()`, `repro_maturity`, `shape_rep()`, `shape_surv()`

Examples

```
data(mpm1)

# mean life expectancy starting from stage class 2
life_expect_mean(mpm1$matU, start = 2)
life_expect_mean(mpm1$matU, start = "small") # equivalent using named life stages

# mean life expectancy starting from first reproduction
rep_stages <- repro_stages(mpm1$matF)
n1 <- mature_distrib(mpm1$matU, start = 2, repro_stages = rep_stages)
life_expect_mean(mpm1$matU, start = n1)

# variance of life expectancy from stage class 1
life_expect_var(mpm1$matU, start = 1)
```

`longevity` *Calculate longevity from a matrix population model*

Description

Calculate longevity (the age *x* at which survivorship for a synthetic cohort falls below some critical proportion) from a matrix population model

Usage

```
longevity(matU, start = 1L, x_max = 1000, lx_crit = 0.01)
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix reflecting survival-related transitions; e.g., progression, stasis, and retrogression). Optionally with named rows and columns indicating the corresponding life stage names.
`start`	The index (or stage name) of the first stage at which the author considers the beginning of life. Defaults to `1`. Alternately, a numeric vector giving the starting population vector (in which case `length(start)` must match `ncol(matU)`). See section *Starting from multiple stages*.
`x_max`	The maximum age, in units of the MPM projection interval, to which survivorship will be calculated. Defaults to `1000`.
`lx_crit`	Proportion of initial cohort remaining before all are considered dead (a value between 0 and 1). Defaults to `0.01`.

Value

Returns longevity, the integer age at which expected survivorship falls below `lx_crit`. If survivorship doesn't reach `lx_crit` by `x_max`, returns `NA` and prints a warning message.

Starting from multiple stages

Rather than specifying argument `start` as a single stage class from which all individuals start life, it may sometimes be desirable to allow for multiple starting stage classes. For example, if we want to start our calculation of longevity from reproductive maturity (i.e., first reproduction), we should account for the possibility that there may be multiple stage classes in which an individual could first reproduce.

To specify multiple starting stage classes, specify argument `start` as the desired starting population vector, giving the proportion of individuals starting in each stage class (the length of `start` should match the number of columns in the relevant MPM).

Note

Note that the units of time in returned values are the same as the (`ProjectionInterval`) of the MPM.

Author(s)

Roberto Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

Hal Caswell <hcaswell@whoi.edu>

References

Caswell, H. 2001. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates; 2nd edition. ISBN: 978-0878930968

Morris, W. F. & Doak, D. F. 2003. Quantitative Conservation Biology: Theory and Practice of Population Viability Analysis. Sinauer Associates, Sunderland, Massachusetts, USA

See Also

`mature_distrib` for calculating the proportion of individuals achieving reproductive maturity in each stage class.

Other life history traits: `entropy_d()`, `entropy_k()`, `gen_time()`, `life_expect_mean()`, `net_repro_rate()`, `repro_maturity`, `shape_rep()`, `shape_surv()`

Examples

```
data(mpm1)

longevity(mpm1$matU, start = 2)
longevity(mpm1$matU, start = "small") # equivalent using named life stages
longevity(mpm1$matU, start = 2, lx_crit = 0.05)

# starting from first reproduction
repstages <- repro_stages(mpm1$matF)
n1 <- mature_distrib(mpm1$matU, start = 2, repro_stages = repstages)
longevity(mpm1$matU, start = n1)
```

`mpm1` *Example matrix population model (MPM)*

Description

An example matrix population model (MPM) used for demonstration and testing purposes. The MPM consists of five stage classes: 'seed', 'small', 'medium', 'large', and 'dormant'.

Usage

```
mpm1
```

Format

A list with two elements:

matU The survival-related component of the MPM.

matF The sexual reproduction component of the MPM.

`mpm_collapse` *Collapse a matrix population model to a smaller number of stages*

Description

Collapse a matrix population model to a smaller number of stages. For instance, to compare properties of multiple projection matrices with different numbers of stages, one might first collapse those matrices to a standardized set of stages (e.g., propagule, pre-reproductive, reproductive, and post-reproductive). The transition rates in the collapsed matrix are a weighted average of the transition rates from the relevant stages of the original matrix, weighted by the relative proportion of each stage class expected at the stable distribution.

Usage

```
mpm_collapse(matU, matF, matC = NULL, collapse)
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix reflecting survival-related transitions; e.g., progression, stasis, and retrogression)
`matF`	The sexual component of a matrix population model (i.e., a square projection matrix reflecting transitions due to sexual reproduction)
`matC`	The clonal component of a matrix population model (i.e., a square projection matrix reflecting transitions due to clonal reproduction). Defaults to `NULL`, indicating no clonal reproduction (i.e., `matC` is a matrix of zeros).

`collapse`	A list giving the mapping between stages of the original matrix and the desired stages of the collapsed matrix (e.g., `list(1, 2:3, 4)`). Original stages may be passed as either indices or stage names corresponding to stage index or name in `matU`, `matF` and `matC`). Names given to the elements of `collapse` are used as stage names in the new, collapsed matrix. See *Missing Stages* for handling of `NA` within `collapse`.

Value

A list with four elements:

`matA`	Collapsed projection matrix
`matU`	Survival component of the collapsed projection matrix
`matF`	Sexual reproduction component of the collapsed projection matrix
`matC`	Clonal reproduction component of the collapsed projection matrix

Missing Stages

The collapsed matrix will always be of dimension `length(collapse)`, even if one or more elements of the `collapse` argument is `NA` (corresponding to a desired stage of the collapsed matrix that is not present in the original matrix). In the collapsed matrix, any row/column corresponding to a missing stage will be coerced to `NA`.

Note

This method of collapsing a matrix population model preserves the equilibrium population growth rate ($lambda$) and relative stable distribution, but is not expected to preserve other traits such as relative reproductive values, sensitivities, net reproductive rates, life expectancy, etc.

Author(s)

Rob Salguero-Gómez <rob.salguero@zoo.ox.ac.uk>

William K. Petry <wpetry@ncsu.edu>

References

Salguero-Gomez, R. & Plotkin, J. B. 2010. Matrix dimensions bias demographic inferences: implications for comparative plant demography. The American Naturalist 176, 710-722. <doi:10.1086/657044>

See Also

`mpm_standardize`

Other transformation: `mpm_rearrange()`, `mpm_split()`, `mpm_standardize()`, `name_stages()`, `repro_stages()`, `standard_stages()`

Examples

```
data(mpm1)

# check which stages reproductive
repro_stages(matR = mpm1$matF)

# collapse reproductive stages (3 and 4) into single stage
mpm_collapse(matU = mpm1$matU, matF = mpm1$matF, collapse = list(1, 2, 3:4, 5))

# use stage names instead, and name stages in the collapsed matrix
mpm_collapse(
  matU = mpm1$matU, matF = mpm1$matF,
  collapse = list(
    seed = "seed", vegetative = "small",
    flowering = c("medium", "large"),
    dormant = "dormant"
  )
)
```

`mpm_rearrange` *Rearrange stages of a matrix population model to segregate reproductive and non-reproductive stages*

Description

Rearrange stages of a matrix population model so that all inter-reproductive stages fall in the final rows/columns of the matrix. This is a preparatory step to collapsing the matrix model into a standardized set of stages (e.g., propagule, pre-reproductive, reproductive, and post-reproductive).

Usage

```
mpm_rearrange(matU, matF, matC = NULL, repro_stages, matrix_stages)
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix reflecting survival-related transitions; e.g., progression, stasis, and retrogression)
`matF`	The sexual component of a matrix population model (i.e., a square projection matrix reflecting transitions due to sexual reproduction)
`matC`	The clonal component of a matrix population model (i.e., a square projection matrix reflecting transitions due to clonal reproduction). Defaults to `NULL`, indicating no clonal reproduction (i.e., `matC` is a matrix of zeros).
`repro_stages`	Logical vector of length `ncol(matU)` indicating which stages are reproductive. Alternatively, a vector of stage indices or stage names of the reproductive classes.
`matrix_stages`	A character vector identifying organized matrix stages.

Value

Returns a list with 6 elements:

`matU`	Rearranged survival matrix
`matF`	Rearranged sexual reproduction matrix
`matC`	Rearranged clonal reproduction matrix
`matrix_stages`	Rearranged vector of organized matrix stages
`repro_stages`	Rearranged logical vector of reproductive stages
`nonRepInterRep`	Numeric index for any rearranged inter-reproductive stages

Author(s)

Rob Salguero-Gómez <rob.salguero@zoo.ox.ac.uk>

See Also

`mpm_standardize`

Other transformation: `mpm_collapse()`, `mpm_split()`, `mpm_standardize()`, `name_stages()`, `repro_stages()`, `standard_stages()`

Examples

```
matU <- rbind(c(0.1,   0,   0,   0,   0),
              c(0.5, 0.2, 0.1,   0,   0),
              c(  0, 0.3, 0.3, 0.1,   0),
              c(  0,   0, 0.4, 0.4, 0.1),
              c(  0,   0,   0, 0.1, 0.4))

matF <- rbind(c(  0, 1.1,   0, 1.6,   0),
              c(  0, 0.8,   0, 0.4,   0),
              c(  0,   0,   0,   0,   0),
              c(  0,   0,   0,   0,   0),
              c(  0,   0,   0,   0,   0))

repro_stages <- c(2, 4)
matrix_stages <- c('prop', 'active', 'active', 'active', 'active')

mpm_rearrange(matU, matF, repro_stages = repro_stages,
              matrix_stages = matrix_stages)
```

`mpm_split` *Convert matrix population model into U, F and C matrices*

Description

Splits a matrix population model into three constituent matrices, **U** (growth and survival processes), **F** (sexual reproduction) and **C** (clonal reproduction). **Warning!** The functionality is very basic: it assumes that sexual reproduction is located in the top row of the matrix, and that everything else is growth or survival (i.e. the **U** matrix). Clonality is assumed to be non-existent.

Usage

