Package 'adapt3'

July 17, 2025

Type Package

Title Adaptive Dynamics and Community Matrix Model Projections

Version 1.0.1

Date 2025-07-11

Description Runs projections of groups of matrix projection models (MPMs), allowing density dependence mechanisms to work across MPMs. This package was developed to run both adaptive dynamics simulations such as pairwise and multiple invasibility analyses, and community projections in which species are represented by MPMs. All forms of MPMs are allowed, including integral projection models (IPMs). Also includes individual-based modeling (IBM) versions of these.

Encoding UTF-8

License GPL (>= 2)

URL https://github.com/dormancy1/adapt3

Imports Rcpp (>= 1.0.12), lefko3, methods, rlang, grDevices

LinkingTo Rcpp, RcppArmadillo, BH, lefko3

LazyData true

BugReports https://github.com/dormancy1/adapt3/issues

RoxygenNote 7.3.2

Suggests knitr, rmarkdown

VignetteBuilder knitr, rmarkdown

NeedsCompilation yes

Author Richard P. Shefferson [aut, cre] (ORCID: <https://orcid.org/0000-0002-5234-3131>)

Maintainer Richard P. Shefferson <cdorm@g.ecc.u-tokyo.ac.jp>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2025-07-16 15:40:08 UTC

Contents

`adapt3-package` *Adaptive Dynamics and Community Matrix Model Projections*

Description

Runs projections of groups of matrix projection models (MPMs), allowing density dependence mechanisms to work across MPMs. This package was developed to run both adaptive dynamics simulations such as pairwise and multiple invasion analysis, and community projections in which species are represented by MPMs. All forms of MPMs are allowed, including integral projection models (IPMs).

Details

The adapt3 package provides three categories of functions:

1. Core projection

2. Function characterizing relationships among MPMs

3. Functions describing, summarizing, or visualizing results

adapt3 also includes example datasets complete with sample code.

Author(s)

Maintainer: Richard P. Shefferson <cdorm@g.ecc.u-tokyo.ac.jp> (ORCID)

Richard P. Shefferson <cdorm@g.ecc.u-tokyo.ac.jp>

References

Pending

See Also

Useful links:

- https://github.com/dormancy1/adapt3
- Report bugs at https://github.com/dormancy1/adapt3/issues

`cypa_data` *Demographic Dataset of* Cypripedium parviflorum *Population, in Horizontal Format*

Description

A dataset containing the states and fates of *Cypripedium parviflorum* (small yellow lady's slipper orchids), family Orchidaceae, from a population in Illinois, USA, resulting from monitoring that occurred annually between 1994 and 2011.

Usage

```
data(cypa_data)
```

Format

A data frame with 1119 individuals and 37 variables. Each row corresponds to an unique individual, and each variable from `Inf.94` on refers to the state of the individual in a particular year.

plant_id A numeric variable giving a unique number to each individual.

Inf.94 Number of inflorescences in 1994.

Veg.94 Number of stems without inflorescences in 1994.

Inf.95 Number of inflorescences in 1995.

Veg.95 Number of stems without inflorescences in 1995.

Inf.96 Number of inflorescences in 1996.

Veg.96 Number of stems without inflorescences in 1996.

Inf.97 Number of inflorescences in 1997.

Veg.97 Number of stems without inflorescences in 1997.

Inf.98 Number of inflorescences in 1998.

Veg.98 Number of stems without inflorescences in 1998.

Inf.99 Number of inflorescences in 1999.

Veg.99 Number of stems without inflorescences in 1999.

Inf.00 Number of inflorescences in 2000.

Veg.00 Number of stems without inflorescences in 2000.

Inf.01 Number of inflorescences in 2001.

Veg.01 Number of stems without inflorescences in 2001.

Inf.02 Number of inflorescences in 2002.

Veg.02 Number of stems without inflorescences in 2002.

Inf.03 Number of inflorescences in 2003.

Veg.03 Number of stems without inflorescences in 2003.

Inf.04 Number of inflorescences in 2004.

Veg.04 Number of stems without inflorescences in 2004.

Inf.05 Number of inflorescences in 2005.

Veg.05 Number of stems without inflorescences in 2005.

Inf.06 Number of inflorescences in 2006.

Veg.06 Number of stems without inflorescences in 2006.

Inf.07 Number of inflorescences in 2007.

Veg.07 Number of stems without inflorescences in 2007.

Inf.08 Number of inflorescences in 2008.

Veg.08 Number of stems without inflorescences in 2008.

Inf.09 Number of inflorescences in 2009.