```
mpm_split(matA)
```

Arguments

`matA`	A matrix population model (i.e., a square projection matrix).

Value

A list of three matrices: `matU`,`matF` and `matC`. `matC` will always contain only zeros.

Author(s)

Owen R. Jones <jones@biology.sdu.dk>

See Also

Other transformation: `mpm_collapse()`, `mpm_rearrange()`, `mpm_standardize()`, `name_stages()`, `repro_stages()`, `standard_stages()`

Examples

```
matA <- rbind(c(0.1,   0, 5.3, 4.2),
              c(0.5, 0.2, 0.1,   0),
              c(  0, 0.3, 0.3, 0.1),
              c(  0,   0, 0.5, 0.6))

mpm_split(matA)
```

mpm_standardize *Transform a matrix population model to a standardized form*

Description

Transform a matrix population model to a standardized set of stage classes (e.g., propagule, pre-reproductive, reproductive, and post-reproductive). The transition rates in the standardized matrix are a weighted mean of the transition rates and per-capita reproductive values from the relevant stages of the original matrix, weighted by the relative proportion of each stage class expected at the stable distribution.

Usage

```
mpm_standardize(matU, matF, matC = NULL, repro_stages, matrix_stages)

mpm_standardise(matU, matF, matC = NULL, repro_stages, matrix_stages)
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix reflecting survival-related transitions; e.g. progression, stasis, and retrogression).
`matF`	The sexual component of a matrix population model (i.e., a square projection matrix reflecting transitions due to sexual reproduction).
`matC`	The clonal component of a matrix population model (i.e., a square projection matrix reflecting transitions due to clonal reproduction). Defaults to `NULL`, indicating no clonal reproduction (i.e., `matC` is a matrix of zeros).
`repro_stages`	Logical vector of length `ncol(matU)` indicating which stages are reproductive. Alternatively, a vector of stage indices or stage names of the reproductive classes.
`matrix_stages`	Character vector of matrix stage types (e.g., "propagule", "active", or "dormant").

Details

This function is a wrapper for the functions `mpm_rearrange`, `standard_stages` and `mpm_collapse`, which it calls in sequence.

Value

A list with four elements reflecting the standardized matrix and its components:

`matA`	Standardized projection matrix
`matU`	Survival component of the standardized projection matrix
`matF`	Sexual reproduction component of the standardized projection matrix
`matC`	Clonal reproduction component of the standardized projection matrix

Missing Stages

The returned standardized matrix will always be of dimension `4`, even if one or more standardized stages is missing from the original matrix population model. If a standardized stage is missing, the corresponding row/column of the standardized matrix will be coerced to `NA`.

Note

The method used by this function to collapse a matrix population model preserves the equilibrium population growth rate (λ) and relative stable distribution, but is not expected to preserve other demographic characteristics such as relative reproductive value, sensitivities, net reproductive rate, life expectancy, etc.

Author(s)

Rob Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

See Also

Other transformation: `mpm_collapse()`, `mpm_rearrange()`, `mpm_split()`, `name_stages()`, `repro_stages()`, `standard_stages()`

Examples

```
matU <- rbind(c(0.1,   0,   0,   0,   0),
              c(0.5, 0.2, 0.1,   0,   0),
              c(  0, 0.3, 0.3, 0.1,   0),
              c(  0,   0, 0.4, 0.4, 0.1),
              c(  0,   0,   0, 0.1, 0.4))

matF <- rbind(c(  0, 1.1,   0, 1.6,   0),
              c(  0, 0.8,   0, 0.4,   0),
              c(  0,   0,   0,   0,   0),
              c(  0,   0,   0,   0,   0),
              c(  0,   0,   0,   0,   0))

matC <- rbind(c(  0, 0.6,   0, 0.5,   0),
              c(  0, 0.1,   0, 0.3,   0),
              c(  0,   0,   0,   0,   0),
              c(  0,   0,   0,   0,   0),
              c(  0,   0,   0,   0,   0))

repro_stages <- c(2, 4)
matrix_stages <- c('prop', 'active', 'active', 'active', 'active')

mpm_standardize(matU, matF, matC, repro_stages, matrix_stages)
```

`mpm_to_table` *Generate a life table from a matrix population model*

Description

This function uses age-from-stage decomposition methods to generate a life table from a matrix population model. A detailed description of these methods can be found in section 5.3 "Age-specific traits from stage-specific models" of Caswell (2001).

Usage

```
mpm_to_table(
  matU,
  matF = NULL,
  matC = NULL,
  start = 1L,
  xmax = 1000,
  lx_crit = 0.01,
  radix = 1
)
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix reflecting survival-related transitions; e.g., progression, stasis, and/or retrogression). Optionally with named rows and columns indicating the corresponding life stage names.
`matF`	(Optional) The sexual component of a matrix population model (i.e., a square projection matrix reflecting transitions due to sexual reproduction). Optionally with named rows and columns indicating the corresponding life stage names.
`matC`	(Optional) The clonal component of a matrix population model (i.e., a square projection matrix reflecting transitions due to clonal reproduction). Optionally with named rows and columns indicating the corresponding life stage names.
`start`	The index (or stage name) of the first stage at which the author considers the beginning of life. Defaults to `1`. Alternately, a numeric vector giving the starting population vector (in which case `length(start)` must match `ncol(matU)`). See section *Starting from multiple stages*.
`xmax`	Maximum age to which the life table will be calculated (defaults to `1000`). Time steps are in the same units as the matrix population model (see MatrixPeriodicity metadata variable COM(P)ADRE).
`lx_crit`	Minimum value of lx to which age-specific traits will be calculated (defaults to `0.01`).
`radix`	The starting number of individuals in the synthetic life table (defaults to `1`). If `radix` is set to 1, a simplified life table is produced.

Value

A `data.frame` containing a variable number columns, depending on input variables. Columns include:

`x`	age at the start of the age interval `[x, x+1)`
`Nx`	The number of individuals alive at age x. The initial number is set with `radix`
`Dx`	proportion of original cohort dying during the age interval `[x, x+1)`
`lx`	survivorship, defined as the proportion of initial cohort surviving to the start of age interval `[x, x+1)`
`dx`	proportion of original cohort dying in the age interval `[x, x+1)`
`ax`	The average time survived within the interval by those that die during the age interval `[x, x+1)`. Assumed to be 0.5
`hx`	force of mortality (hazard) during the age interval `[x, x+1)`
`qx`	probability of death during the interval `[x, x+1)` for those entering the interval
`px`	probability of survival for the interval `[x, x+1)` for those entering the interval
`Lx`	total person-years lived during the interval `[x, x+1)`
`Tx`	total person years lived beyond age x
`ex`	remaining life expectancy at age x

If `matF` is provided, also includes:

`mx`	per-capita rate of sexual reproduction during the interval `[x, x+1)`
`lxmx`	expected number of sexual offspring per original cohort member produced during the interval `[x, x+1)`

If `matC` is provided, also includes:

`cx`	per-capita rate of clonal reproduction during the interval `[x, x+1)`
`lxcx`	expected number of clonal offspring per original cohort member produced during the interval `[x, x+1)`

If both `matF` and `matC` are provided, also includes:

`mxcx`	per-capita rate of total reproduction (sexual + clonal) during the interval `[x, x+1)`
`lxmxcx`	expected number of total offspring (sexual + clonal) per original cohort member produced during the interval `[x, x+1)`

Starting from multiple stages

Rather than specifying argument `start` as a single stage class from which all individuals start life, it may sometimes be desirable to allow for multiple starting stage classes. For example, if the user wants to start the calculation of age-specific traits from reproductive maturity (i.e., first reproduction), the user should account for the possibility that there may be multiple stage classes in which an individual could first reproduce.

To specify multiple starting stage classes, specify argument `start` as the desired starting population vector (**n1**), giving the proportion of individuals starting in each stage class (the length of `start` should match the number of columns in the relevant MPM).

See function `mature_distrib` for calculating the proportion of individuals achieving reproductive maturity in each stage class.

Note

The life table is calculated recursively until the age class (x) reaches `xmax` or survivorship (lx) falls below `lx_crit` — whichever comes first. To force calculation to `xmax`, set `lx_crit = 0`. Conversely, to force calculation to `lx_crit`, set `xmax = Inf`.

The life table calculations assume that the final age interval is closed and that all remaining individuals die in this interval. Therefore, for this interval, the probability of death `qx` is 1, the probability of survival `px` is 0 and, because we assume that deaths are evenly distributed during the interval, the remaining life expectancy for individuals at the start of the interval is 0.5.

If `lx_crit` is sufficiently small that only a very small proportion of the cohort reach this age (i.e., < 0.05), this should have minimal impact on results. Nevertheless, for many analyses, the final row of the life table should be treated with caution and perhaps removed from subsequent analyses.

Note that the units of time (e.g.. ‘x‘ and ‘ex‘) in the returned life table are the same as the projection interval (‘ProjectionInterval‘) of the MPM.

Author(s)

Owen R. Jones <jones@biology.sdu.dk>

Roberto Salguero-Gómez <rob.salguero@zoo.ox.ac.uk>

Hal Caswell <h.caswell@uva.nl>

References

Caswell, H. 2001. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates; 2nd edition. ISBN: 978-0878930968

Caswell, H. 2006. Applications of Markov chains in demography. pp. 319-334 in A.N. Langville and W.J. Stewart (editors) MAM2006: Markov Anniversary Meeting. Boson Books, Raleigh, North Caroline, USA

Horvitz, C. & Tuljapurkar, S. 2008. Stage dynamics, period survival, and mortality plateaus. The American Naturalist 172: 203-2015. <doi:10.1086/589453>

Jones, O. R., Scheuerlein, A., Salguero-Gomez, R., Camarda, C. G., Schaible, R., Casper, B. B., Dahlgren, J. P., Ehrlén, J., García, M. B., Menges, E., Quintana-Ascencio, P. F., Caswell, H., Baudisch, A. & Vaupel, J. 2014. Diversity of ageing across the tree of life. Nature 505, 169-173. <doi:10.1038/nature12789>

Jones O. R. 2021. Life tables: Construction and interpretation In: Demographic Methods Across the Tree of Life. Edited by Salguero-Gomez R & Gamelon M. Oxford University Press. Oxford, UK. ISBN: 9780198838609

Preston, S., Heuveline, P., & Guillot, M. 2000. Demography: Measuring and Modeling Population Processes. Wiley. ISBN: 9781557864512

See Also

Other life tables: `age_from_stage`, `lifetable_convert`, `qsd_converge()`

Examples

```
data(mpm1)

mpm_to_table(matU = mpm1$matU, start = 2, xmax = 15)
mpm_to_table(matU = mpm1$matU, start = "small", xmax = 15) # equivalent using named life stages
mpm_to_table(matU = mpm1$matU, matF = mpm1$matF, start = 2, xmax = 15)

### starting from first reproduction
repStages <- repro_stages(mpm1$matF)
n1 <- mature_distrib(matU = mpm1$matU, start = 2, repro_stages = repStages)
mpm_to_table(matU = mpm1$matU, start = n1)
```

`name_stages` *Add stage names to matrices*

Description

Adds user-supplied or automatically-generated stage names to a matrix population model (MPM).

Usage

```
name_stages(mat, names = NULL, prefix = "stage_", left_pad = TRUE)
```

Arguments

`mat`	An MPM, either as a single matrix or list of matrices.
`names`	A character vector specifying the name of each life stage, in order. If provided, `prefix` and `left_pad` arguments are ignored.
`prefix`	A string to be pre-pended to the stage number when automatically naming stages. Defaults to `stage_`.
`left_pad`	Logical, whether to pre-pend `0` to stage names such that all stage numbers have equal length, enabling lexicographic sorting. For example, stage `1` becomes `01` for matrices with 10-99 stages, `001` for matrices with 100-999 stages, and so on. Defaults to `TRUE`.

Value

The input matrix or matrices with named rows and columns.

Author(s)

William K. Petry <wpetry@ncsu.edu>

See Also

Other transformation: `mpm_collapse()`, `mpm_rearrange()`, `mpm_split()`, `mpm_standardize()`, `repro_stages()`, `standard_stages()`

Examples

```
matU <- rbind(
  c(0.0, 0.0, 0.0),
  c(0.3, 0.1, 0.0),
  c(0.0, 0.5, 0.8)
)
# (semi)automated naming
name_stages(matU)
name_stages(matU, prefix = "s")
# custom stage names
name_stages(matU, names = c("small", "medium", "large"))
# overwrite existing stage names
data(mpm1)
name_stages(mpm1)
```

`net_repro_rate` *Calculate net reproductive value (R0) from a matrix population model*

Description

Calculate net reproductive value (R0) from a matrix population model. The net reproduction value (R0) is the mean number of recruits produced during the mean life expectancy of an individual. See section 5.3.5 of Caswell (2001).

Usage

```
net_repro_rate(matU, matR, start = 1, method = "generation")
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix reflecting survival-related transitions; e.g. progression, stasis, and retrogression). Optionally with named rows and columns indicating the corresponding life stage names.
`matR`	The reproductive component of a matrix population model (i.e., a square projection matrix only reflecting transitions due to reproduction; either sexual, clonal, or both). Optionally with named rows and columns indicating the corresponding life stage names.
`start`	Index (or stage name) of the first stage at which the author considers the beginning of life. Only used if `method = "start"`. Defaults to 1.
`method`	The method used to calculate net reproductive value, either `"generation"` or `"start"`. Defaults to `"generation"`. See Details.

Details

The `method` argument controls how net reproductive rate is calculated.

If `method = "generation"`, net reproductive value is calculated as the per-generation population growth rate (i.e., the dominant eigenvalue of `matR %*% N`, where `N` is the fundamental matrix). See Caswell (2001) Section 5.3.4.

If `method = "start"`, net reproductive value is calculated as the expected lifetime production of offspring that start life in stage `start`, by an individual also starting life in stage `start` (i.e., `(matR %*% N)[start,start]`).

If offspring only arise in stage `start`, the two methods give the same result.

Value

Returns the net reproductive value. If `matU` is singular (often indicating infinite life expectancy), returns `NA`.

Author(s)

Roberto Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

Hal Caswell <h.caswell@uva.nl>

References

Caswell, H. 2001. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates; 2nd edition. ISBN: 978-0878930968

See Also

Other life history traits: `entropy_d()`, `entropy_k()`, `gen_time()`, `life_expect_mean()`, `longevity()`, `repro_maturity`, `shape_rep()`, `shape_surv()`

Examples

```
data(mpm1)

net_repro_rate(mpm1$matU, mpm1$matF)

# calculate R0 using the start method, specifying either the life stage index
# or name
net_repro_rate(mpm1$matU, mpm1$matF, method = "start", start = 2)
net_repro_rate(mpm1$matU, mpm1$matF, method = "start", start = "small")
```

`perturb_matrix` *Perturbation analysis of a matrix population model*

Description

Perturbs elements within a matrix population model and measures the response (sensitivity or elasticity) of the per-capita population growth rate at equilibrium (λ), or, with a user-supplied function, any other demographic statistic.

Usage

```
perturb_matrix(
  matA,
  pert = 1e-06,
  type = "sensitivity",
  demog_stat = "lambda",
  ...
)
```

Arguments

`matA`	A matrix population model (i.e., a square projection matrix).
`pert`	Magnitude of the perturbation. Defaults to `1e-6`.
`type`	Whether to return `sensitivity` or `elasticity` values.
`demog_stat`	The demographic statistic to be used, as in "the sensitivity/elasticity of `demog_stat` to matrix element perturbations." Defaults to the per-capita population growth rate at equilibrium (λ). Also accepts a user-supplied function that performs a calculation on a projection matrix and returns a single numeric value.
`...`	Additional arguments passed to the function `demog_stat`

Value

A sensitivity or elasticity matrix.

Author(s)

Rob Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

References

Caswell, H. 2001. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates; 2nd edition. ISBN: 978-0878930968

See Also

Other perturbation analysis: `perturb_stochastic()`, `perturb_trans()`, `perturb_vr()`, `pop_vectors()`

Examples

```
matA <- rbind(
  c(0.1, 0, 1.5, 4.6),
  c(0.5, 0.2, 0.1, 0),
  c(0, 0.3, 0.3, 0.1),
  c(0, 0, 0.5, 0.6)
)

perturb_matrix(matA)

# use a larger perturbation than the default
perturb_matrix(matA, pert = 0.01)

# calculate the sensitivity/elasticity of the damping ratio to perturbations
damping <- function(matA) { # define function for damping ratio
  eig <- eigen(matA)$values
  dm <- rle(Mod(eig))$values
  return(dm[1] / dm[2])
}

perturb_matrix(matA, demog_stat = "damping")
```

`perturb_stochastic` *Calculate stochastic elasticities from a time-series of matrix population models and corresponding population vectors*

Description

Calculate stochastic elasticities given a time-series of matrix population models and corresponding population vectors, using the method described in Haridas et al. (2009).

Usage

```
perturb_stochastic(X_t, u_t)
```

Arguments

`X_t`	A list of matrix population models
`u_t`	A list of corresponding population vectors

Value

A list of three matrices:

`E`	matrix of stochastic elasticities
`E_mu`	matrix of stochastic elasticities to mean transition rates
`E_sigma`	matrix of stochastic elasticities to the variance in transition rates

Author(s)

Patrick Barks <patrick.barks@gmail.com>

References

Haridas, C. V., Tuljapurkar, S., & Coulson, T. 2009. Estimating stochastic elasticities directly from longitudinal data. Ecology Letters, 12, 806-812. <doi:10.1111/j.1461-0248.2009.01330.x>

See Also

Other perturbation analysis: `perturb_matrix()`, `perturb_trans()`, `perturb_vr()`, `pop_vectors()`

Examples

```
# generate list of random MPMs
N <- 20 # number of years
s <- 3 # matrix dimension
X <- list() # matrix population model at time t
u <- list() # population vector at time t

for (t in 1:N) {
  X[[t]] <- matrix(runif(s^2), nrow = s, ncol = s)
}

# derive corresponding series of population vectors
u <- pop_vectors(X)

# calculate stochastic elasticities
perturb_stochastic(X, u)
```

`perturb_trans` *Perturbation analysis of transition types within a matrix population model*

Description

Calculates the summed sensitivities or elasticities for various transition types within a matrix population model (MPM), including stasis, retrogression, progression, fecundity, and clonality.