Veg.09 Number of stems without inflorescences in 2009.

Inf.10 Number of inflorescences in 2010.

Veg.10 Number of stems without inflorescences in 2010.

Inf.11 Number of inflorescences in 2011.

Veg.11 Number of stems without inflorescences in 2011.

Source

Shefferson, R.P., R. Mizuta, and M.J. Hutchings. 2017. Predicting evolution in response to climate change: the example of sprouting probability in three dormancy-prone orchid species. *Royal Society Open Science* 4(1):160647.

Examples

```
library(lefko3)

data(cypa_data)

sizevector <- c(0, 0, 0, 0, 0, 0, 1, 2.5, 4.5, 8, 17.5)
stagevector <- c("SD", "P1", "P2", "P3", "SL", "D", "XSm", "Sm", "Md", "Lg",
  "XLg")
repvector <- c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
obsvector <- c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
immvector <- c(0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 0, 0, 0, 0, 0.5, 0.5, 1, 1, 2.5, 7)
```

```
cypframe_raw <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  propstatus = propvector, immstatus = immvector, indataset = indataset,
  binhalfwidth = binvec)

cypraw_v1 <- verticalize3(data = cypa_data, noyears = 18, firstyear = 1994,
  individcol = "plant_id", blocksize = 2, sizeacol = "Inf.94",
  sizebcol = "Veg.94", repstracol = "Inf.94", fecacol = "Inf.94",
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cypsupp2r <- supplemental(stage3 = c("SD", "P1", "P2", "P3", "SL", "D",
    "XSm", "Sm", "SD", "P1"),
  stage2 = c("SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "rep",
    "rep"),
  eststage3 = c(NA, NA, NA, NA, NA, "D", "XSm", "Sm", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.10, 0.20, 0.20, 0.20, 0.25, NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, 0.5, 0.5),
  type =c(1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  stageframe = cypframe_raw, historical = FALSE)

cypmatrix2r <- rlefko2(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", indivcol = "individ")

lambda3(cypmatrix2r)
```

---

equiv_input *Create an Equivalence Vector for Each Population*

Description

Function `equiv_input()` creates a data frame summarizing the degree to which an individual in each stage of a life history is equivalent to a standard individual.

Usage

```
equiv_input(mpm, stage2 = NA, stage1 = NA, age2 = NA, value = 1)
```

Arguments

`mpm`	The lefkoMat object to be used in projection. Can be an example MPM if function-based projection is planned.
`stage2`	A vector showing the name or number of a stage in occasion *t* that should be set to a positive number of individuals in the start vector. Abbreviations for groups of stages are also usable (see Notes). This input is required for all stage-based and age-by-stage MPMs. Defaults to `NA`.

stage1	A vector showing the name or number of a stage in occasion *t*-1 that should be set to a positive number of individuals in the start vector. Abbreviations for groups of stages are also usable (see Notes). This is only used for historical MPMs, since the rows of hMPMs correspond to stage-pairs in times *t* and *t*-1 together. Only required for historical MPMs, and will result in errors if otherwise used.
age2	A vector showing the age of each respective stage in occasion *t* that should be set to a positive number of individuals in the start vector. Only used for Leslie and age-by-stage MPMs. Defaults to NA.
value	A vector showing the values, in order, of the number of individuals set for the stage or stage-pair in question. Defaults to 1.

Value

A list of class adaptEq, with four objects, which can be used as input in function project3(). The last three include the ahstages, hstages, and agestages objects from the lefkoMat object supplied in mpm. The first element in the list is a data frame with the following variables:

stage2	Stage at occasion *t*.
stage_id_2	The stage number associated with stage2.
stage1	Stage at occasion *t*-1, if historical. Otherwise NA.
stage_id_1	The stage number associated with stage1.
age2	The age of individuals in stage2 and, if applicable, stage1. Only used in age-by-stage MPMs.
row_num	A number indicating the respective starting vector element.
value	Number of individuals in corresponding stage or stage-pair.

Notes

Entries in stage2, and stage1 can include abbreviations for groups of stages. Use rep if all reproductive stages are to be used, nrep if all mature but non-reproductive stages are to be used, mat if all mature stages are to be used, immat if all immature stages are to be used, prop if all propagule stages are to be used, npr if all non-propagule stages are to be used, obs if all observable stages are to be used, nobs if all unobservable stages are to be used, and leave empty or use all if all stages in stageframe are to be used.