Sensitivities or elasticities are calculated by perturbing elements of the MPM and measuring the response of the per-capita population growth rate at equilibrium (λ), or, with a user-supplied function, any other demographic statistic.

Usage

```
perturb_trans(
  matU,
  matF,
  matC = NULL,
  posU = matU > 0,
  posF = matF > 0,
  posC = matC > 0,
  exclude_row = NULL,
  exclude_col = NULL,
  pert = 1e-06,
  type = "sensitivity",
  demog_stat = "lambda",
  ...
)
```

Arguments

`matU`	The survival component submatrix of a MPM (i.e., a square projection matrix reflecting survival-related transitions; e.g., progression, stasis, and retrogression).
`matF`	The sexual component submatrix of a MPM (i.e., a square projection matrix reflecting transitions due to sexual reproduction).
`matC`	The clonal component submatrix of a MPM (i.e., a square projection matrix reflecting transitions due to clonal reproduction). Defaults to `NULL`, indicating no clonal reproduction possible.
`posU`	A logical matrix of the same dimension as `matU`, with elements indicating whether a given `matU` transition is possible (`TRUE`) or not (`FALSE`). Defaults to `matU > 0` (see Details).
`posF`	A logical matrix of the same dimension as `matF`, with elements indicating whether a given `matF` transition is possible (`TRUE`) or not (`FALSE`). Defaults to `matF > 0` (see Details).
`posC`	A logical matrix of the same dimension as `matC`, with elements indicating whether a given `matC` transition is possible (`TRUE`) or not (`FALSE`). Defaults to `matC > 0` (see Details).
`exclude_row`	A vector of row indices or stages names indicating stages for which transitions *to* the stage should be excluded from perturbation analysis. Alternatively, a logical vector of length `nrow(matU)` indicating which stages to include `TRUE` or exclude `FALSE` from the calculation. See section *Excluding stages*.
`exclude_col`	A vector of column indices or stages names indicating stages for which transitions *to* the stage should be excluded from perturbation analysis. Alternatively, a logical vector of length `ncol(matU)` indicating which stages to include `TRUE` or exclude `FALSE` from the calculation. See section *Excluding stages*.
`pert`	The magnitude of the perturbation (defaults to `1e-6`).
`type`	An argument defining whether to return ‘sensitivity‘ or ‘elasticity‘ values. Defaults to ‘sensitivity‘.

`demog_stat`	An argument defining which demographic statistic should be used, as in "the sensitivity/elasticity of `demog_stat` to matrix element perturbations." Defaults to the per-capita population growth rate at equilibrium ($lambda$). Also accepts a user-supplied function that performs a calculation on a MPM and returns a single numeric value.
`...`	Additional arguments passed to the function `demog_stat`.

Details

A transition rate of `0` within a matrix population model can either indicate that the transition is not possible in the given life cycle (e.g., tadpoles never revert to eggs), or that the transition is possible but was estimated to be `0` in the relevant population and time period. Because transition rates of zero *do* generally yield non-zero sensitivities, it is important to distinguish between structural (i.e. impossible) zeros and sampled zeros when summing multiple sensitivities for a given process (e.g., progression/growth).

By default, the `perturb_` functions assume that a transition rate of `0` indicates an impossible transition, in which case the sensitivity for that transition will not be included in any calculation. Specifically, the arguments `posX` are specified by the logical expression `(matX > 0)`. If the matrix population model includes transitions that are possible but estimated to be `0`, users should specify the `posX` argument(s) manually.

If there are no possible transitions for a given process (e.g., clonality, in many species), the value of sensitivity or elasticity returned for that process will be `NA`.

Value

A list with 5 elements:

`stasis`	The sensitivity or elasticity of `demog_stat` to stasis.
`retrogression`	The sensitivity or elasticity of `demog_stat` to retrogression.
`progression`	The sensitivity or elasticity of `demog_stat` to progression.
`fecundity`	The sensitivity or elasticity of `demog_stat` to sexual fecundity.
`clonality`	The sensitivity or elasticity of `demog_stat` to clonality.

Excluding stages

It may be desirable to exclude one or more stages from the calculation. For instance, we might not believe that 'progression' to a dormant stage class truly reflects progression. In this case we could exclude transitions *to* the dormant stage class using the argument `exclude_row`. We may or may not want to ignore progression transitions *from* the dormant stage class, which can be done in a similar way using the argument `exclude_col`. The `exclude_` arguments simply set the relevant row or column of the `posX` arguments to `FALSE`, to prevent those transitions from being used in subsequent calculations.

Author(s)

Rob Salguero-Gómez <rob.salguero@zoo.ox.ac.uk>

Patrick Barks <patrick.barks@gmail.com>

See Also

Other perturbation analysis: `perturb_matrix()`, `perturb_stochastic()`, `perturb_vr()`, `pop_vectors()`

Examples

```
matU <- rbind(
  c(0.1, 0, 0, 0),
  c(0.5, 0.2, 0.1, 0),
  c(0, 0.3, 0.3, 0.1),
  c(0, 0, 0.5, 0.6)
)

matF <- rbind(
  c(0, 0, 1.1, 1.6),
  c(0, 0, 0.8, 0.4),
  c(0, 0, 0, 0),
  c(0, 0, 0, 0)
)


perturb_trans(matU, matF)

# Use a larger perturbation than the default of 1e-6.
perturb_trans(matU, matF, pert = 0.01)

# Calculate the sensitivity/elasticity of the damping ratio to perturbations.
# First, define function for damping ratio:
damping <- function(matA) {
  eig <- eigen(matA)$values
  dm <- rle(Mod(eig))$values
  return(dm[1] / dm[2])
}

# Second, run the perturbation analysis using demog_stat = "damping".
perturb_trans(matU, matF, demog_stat = "damping")
```

`perturb_vr` *Perturbation analysis of vital rates in a matrix population model*

Description

Perturbs lower-level vital rates within a matrix population model and measures the response (sensitivity or elasticity) of the per-capita population growth rate at equilibrium (λ), or, with a user-supplied function, any other demographic statistic.

These decompositions assume that all transition rates are products of a stage-specific survival term (column sums of `matU`) and a lower level vital rate that is conditional on survival (growth, shrinkage, stasis, dormancy, or reproduction). Reproductive vital rates that are not conditional on survival (i.e., within a stage class from which there is no survival) are also allowed.

Usage

```
perturb_vr(
  matU,
  matF,
  matC = NULL,
  pert = 1e-06,
  type = "sensitivity",
  demog_stat = "lambda",
  ...
)
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix reflecting survival-related transitions; e.g., progression, stasis, and retrogression).
`matF`	The sexual component of a matrix population model (i.e., a square projection matrix reflecting transitions due to sexual reproduction).
`matC`	The clonal component of a matrix population model (i.e., a square projection matrix reflecting transitions due to clonal reproduction). Defaults to `NULL`, indicating no clonal reproduction (i.e., `matC` is a matrix of zeros).
`pert`	Magnitude of the perturbation. Defaults to `1e-6`.
`type`	Whether to return `sensitivity` or `elasticity` values. Defaults to `sensitivity`.
`demog_stat`	The demographic statistic to be used, as in "the sensitivity/elasticity of `demog_stat` to vital rate perturbations." Defaults to the per-capita population growth rate at equilibrium (λ). Also accepts a user-supplied function that performs a calculation on a projection matrix and returns a single numeric value.
`...`	Additional arguments passed to the function `demog_stat`.

Value

A list with 5 elements:

`survival`	sensitivity or elasticity of `demog_stat` to survival
`growth`	sensitivity or elasticity of `demog_stat` to growth
`shrinkage`	sensitivity or elasticity of `demog_stat` to shrinkage
`fecundity`	sensitivity or elasticity of `demog_stat` to sexual fecundity
`clonality`	sensitivity or elasticity of `demog_stat` to clonality

Author(s)

Rob Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

Patrick Barks <patrick.barks@gmail.com>

See Also

Other perturbation analysis: `perturb_matrix()`, `perturb_stochastic()`, `perturb_trans()`, `pop_vectors()`

Examples

```
matU <- rbind(
  c(0.1, 0, 0, 0),
  c(0.5, 0.2, 0.1, 0),
  c(0, 0.3, 0.3, 0.1),
  c(0, 0, 0.5, 0.6)
)

matF <- rbind(
  c(0, 0, 1.1, 1.6),
  c(0, 0, 0.8, 0.4),
  c(0, 0, 0, 0),
  c(0, 0, 0, 0)
)

perturb_vr(matU, matF)

# use elasticities rather than sensitivities
perturb_vr(matU, matF, type = "elasticity")

# use a larger perturbation than the default
perturb_vr(matU, matF, pert = 0.01)

# calculate the sensitivity/elasticity of the damping ratio to vital rate
#  perturbations
damping <- function(matA) { # define function for damping ratio
  eig <- eigen(matA)$values
  dm <- rle(Mod(eig))$values
  return(dm[1] / dm[2])
}

perturb_vr(matU, matF, demog_stat = "damping")
```

`plot_life_cycle` *Plot a life cycle diagram from a matrix population model*

Description

Plots the life cycle diagram illustrated by a matrix population model. This function processes the matrix model and passes the information to the graphViz function in DiagrammeR. See `http://rich-iannone.github.io/DiagrammeR/`.

Usage

```
plot_life_cycle(
  matA,
  stages,
  title = NULL,
```

```
    shape = "egg",
    fontsize = 10,
    nodefontsize = 12,
    edgecol = "grey"
  )
```

Arguments

`matA`	A matrix population model (i.e., a square projection matrix)
`stages`	Optional vector of stage class labels. If missing, it first attempts to infer them from `dimnames(matA)`. If these are also `NULL`, then reverts to integers `1:ncol(A)`.
`title`	Optional title for the plot. Defaults to `NULL`.
`shape`	The shape to be used for the stages of the diagram. Any node shape accepted by `graphViz` is acceptable.
`fontsize`	Size of the font used in the diagram.
`nodefontsize`	Size of the font used in the node part of the diagram.
`edgecol`	Colour of the arrows in the diagram.

Value

An object of class `grViz` representing the life cycle diagram

Author(s)

Owen R. Jones <jones@biology.sdu.dk>

Examples

```
matA <- rbind(
  c(0.1, 0, 0, 0, 1.4),
  c(0.5, 0.2, 0, 0, 0),
  c(0, 0.3, 0.3, 0, 0),
  c(0, 0, 0.4, 0.4, 0.1),
  c(0, 0, 0, 0.1, 0.4)
)

plot_life_cycle(matA)
```

`pop_vectors` *Derive a hypothetical set of population vectors corresponding to a time-series of matrix population models*

Description

Derive a hypothetical set of population vectors (i.e. population size distributions across stages) given a time-series of matrix population models (MPMs), by taking the stable stage distribution of the mean matrix as the starting vector (or optionally, a uniform or random starting vector), and deriving subsequent vectors through recursive population projection.

Usage

```
pop_vectors(A, start = "stable.stage")
```

Arguments

`A`	A list of MPMs (i.e., square population projection matrices).
`start`	Method to derive the first population vector in the series. Either `stable.stage` to use the stable stage distribution of the mean matrix as the starting vector, `uniform` to use a uniform starting vector (all elements equal), or `random` to use a randomly-generated starting vector. Defaults to the stable stage distribution.

Details

This function is useful for providing population vectors as input to the `perturb_stochastic` function which calculates stochastic elasticities given a time-series of matrix population models and corresponding population vectors, using the method described in Haridas et al. (2009).

Value

A list of population vectors

Author(s)

Patrick Barks <patrick.barks@gmail.com>

References

Haridas, C. V., Tuljapurkar, S., & Coulson, T. 2009. Estimating stochastic elasticities directly from longitudinal data. Ecology Letters, 12, 806-812. <doi:10.1111/j.1461-0248.2009.01330.x>

See Also

Other perturbation analysis: `perturb_matrix()`, `perturb_stochastic()`, `perturb_trans()`, `perturb_vr()`

Examples

```
# generate list of matrices
matA_l <- replicate(5, matrix(runif(9), 3, 3), simplify = FALSE)

# calculate corresponding population vectors
pop_vectors(matA_l)
pop_vectors(matA_l, start = "uniform")
pop_vectors(matA_l, start = "random")
```

`qsd_converge` *Calculate time to reach quasi-stationary stage distribution*

Description

Calculates the time for a cohort projected with a matrix population model to reach a defined quasi-stationary stage distribution.

Usage

```
qsd_converge(mat, start = 1L, conv = 0.01, N = 100000L)
```

Arguments

`mat`	A matrix population model, or component thereof (i.e., a square projection matrix). Optionally with named rows and columns indicating the corresponding life stage names.
`start`	The index (or stage name) of the first stage at which the author considers the beginning of life. Defaults to `1`. Alternately, a numeric vector giving the starting population vector (in which case `length(start)` must match `ncol(matU)`). See section *Starting from multiple stages*.
`conv`	Proportional distance threshold from the stationary stage distribution indicating convergence. For example, this value should be `0.01` if the user wants to obtain the time step when the stage distribution is within a distance of 1% of the stationary stage distribution.
`N`	Maximum number of time steps over which the population will be projected. Time steps are in the same units as the matrix population model (see `AnnualPeriodicity` column in COM(P)ADRE metadata). Defaults to `100,000`.

Details

Some matrix population models are parameterised with a stasis loop at the largest/most-developed stage class, which can lead to artefactual plateaus in the mortality or fertility trajectories derived from such models. These plateaus occur as a projected cohort approaches its stationary stage distribution (SSD). Though there is generally no single time point at which the SSD is reached, we can define a quasi-stationary stage distribution (QSD) based on a given distance threshold from the SSD, and calculate the number of time steps required for a cohort to reach the QSD. This quantity can then be used to subset age trajectories of mortality or fertility to periods earlier than the QSD, so as to avoid artefactual plateaus in mortality or fertility.

Starting from multiple stages

Rather than specifying argument `start` as a single stage class from which all individuals start life, it may sometimes be desirable to allow for multiple starting stage classes. For example, if we want to start our calculation of QSD from reproductive maturity (i.e., first reproduction), we should account for the possibility that there may be multiple stage classes in which an individual could first reproduce.

To specify multiple starting stage classes, specify argument `start` as the desired starting population vector, giving the proportion of individuals starting in each stage class (the length of `start` should match the number of columns in the relevant MPM).

Value

An integer indicating the first time step at which the quasi-stationary stage distribution is reached (or an `NA` and a warning if the quasi-stationary distribution is not reached).

Note

The time required for a cohort to reach its QSD depends on the initial population vector of the cohort (for our purposes, the starting stage class), and so does not fundamentally require an ergodic matrix (where the long-term equilibrium traits are independent of the initial population vector). However, methods for efficiently calculating the stationary stage distribution (SSD) generally do require ergodicity.

If the supplied matrix (`mat`) is non-ergodic, `qsd_converge` first checks for stage classes with no connection (of any degree) from the starting stage class specified by argument `start`, and strips such stages from the matrix. These unconnected stages have no impact on age-specific traits that we might derive from the matrix (given the specified starting stage), but often lead to non-ergodicity and therefore prevent the reliable calculation of SSD. If the reduced matrix is ergodic, the function internally updates the starting stage class and continues with the regular calculation. Otherwise, if the matrix cannot be made ergodic, the function will return `NA` with a warning.

Author(s)

Hal Caswell <h.caswell@uva.nl>

Owen Jones <jones@biology.sdu.dk>

Roberto Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

Patrick Barks <patrick.barks@gmail.com>

References

Caswell, H. 2001. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates; 2nd edition. ISBN: 978-0878930968

Horvitz, C. C., & Tuljapurkar, S. 2008. Stage dynamics, period survival, and mortality plateaus. The American Naturalist, 172(2), 203–215.

Jones, O. R., Scheuerlein, A., Salguero-Gomez, R., Camarda, C. G., Schaible, R., Casper, B. B., Dahlgren, J. P., Ehrlén, J., García, M. B., Menges, E., Quintana-Ascencio, P. F., Caswell, H., Baudisch, A. & Vaupel, J. 2014. Diversity of ageing across the tree of life. Nature 505, 169-173. <doi:10.1038/nature12789>

Salguero-Gomez R. 2018. Implications of clonality for ageing research. Evolutionary Ecology, 32, 9-28. <doi:10.1007/s10682-017-9923-2>

See Also

`mature_distrib` for calculating the proportion of individuals achieving reproductive maturity in each stage class.

Other life tables: `age_from_stage`, `lifetable_convert`, `mpm_to_table()`

Examples

```
data(mpm1)

# starting stage = 2 (i.e., "small")
qsd_converge(mpm1$matU, start = 2)
qsd_converge(mpm1$matU, start = "small") # equivalent using named life stages

# convergence threshold = 0.001
qsd_converge(mpm1$matU, start = 2, conv = 0.001)

# starting from first reproduction
repstages <- repro_stages(mpm1$matF)
n1 <- mature_distrib(mpm1$matU, start = 2, repro_stages = repstages)
qsd_converge(mpm1$matU, start = n1)
```

`repro_maturity` *Age of reproductive maturity*

Description

Apply Markov chain approaches to compute age-specific trajectory of reproduction for individuals in a matrix population model. Includes functions to calculate the probability of achieving reproductive maturity (`mature_prob`), mean age at first reproduction (`mature_age`), and distribution of individuals first achieving reproductive maturity among stage class (`mature_distrib`).

Usage

```
mature_prob(matU, matR, start = 1L)

mature_age(matU, matR, start = 1L)

mature_distrib(matU, start = 1L, repro_stages)
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix reflecting survival-related transitions; e.g. progression, stasis, and retrogression). Optionally with named rows and columns indicating the corresponding life stage names.

`matR`	The reproductive component of a matrix population model (i.e., a square projection matrix reflecting transitions due to reproduction; either sexual, clonal, or both). Optionally with named rows and columns indicating the corresponding life stage names.
`start`	The index (or stage name) of the first stage at which the author considers the beginning of life. Defaults to `1`.
`repro_stages`	A vector of stage names or indices indicating which stages are reproductive. Alternatively, a logical vector of length `ncol(matU)` indicating whether each stage is reproductive (`TRUE`) or not (`FALSE`).

Value

For `mature_distrib`, a vector giving the proportion of individuals that first reproduce within each stage class. For all others, a scalar trait value.

Note

Note that the units of time in returned values are the same as the `ProjectionInterval` of the MPM.