Examples

```
library(lefko3)
data(cypdata)

data(cypa_data)

sizevector <- c(0, 0, 0, 0, 0, 0, 1, 2.5, 4.5, 8, 17.5)
stagevector <- c("SD", "P1", "P2", "P3", "SL", "D", "XSm", "Sm", "Md", "Lg",
  "XLg")
repvector <- c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
obsvector <- c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
```

```
matvector <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
immvector <- c(0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 0, 0, 0, 0, 0.5, 0.5, 1, 1, 2.5, 7)

cypframe_raw <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  propstatus = propvector, immstatus = immvector, indataset = indataset,
  binhalfwidth = binvec)

cycaraw_v1 <- verticalize3(data = cypdata, noyears = 6, firstyear = 2004,
  patchidcol = "patch", individcol = "plantid", blocksize = 4,
  sizeacol = "Inf2.04", sizebcol = "Inf.04", sizeccol = "Veg.04",
  repstracol = "Inf.04", repstrbcol = "Inf2.04", fecacol = "Pod.04",
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cyparaw_v1 <- verticalize3(data = cypa_data, noyears = 18, firstyear = 1994,
  individcol = "plant_id", blocksize = 2, sizeacol = "Inf.94",
  sizebcol = "Veg.94", repstracol = "Inf.94", fecacol = "Inf.94",
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cypsupp2r <- supplemental(stage3 = c("SD", "P1", "P2", "P3", "SL", "D",
    "XSm", "Sm", "SD", "P1"),
  stage2 = c("SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "rep",
    "rep"),
  eststage3 = c(NA, NA, NA, NA, NA, "D", "XSm", "Sm", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.10, 0.20, 0.20, 0.20, 0.25, NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, 0.5, 0.5),
  type =c(1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  stageframe = cypframe_raw, historical = FALSE)
cyp_supp_list1 <- list(cypsupp2r, cypsupp2r)

cycamatrix2r <- rlefko2(data = cycaraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

cypamatrix2r <- rlefko2(data = cyparaw_v1, stageframe = cypframe_raw,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

cyp_mpm_list <- list(cycamatrix2r, cypamatrix2r)

cyca2_start <- start_input(cycamatrix2r, stage2 = c("SD", "P1", "P2"),
  value = c(500, 100, 200))
cypa2_start <- start_input(cypamatrix2r, stage2 = c("SD", "P1", "P2"),
  value = c(5000, 1000, 2000))
cyp_start_list <- list(cyca2_start, cypa2_start)
```

```
cyp2_dv <- density_input(cypamatrix2r, stage3 = c("SD", "P1"),
  stage2 = c("rep", "rep"), style = c(1, 1), alpha = c(0.5, 1.2),
  beta = c(1.0, 2.0), type = c(2, 1))
cyp_dv_list <- list(cyp2_dv, cyp2_dv)

cyp_eq <- equiv_input(cycamatrix2r,
  stage2 = c("SD", "P1", "SL", "D", "XSm", "Sm", "Md", "Lg", "XLg"),
  value = c(0, 1, 1, 1, 1, 1, 1, 1, 1))

eq_list <- list(cyp_eq, cyp_eq)

cyp_comm_proj <- project3(mpms = cyp_mpm_list, starts = cyp_start_list,
  density = cyp_dv_list, times = 10)

summary(cyp_comm_proj)
```

---

invade3 *Run Pairwise and Multiple Invasion Analysis*

Description

Function invade3 runs pairwise and multiple invasion analyses.

Usage

```
invade3(
  axis = NULL,
  mpm = NULL,
  vrm = NULL,
  stageframe = NULL,
  supplement = NULL,
  equivalence = NULL,
  starts = NULL,
  years = NULL,
  patches = NULL,
  tweights = NULL,
  format = NULL,
  entry_time = NULL,
  sp_density = NULL,
  ind_terms = NULL,
  dev_terms = NULL,
  fb_sparse = NULL,
  firstage = NULL,
  finalage = NULL,
  fecage_min = NULL,
  fecage_max = NULL,
```

```
  cont = NULL,
  prebreeding = NULL,
  fecmod = NULL,
  density = NULL,
  density_vr = NULL,
  stochastic = NULL,
  A_only = NULL,
  integeronly = NULL,
  fitness_table = NULL,
  trait_optima = NULL,
  zap_min = NULL,
  converged_only = NULL,
  err_check = NULL,
  var_per_run = 2L,
  substoch = 0L,
  elast_mult = 0.995,
  nreps = 1L,
  times = 10000L,
  fitness_times = 100L,
  exp_tol = 700,
  theta_tol = 1e+08,
  threshold = 1e-08,
  loop_max = 150L
)
```