Author(s)

Roberto Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

Hal Caswell <hcaswell@whoi.edu>

Owen R. Jones <jones@biology.sdu.dk>

Patrick Barks <patrick.barks@gmail.com>

References

Caswell, H. 2001. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates; 2nd edition. ISBN: 978-0878930968

See Also

Other life history traits: `entropy_d()`, `entropy_k()`, `gen_time()`, `life_expect_mean()`, `longevity()`, `net_repro_rate()`, `shape_rep()`, `shape_surv()`

Examples

```
data(mpm1)

mature_prob(mpm1$matU, mpm1$matF, start = 2)
mature_age(mpm1$matU, mpm1$matF, start = 2)

### distribution of first reproductive maturity among stage classes
repstage <- repro_stages(mpm1$matF)
mature_distrib(mpm1$matU, start = 2, repro_stages = repstage)
```

`repro_stages` *Identify which stages in a matrix population model are reproductive*

Description

Takes a reproductive matrix and returns a vector of logical values (`TRUE`/`FALSE`) indicating which stages are reproductive (i.e., exhibit any positive values for reproduction). This function is a preparatory step to collapsing the matrix model into a standardized set of stage classes using the function `mpm_standardize`.

Usage

```
repro_stages(matR, na_handling = "return.true")
```

Arguments

`matR`	The reproductive component of a matrix population model (i.e., a square projection matrix reflecting transitions due to reproduction; either sexual (e.g., `matF`), clonal (e.g., `matC`), or both).
`na_handling`	One of `"return.na"`, `"return.true"`, or `"return.false"`. Determines how values of `NA` within `matR` should be handled. See Value for more details.

Value

A logical vector of length `ncol(matR)`, with values of `FALSE` corresponding to non-reproductive stages and values of `TRUE` corresponding to reproductive stages.

For a given matrix stage (i.e., column of `matR`), if there are any positive values of reproduction, the function will return `TRUE`. However, for a given stage, if there are no positive values of reproduction and one or more values of `NA`, the function will return `NA` if `na_handling == "return.na"`, `TRUE` if `na_handling == "return.true"`, or `FALSE` if `na_handling == "return.false"`.

Author(s)

Rob Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

Patrick Barks <patrick.barks@gmail.com>

See Also

Other transformation: `mpm_collapse()`, `mpm_rearrange()`, `mpm_split()`, `mpm_standardize()`, `name_stages()`, `standard_stages()`

Examples

```
matR1 <- rbind(
  c(0, 0.2, 0, 0.5),
  c(0, 0.3, 0, 0.6),
  c(0, 0, 0, 0),
  c(0, 0, 0, 0)
)

matR2 <- rbind(
  c(NA, NA, NA, 1.1),
  c(0, 0, 0.3, 0.7),
  c(0, 0, 0, 0),
  c(0, 0, 0, 0)
)

repro_stages(matR1)

# compare different methods for handling NA
repro_stages(matR2, na_handling = "return.na")
repro_stages(matR2, na_handling = "return.true")
repro_stages(matR2, na_handling = "return.false")
```

`shape_rep` *Calculate shape of reproduction over age*

Description

Calculates a 'shape' value of distribution of reproduction over age by comparing the area under a cumulative reproduction curve (over age) with the area under a cumulative function describing constant reproduction.

Usage

```
shape_rep(rep, xmin = NULL, xmax = NULL)
```

Arguments

`rep`	Either 1) a numeric vector describing reproduction over age (mx), or 2) a `data.frame` / `list` with one column / element titled 'mx' describing a reproduction over age, optionally a column / element 'x' containing age classes (each element a number representing the age at the start of the class). If x is not supplied, the function will assume age classes starting at 0 with time steps of unit. If x ends at maximum longevity, `mx[which.max(x)]` should equal 0; however it is possible to supply partial reproduction schedules.
`xmin, xmax`	The minimum and maximum age respectively over which to evaluate shape. If not given, these default to `min(x)` and `max(x)` respectively.

Value

a shape value describing symmetry of reproduction over age by comparing the area under a cumulative reproduction curve over age with the area under constant reproduction. May take any real value between -0.5 and +0.5. A value of 0 indicates negligible ageing (neither generally increasing nor generally decreasing reproduction with age); positive values indicate senescence (generally decreasing reproduction with age); negative values indicate negative senescence (generally increasing reproduction with age). A value of +0.5 indicates that (hypothetically) all individuals are born to individuals of age 0; a value of -0.5 indicates that all individuals are born at the age of maximum longevity.

Author(s)

Iain Stott <iainmstott@gmail.com>

References

Baudisch, A, Stott, I. 2019. A pace and shape perspective on fertility. Methods Ecol Evol. 10: 1941– 1951. <https://doi.org/10.1111/2041-210X.13289>

See Also

Other life history traits: `entropy_d()`, `entropy_k()`, `gen_time()`, `life_expect_mean()`, `longevity()`, `net_repro_rate()`, `repro_maturity`, `shape_surv()`

Examples

```
# increasing mx yields negative shape
mx <- c(0, 0, 0.3, 0.4, 0.5, 0.6)
shape_rep(mx)

# decreasing mx yields positive shape
mx <- c(1.1, 1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4)
shape_rep(mx)

# constant mx yields shape = 0
mx <- c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
shape_rep(mx)
```

`shape_surv` *Calculate shape of survival over age*

Description

Calculates a 'shape' value of survival lifespan inequality by comparing the area under a survival curve (over age) with the area under a constant survival function.

Usage

```
shape_surv(surv, xmin = NULL, xmax = NULL, trunc = FALSE)
```

Arguments

`surv`	Either 1) a numeric vector describing a survival curve (lx), or 2) a `data.frame` / `list` with one column / element titled 'lx' describing a survival curve, optionally a column / element 'x' containing age classes (each element a number representing the age at the start of the class). If `x` is not supplied, the function will assume age classes starting at `0` with time steps of `1` unit of the `ProjectionInterval`. If `x` begins at `0` then `lx[1]` should equal `1`. If `x` ends at maximum longevity, then `lx[which.max(x)]` should equal `0`; however it is possible to supply partial survivorship curves.
`xmin, xmax`	The minimum and maximum age respectively over which to evaluate shape. If not given, these default to `min(x)` and `max(x)` respectively.
`trunc`	logical determining whether to truncate life tables or not when any `lx == 0`. Usually this is the case only for the final value of `lx`. As the function calculates `log(lx)`, these value(s) cannot be handled. `trunc == TRUE` strips out the zero value(s). An alternative to this is to transform the zeroes to something approximating zero (e.g., 1e-7).

Value

a shape value describing lifespan inequality by comparing the area under a survival (`lx`) curve over age with the area under a constant (Type II) survival function. The shape value may take any real value between -0.5 and +0.5. A value of 0 indicates negligible ageing (neither generally increasing nor generally decreasing survival with age); negative values indicate negative senescence (generally increasing survival with age); positive values indicate senescence (generally decreasing survival with age). A value of +0.5 indicates that all individuals die at age of maximum longevity; a value of -0.5 indicates that (hypothetically) all individuals die at birth.

Author(s)

Iain Stott <iainmstott@gmail.com>

See Also

Other life history traits: `entropy_d()`, `entropy_k()`, `gen_time()`, `life_expect_mean()`, `longevity()`, `net_repro_rate()`, `repro_maturity`, `shape_rep()`

Examples

```
# exponential decline in lx yields shape = 0
lx <- 0.7^(0:20)
shape_surv(lx)
```

`standard_stages` *Identify stages corresponding to different parts of the reproductive life cycle*

Description

Identify the stages of a matrix population model that correspond to different parts of the reproductive life cycle, namely propagule, pre-reproductive, reproductive and post-reproductive. These classifications are used to standardise matrices to allow comparisons across species with different life cycle structures, see `mpm_standardize`.

Usage

```
standard_stages(matF, repro_stages, matrix_stages)
```

Arguments

`matF`	The sexual component of a matrix population model (i.e., a square projection matrix reflecting transitions only due to *sexual* reproduction). It assumes that it has been rearranged so that non-reproductive stages are in the final rows/columns.
`repro_stages`	Logical vector identifying which stages are reproductive.
`matrix_stages`	(character) vector of stages, values are `prop` (propagule), `active`, and `dorm` (dormant).

Details

Assumes that fecundity and mean fecundity matrices have been rearranged so that non-reproductive stages are in the final rows/columns. Output indicates groupings to be used when collapsing the matrix model.

Value

A list with four elements:

`propStages`	Position of the propagule stages
`preRepStages`	Position of the pre-reproductive stages
`repStages`	Position of the reproductive stages
`postRepStages`	Position of the post-reproductive stages

Note

Dormant stages are not currently handled.

Author(s)

Rob Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

See Also

mpm_standardize

Other transformation: mpm_collapse(), mpm_rearrange(), mpm_split(), mpm_standardize(), name_stages(), repro_stages()

Examples

```
matU <- rbind(
  c(0.1, 0, 0, 0, 0),
  c(0.5, 0.2, 0.1, 0, 0),
  c(0, 0.3, 0.3, 0.1, 0),
  c(0, 0, 0.4, 0.4, 0.1),
  c(0, 0, 0, 0.1, 0.4)
)

matF <- rbind(
  c(0, 1.1, 0, 1.6, 0),
  c(0, 0.8, 0, 0.4, 0),
  c(0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 0)
)

repro_stages <- c(FALSE, TRUE, FALSE, TRUE, FALSE)
matrix_stages <- c("prop", "active", "active", "active", "active")

r <- mpm_rearrange(matU, matF,
  repro_stages = repro_stages,
  matrix_stages = matrix_stages
)

standard_stages(r$matF, r$repro_stages, r$matrix_stages)
```

---

vital_rates *Derive mean vital rates from a matrix population model*

Description

Derive mean vital rates corresponding to separate demographic processes from a matrix population model. Specifically, this function decomposes vital rates of survival, progression, retrogression, sexual reproduction and clonal reproduction, with various options for weighting and grouping stages of the life cycle.

Usage

```
vital_rates(
  matU,
```

```
  matF,
  matC = NULL,
  weights = NULL,
  splitStages = "all",
  matrixStages = NULL
)
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix reflecting survival-related transitions; e.g. progression, stasis, and retrogression).
`matF`	The sexual component of a matrix population model (i.e., a square projection matrix reflecting transitions due to sexual reproduction)
`matC`	The clonal component of a matrix population model (i.e., a square projection matrix reflecting transitions due to clonal reproduction). Defaults to `NULL`, indicating no clonal reproduction (i.e. `matC` is a matrix of zeros).
`weights`	Vector of stage-specific weights to apply while averaging vital rates. Default is `NULL` reflecting equal weighting for all stages. May also be `"SSD"` to weight vital rates by the stable distribution of `matA`.
`splitStages`	What groups should vital rates be averaged over. Either: `"all"`: all stages grouped. `"ontogeny"`: group juvenile stages (all stages prior to the first stage with sexual reproduction) and adult stages. `"matrixStages"`: group according to a standardized set of stage classes (propagule, active, and dormant). If `splitStages = "matrixStages"`, must also specify separate argument `matrixStages`.
`matrixStages`	Vector of stage-specific standardized matrix classes ("prop" for propagule, "active", and/or "dorm" for dormant). Only used if `splitStages = "matrixClass"`.

Value

A list of averaged vital rates.

Author(s)

Roberto Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

References

Caswell, H. 2001. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates; 2nd edition. ISBN: 978-0878930968

See Also

Other vital rates: `vr_mat`, `vr_vec`, `vr`

Examples

```
matU <- rbind(
  c(0.1, 0, 0, 0),
  c(0.5, 0.2, 0.1, 0),
  c(0, 0.3, 0.3, 0.1),
  c(0, 0, 0.5, 0.6)
)

matF <- rbind(
  c(0, 0, 1.1, 1.6),
  c(0, 0, 0.8, 0.4),
  c(0, 0, 0, 0),
  c(0, 0, 0, 0)
)

matC <- rbind(
  c(0, 0, 0.4, 0.5),
  c(0, 0, 0.3, 0.1),
  c(0, 0, 0, 0),
  c(0, 0, 0, 0)
)

# Vital rate outputs without weights
vital_rates(matU, matF, matC, splitStages = "all")
vital_rates(matU, matF, matC, splitStages = "ontogeny")

# Group vital rates according to specified matrixStages
ms <- c("prop", "active", "active", "active")
vital_rates(matU, matF, matC, splitStages = "matrixStages", matrixStages = ms)

# Vital rate outputs weighted by the stable stage distribution of 'matA'
vital_rates(matU, matF, matC, splitStages = "all", weights = "SSD")
```

`vr` *Derive mean vital rates from a matrix population model*

Description

Derive mean vital rates of survival, growth (or development), shrinkage (or de-development), stasis, dormancy, or reproduction from a matrix population model, by averaging across stage classes. These functions include optional arguments for custom weighting of different stage classes (see *Weighting stages*), excluding certain stage classes from the calculation (see *Excluding stages*), and defining the set of biologically-possible transitions (see *Possible transitions*).

These decompositions assume that all transition rates are products of a stage-specific survival term (column sums of `matU`) and a lower level vital rate that is conditional on survival (growth/development, shrinkage/de-development, stasis, dormancy, or a/sexual reproduction). Reproductive vital rates that are not conditional on survival (i.e., within a stage class from which there is no survival) are also allowed.

Usage

```
vr_survival(matU, posU = matU > 0, exclude_col = NULL, weights_col = NULL)

vr_growth(
  matU,
  posU = matU > 0,
  exclude = NULL,
  exclude_row = NULL,
  exclude_col = NULL,
  weights_col = NULL,
  surv_only_na = TRUE
)

vr_shrinkage(
  matU,
  posU = matU > 0,
  exclude = NULL,
  exclude_row = NULL,
  exclude_col = NULL,
  weights_col = NULL,
  surv_only_na = TRUE
)

vr_stasis(
  matU,
  posU = matU > 0,
  exclude = NULL,
  weights_col = NULL,
  surv_only_na = TRUE
)

vr_dorm_enter(matU, posU = matU > 0, dorm_stages, weights_col = NULL)

vr_dorm_exit(matU, posU = matU > 0, dorm_stages, weights_col = NULL)

vr_fecundity(
  matU,
  matR,
  posR = matR > 0,
  exclude_col = NULL,
  weights_row = NULL,
  weights_col = NULL
)
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix reflecting survival-related transitions; e.g., progression, stasis, and retro-

	gression)
`posU`	A logical matrix of the same dimension as `matU`, with elements indicating whether a given `matU` transition is possible (`TRUE`) or not (`FALSE`). Defaults to `matU > 0` (see *Possible transitions*).
`exclude_col`	Integer, character or logical vector indicating stages for which transitions both *to* and *from* the stage should be excluded from the calculation of vital rates. See section *Excluding stages*.
`weights_col`	Vector of stage-specific weights to apply while averaging vital rates across columns. See section *Weighting stages*.
`exclude`	Integer, character or logical vector indicating stages for which transitions both *to* and *from* the stage should be excluded from the calculation of vital rates. See section *Excluding stages*.
`exclude_row`	Integer, character or logical vector indicating stages for which transitions both *to* and *from* the stage should be excluded from the calculation of vital rates. See section *Excluding stages*.
`surv_only_na`	If there is only one possible `matU` transition in a given column, should that transition be attributed exclusively to survival? If `TRUE`, the vital rate of growth/stasis/shrinkage in that column will be coerced to `NA`. If `FALSE`, dividing the single transition by the stage-specific survival probability will always yield a value of `1`. Defaults to `TRUE`.
`dorm_stages`	Integer or character vector indicating dormant stage classes.
`matR`	The reproductive component of a matrix population model (i.e., a square projection matrix reflecting transitions due to reproduction; either sexual, clonal, or both)
`posR`	A logical matrix of the same dimension as `matR`, with elements indicating whether a given `matR` transition is possible (`TRUE`) or not (`FALSE`). Defaults to `matR > 0` (see *Possible transitions*).
`weights_row`	Vector of stage-specific weights to apply while summing vital rates across rows within columns. See section *Weighting stages*.

Value

Vector of vital rates. Vital rates corresponding to impossible transitions are coerced to `NA` (see *Possible transitions*).

Possible transitions

A transition rate of `0` within a matrix population model may indicate that the transition is not possible in the given life cycle (e.g., tadpoles never revert to eggs), or that the transition rate is possible but was estimated to be `0` in the relevant population and time period. If vital rates are to be averaged across multiple stage classes, or compared across populations, it may be important to distinguish between these two types of zeros.

By default, the `vr_` functions assume that a transition rate of `0` indicates an impossible transition, in which case a value of `NA` will be used in relevant calculations. Specifically, the arguments `posU` and `posR` are specified by the logical expressions `(matU > 0)` and `(matR > 0)`, respectively. If the matrix population model includes transitions that are estimated to be `0` but still in fact possible, one should specify the `posU` and/or `posR` arguments manually.

Weighting stages

In averaging vital rates across stages, it may be desirable to weight stage classes differently (e.g., based on reproductive values or stable distributions). Weights are generally applied when averaging across columns, i.e., across transitions *from* a set of stage classes (e.g., averaging stage-specific survival probabilities across multiple stages). All `vr_` functions therefore include an optional argument `weights_from`.

In principle, particularly for vital rates of reproduction, the user can also apply weights when summing across rows within columns, i.e., across reproductive transitions *to* a set of stage classes (e.g., summing the production of different types of offspring, such as seeds vs. seedlings). The function `vr_fecundity` therefore also includes an optional argument `weights_to`.

If supplied, `weights_from` will automatically be scaled to sum to 1 over the set of possible transitions, whereas `weights_to` will not be rescaled because we wish to enable the use of reproductive values here, which do not naturally sum to 1.

Excluding stages

It may be desirable to exclude one or more stages from the calculation of certain vital rates. For instance, we might not believe that 'growth' to a dormant stage class really reflects biological growth, in which case we could exclude transitions *to* the dormant stage class using the argument `exclude_row`. We may or may not want to ignore 'growth' transitions *from* the dormant stage class, which can be done using `exclude_col`. To exclude transitions both *to and from* a given set of stages, use argument `exclude`.

Author(s)

Patrick Barks <patrick.barks@gmail.com>

See Also

Other vital rates: `vital_rates`(), `vr_mat`, `vr_vec`

Examples

```
# create example MPM (stage 4 is dormant)
matU <- rbind(
  c(0.1, 0, 0, 0),
  c(0.5, 0.2, 0.1, 0.1),
  c(0, 0.3, 0.3, 0.1),
  c(0, 0, 0.5, 0.4)
)

matF <- rbind(
  c(0, 0.7, 1.1, 0),
  c(0, 0.3, 0.8, 0),
  c(0, 0, 0, 0),
  c(0, 0, 0, 0)
)

vr_survival(matU, exclude_col = 4)
vr_growth(matU, exclude = 4)
```

```
vr_shrinkage(matU, exclude = 4)
vr_stasis(matU, exclude = 4)

# `exclude*` and `*_stages` arguments can accept stage names
matU <- name_stages(matU)
matF <- name_stages(matF)
vr_dorm_enter(matU, dorm_stages = "stage_4")
vr_dorm_exit(matU, dorm_stages = 4)

vr_fecundity(matU, matF, exclude_col = 4)
```

`vr_mat` *Derive survival-independent vital rates for growth, stasis, shrinkage, and reproduction*

Description

Divides columns of a matrix population model by the corresponding stage-specific survival probability, to obtain lower-level vital rates for growth, stasis, shrinkage, and reproduction. Vital rates corresponding to biologically impossible transitions are coerced to `NA`.