Arguments

axis	The adaptAxis object detailing all variant characteristics. Essentially, a data frame giving the values of all changes to vital rates and transition elements to test, where each value is change is given by row.
mpm	An MPM of class lefkoMat, for use if using existing MPMs.
vrm	A vrm_input object corresponding to a distinct MPM that will be created during analysis. Requires a stageframe, entered in argument stageframe.
stageframe	A stageframe defining stages and the life cycle for the entered object in argument vrms. Must be of class stageframe.
supplement	An optional data frame of class lefkoSD providing supplemental data that should be incorporated into function-based MPMs. See supplemental() for details. Use only with argument vrm.
equivalence	An optional object of class adaptEq giving the degree to which individuals in each stage are equivalent to one another. May also be a numeric vector, in which case the vector must have the same number of elements as the number of rows in the associated MPM, with each element giving the effect of an individual of that age, stage, age-stage, or stage-pair, depending on whether the MPM is age-based, ahistorical stage-based, age-by-stage, or historical stage-based, respectively. Numeric entries used in these vectors can be thought of as Lotka-Volterra interaction terms, such as are used in multiple species competition models.

starts — An optional lefkoSV object, which is a data frame providing the starting numbers of individuals of each stage. If not provided, then all projections start with a single individual per stage.

years — An optional term corresponding to a single integer vector of time t values. If a vector shorter than times is supplied, then this vector will be cycled. Defaults to a vector of all detected years in argument mpm or argument vrm.

patches — An optional single string giving a single pop-patch to be used during invasion analysis. Defaults to the population-level set or the first patch, depending on whether the former exists.

tweights — An optional numeric vector or matrice denoting the probabilities of choosing each matrix in each MPM in a stochastic projection. If a matrix, then a first-order Markovian environment is assumed, in which the probability of choosing a specific annual matrix depends on which annual matrix is currently chosen. If an element of the list is a vector, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices.

format — An optional integer indicating the kind of function-based MPM to create, if argument vrm is provided. Possible choices include: 1, Ehrlen-format historical MPM; 2, deVries-format historical MPM; 3, ahistorical MPM (default); 4, age-by-stage MPM; and 5, Leslie (age-based) MPM. Defaults to 3.

entry_time — An optional integer vector giving the entry time for each variant into each simulation. Defaults to a zero vector with length equal to the number of variants to run concurrently in each simulation, as given by argument var_per_run. Note that if two variants are to be run at a time, as in a pairwise invasion analysis, then the length of the vector should be equal to 2.

sp_density — An optional argument for use with argument vrm that specifies the spatial density to be used in each time step. If used, then may either be a numeric vector giving a single spatial density for each time step. If vectors are shorter than specified in times, then these values will be cycled.

ind_terms — An optional argument providing values of individual or environmental covariate values for argument vrm. Should be set to a single data frame with 3 columns giving values for up to 3 covariates across time (rows give the time order of these values). Unused terms within the data frame must be set to 0 (use of NA will produce errors). If the number of rows is less than times, then these values will be cycled.

dev_terms — An optional data frame including 14 columns and up to times rows showing the values of the deviation terms to be added to each linear vital rate. The column order should be: 1: survival, 2: observation, 3: primary size, 4: secondary size, 5: tertiary size, 6: reproduction, 7: fecundity, 8: juvenile survival, 9: juvenile observation, 10: juvenile primary size, 11: juvenile secondary size, 12: juvenile tertiary size, 13: juvenile reproduction, and 14: juvenile maturity transition. Unused terms must be set to 0 (use of NA will produce errors). Single or small numbers of values per vital rate model are also allowed, and if the number of rows is less than times, then the terms will be cycled.

fb_sparse — A logical value indicating whether function-based MPMs should be produced in sparse matrix format. Defaults to FALSE.

`firstage` An optional integer used for function-based Leslie and age-by-stage MPMs giving the starting age in such MPMs. Use only if the MPM is both function-based and has age structure. Typically, the starting age in such MPMs should be set to `0` if post-breeding and `1` if pre-breeding. All other MPMs should be set to `0`.

`finalage` An optional integer used for function-based Leslie and age-by-stage MPMs giving the final age in such MPMs. Use only if the MPM is both function-based and has age structure.

`fecage_min` An optional integer used for function-based Leslie MPMs giving the first age at which organisms can reproduce. Use only if the MPM is both function-based and has age structure. Defaults to the value given in `firstage`.

`fecage_max` An optional integer used for function-based Leslie MPMs giving the final age at which organisms can reproduce. Use only if the MPM is both function-based and has age structure. Defaults to the value given in `finalage`.