These decompositions assume that all transition rates are products of a stage-specific survival term (column sums of `matU`) and a lower level vital rate that is conditional on survival (growth, shrinkage, stasis, or reproduction). Reproductive vital rates that are not conditional on survival (i.e., within a stage class from which there is no survival) are also allowed.

Usage

```
vr_mat_U(matU, posU = matU > 0, surv_only_na = TRUE)

vr_mat_R(matU, matR, posR = matR > 0)
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix reflecting survival-related transitions; e.g. progression, stasis, and retrogression)
`posU`	A logical matrix of the same dimension as `matU`, with elements indicating whether a given `matU` transition is possible (`TRUE`) or not (`FALSE`). Defaults to `matU > 0` (see Details).
`surv_only_na`	If there is only one possible `matU` transition in a given column, should that transition be attributed exclusively to survival? If `TRUE`, the vital rate of growth/stasis/shrinkage in that column will be coerced to `NA`. If `FALSE`, dividing the single transition by the stage-specific survival probability will always yield a value of `1`. Defaults to `TRUE`.
`matR`	The reproductive component of a matrix population model (i.e., a square projection matrix reflecting transitions due to reproduction; either sexual, clonal, or both)

`posR`	A logical matrix of the same dimension as `matR`, with elements indicating whether a given `matR` transition is possible (`TRUE`) or not (`FALSE`). Defaults to `matR > 0` (see Details).

Details

A transition rate of `0` within a matrix population model may indicate that the transition is not possible in the given life cycle (e.g., tadpoles never revert to eggs), or that the transition is possible but was estimated to be `0` in the relevant population and time period. If vital rates are to be averaged across multiple stage classes, or compared across populations, it may be important to distinguish between these two types of zeros.

By default, `vr_mat` assumes that a transition rate of `0` indicates an impossible transition, in which case a value of `NA` will be returned in the relevant matrix cell. Specifically, the arguments `posU` and `posR` are specified by the logical expressions `(matU > 0)` and `(matR > 0)`, respectively. If the matrix population model includes transitions that are possible but estimated to be `0`, one should specify the `posU` and/or `posR` arguments manually.

Value

A matrix of vital rates. Vital rates corresponding to impossible transitions will be coerced to `NA` (see Details).

Author(s)

Patrick Barks <patrick.barks@gmail.com>

References

Caswell, H. 2001. Matrix Population Models: Construction, Analysis, and Interpretation. Sinauer Associates; 2nd edition. ISBN: 978-0878930968

See Also

Other vital rates: `vital_rates()`, `vr_vec`, `vr`

Examples

```
matU <- rbind(
  c(0.1, 0, 0, 0),
  c(0.5, 0.2, 0.1, 0),
  c(0, 0.3, 0.3, 0.1),
  c(0, 0, 0.5, 0.6)
)

matR <- rbind(
  c(0, 0, 1.1, 1.6),
  c(0, 0, 0.8, 0.4),
  c(0, 0, 0, 0),
  c(0, 0, 0, 0)
)
```

```
# extract vital rates of survival from matU
vr_mat_U(matU)

# extract vital rates of reproduction from matR
vr_mat_R(matU, matR)
```

---

vr_vec *Derive stage-specific vital rates from a matrix population model*

Description

Derive a vector of stage-specific vital rates of survival, growth, shrinkage, stasis, dormancy, or reproduction from a matrix population model. These functions include optional arguments for excluding certain stage classes from the calculation (see *Excluding stages*), and defining the set of biologically-possible transitions (see *Possible transitions*).

This decomposition assume that all transition rates are products of a stage-specific survival term (column sums of `matU`) and a lower level vital rate that is conditional on survival (growth, shrinkage, stasis, dormancy, or reproduction). Reproductive vital rates that are not conditional on survival (i.e., within a stage class from which there is no survival) are also allowed.

Usage

```
vr_vec_survival(matU, posU = matU > 0, exclude_col = NULL)

vr_vec_growth(
  matU,
  posU = matU > 0,
  exclude = NULL,
  exclude_row = NULL,
  exclude_col = NULL,
  surv_only_na = TRUE
)

vr_vec_shrinkage(
  matU,
  posU = matU > 0,
  exclude = NULL,
  exclude_row = NULL,
  exclude_col = NULL,
  surv_only_na = TRUE
)

vr_vec_stasis(matU, posU = matU > 0, exclude = NULL, surv_only_na = TRUE)

vr_vec_dorm_enter(matU, posU = matU > 0, dorm_stages)
```

```
vr_vec_dorm_exit(matU, posU = matU > 0, dorm_stages)

vr_vec_reproduction(
  matU,
  matR,
  posR = matR > 0,
  exclude_col = NULL,
  weights_row = NULL
)
```

Arguments

`matU`	The survival component of a matrix population model (i.e., a square projection matrix only containing survival-related transitions; progression, stasis, and retrogression).
`posU`	A logical matrix of the same dimension as `matU`, with elements indicating whether a given `matU` transition is possible (`TRUE`) or not (`FALSE`). Defaults to `matU > 0` (see Details).
`exclude_col`	Integer, character or logical vector indicating stages for which transitions both *to* and *from* the stage should be excluded from the calculation of vital rates. See section *Excluding stages*.
`exclude`	Integer, character or logical vector indicating stages for which transitions both *to* and *from* the stage should be excluded from the calculation of vital rates. See section *Excluding stages*.
`exclude_row`	Integer, character or logical vector indicating stages for which transitions both *to* and *from* the stage should be excluded from the calculation of vital rates. See section *Excluding stages*.
`surv_only_na`	If there is only one possible `matU` transition in a given column, should that transition be attributed exclusively to survival? If `TRUE`, the vital rate of growth/stasis/shrinkage in that column will be coerced to `NA`. If `FALSE`, dividing the single transition by the stage-specific survival probability will always yield a value of `1`. Defaults to `TRUE`.
`dorm_stages`	Integer or character vector indicating dormant stage classes.
`matR`	The reproductive component of a matrix population model (i.e., a square projection matrix only reflecting transitions due to reproduction; either sexual, clonal, or both).
`posR`	A logical matrix of the same dimension as `matR`, with elements indicating whether a given `matR` transition is possible (`TRUE`) or not (`FALSE`). Defaults to `matR > 0` (see Details).
`weights_row`	Vector of stage-specific weights to apply while summing vital rates across rows within columns (e.g., reproductive value vector).

Value

Vector of vital rates. Vital rates corresponding to impossible transitions are coerced to `NA` (see *Possible transitions*).

Possible transitions

A transition rate of `0` within a matrix population model may indicate that the transition is not possible in the given life cycle (e.g., tadpoles never revert to eggs), or that the transition rate is possible but was estimated to be `0` in the relevant population and time period. If vital rates are to be averaged across multiple stage classes, or compared across populations, it may be important to distinguish between these two types of zeros.

By default, the `vitals_` functions assume that a transition rate of `0` indicates an impossible transition, in which case a value of `NA` will be used in relevant calculations. Specifically, the arguments `posU` and `posR` are specified by the logical expressions `(matU > 0)` and `(matR > 0)`, respectively. If the matrix population model includes transitions that are estimated to be `0` but still in fact possible, one should specify the `posU` and/or `posR` arguments manually.

Excluding stages

It may be desirable to exclude one or more stages from the calculation of certain vital rates. For instance, a user might not believe that 'growth' to a dormant stage class really reflects biological growth, in which case the user could exclude transitions *to* the dormant stage class using the argument `exclude_row`. The user may or may not want to ignore 'growth' transitions *from* the dormant stage class, which can be done using `exclude_col`. The argument `exclude_col` effectively just coerces the respective vital rate to `NA`, to prevent it from getting used in subsequent calculations. To exclude transitions both *to and from* a given set of stages, use argument `exclude`.

Author(s)

Patrick Barks <patrick.barks@gmail.com>

See Also

Other vital rates: `vital_rates()`, `vr_mat`, `vr`

Examples

```
# create example MPM (stage 4 is dormant)
matU <- rbind(
  c(0.1, 0, 0, 0),
  c(0.5, 0.2, 0.1, 0.1),
  c(0, 0.3, 0.3, 0.1),
  c(0, 0, 0.5, 0.4)
)

matR <- rbind(
  c(0, 0.7, 1.1, 0),
  c(0, 0.3, 0.8, 0),
  c(0, 0, 0, 0),
  c(0, 0, 0, 0)
)

vr_vec_survival(matU, exclude_col = 4)
vr_vec_growth(matU, exclude = 4)
```

```
# `exclude*` and `*_stages` arguments can accept stage names
matU <- name_stages(matU)
matR <- name_stages(matR)
vr_vec_shrinkage(matU, exclude = 4)
vr_vec_stasis(matU, exclude = "stage_4")

vr_vec_dorm_enter(matU, dorm_stages = 4)
vr_vec_dorm_exit(name_stages(matU), dorm_stages = "stage_4")

vr_vec_reproduction(matU, matR, exclude_col = "stage_4")
```

Index