`cont` An optional logical value for function-based Leslie and age-by-stage MPMs stating whether the MPM should should include a stasis transition within the final age. This should be used only when an organism can maintain the demographic characteristics of the final described age after reaching that age.

`prebreeding` An optional logical value indicating whether the life cycle is prebreeding (`TRUE`) or postbreeding (`FALSE`). Defaults to `TRUE`.

`fecmod` An optional numeric value for function-based MPMs giving scalar multipliers for fecundity terms, when two fecundity variables are used for a collective fecundity per individual.

`density` An optional data frames of class `lefkoDens`, which provides details for density dependence in MPM elements and is created with function `density_input()`. Defaults to `NULL`, in which case no density dependence is built into matrix elements.

`density_vr` An optional data frame of class `lefkoDensVR`, which provides details for density dependence in vital rate models and has been created with function `link[lefko3]{density_vr}()`. Can only be used with function-based projections. Defaults to `NULL`, in which case no density dependence is built into vital rates.

`stochastic` A logical value indicating whether the projection will be run as a temporally stochastic projection. Defaults to `FALSE`.

`A_only` A logical value indicating whether to alter survival and fecundity matrix elements separately prior to creating the overall `A` matrix, or whether to alter elements directly on `A` matrices. Defaults to `TRUE`, and should be kept to that setting unless some matrix elements to be altered are sums of survival and fecundity transitions.

`integeronly` A logical value indicating whether to round the number of individuals projected in each stage at each occasion down to the next lower integer. Defaults to `FALSE`.

`fitness_table` A logical value dictating whether to include a data frame giving Lyapunov coefficients for all combinations of variants tested. Necessary for the creation of pairwise invasibility plots (PIPs). Defaults to `TRUE`.

`trait_optima` A logical value indicating whether to assess the optimal values of traits, generally as kinds of evolutionary stage equilibrium (ESS) points. Trait optimization is conducted via elasticity analysis of traits that are variable within the `trait_axis` table. Defaults to `FALSE`.

zap_min	A logical value indicating whether to treat traits and fitness as 0 when their absolute values are less than the value given in argument threshold.
converged_only	A logical value indicating whether to show predicted trait optima only in cases where the Lyapunov coefficient in elasticity analysis has converged to 0. Defaults to TRUE.
err_check	A logical value indicating whether to include an extra list of output objects for error checking. Can also be set to the text value "extreme", in which case all err_check output plus a multiple level list with each MPM used in each time step will be output.
var_per_run	The number of variants to run in each simulation. Defaults to 2, resulting in pairwise invasibility analysis. See Notes for details.
substoch	An integer value indicating whether to force survival- transition matrices to be substochastic in density dependent and density independent simulations. Defaults to 0, which does not enforce substochasticity. Alternatively, 1 forces all survival-transition elements to range from 0.0 to 1.0, and forces fecundity to be non-negative; and 2 forces all column rows in the survival-transition matrices to total no more than 1.0, in addition to the actions outlined for option 1. Both settings 1 and 2 change negative fecundity elements to 0.0, while setting 0 does not alter fecundity.
elast_mult	A multiplier for traits to assess the elasticity of fitness in trait optimization. Defaults to 0.995.
nreps	The number of replicate projections. Defaults to 1.
times	Number of occasions to iterate per replicate. Defaults to 10000.
fitness_times	An integer giving the number of time steps at the end of each run to use to estimate the fitness of the respective genotype. Defaults to 100, but if times < 100, then is set equal to times.
exp_tol	A numeric value used to indicate a maximum value to set exponents to in the core kernel to prevent numerical overflow. Defaults to 700.
theta_tol	A numeric value used to indicate a maximum value to theta as used in the negative binomial probability density kernel. Defaults to 100000000, but can be reset to other values during error checking.
threshold	The lower limit for the absolute value of fitness, below which fitness is rounded to 0. Defaults to 0.00000001.
loop_max	An integer value denoting the number of search cycles allowed per ESS during ESS optimization. Defaults to 150.

Value

A list of class adaptInv, with the following elements:

fitness	A data frame giving the Lyapunov coefficients estimated for each variant, per replicate.
variants_out	A two-level list with the top level list having number of elements equal to the number of variants, and the lower level corresponding to the number of replicates. Each element of the lower level list is a matrix showing the number of individuals in each stage (row) at each time (column).

N_out — A list with the number of elements equal to the number of replicates. Each element within this list is data frame showing the number of individuals of each species or genotype alive at each time. The number of rows are equal to the number of MPMs used, and the columns correspond to the time steps.

stageframe_list — A list in which each element is the stageframe for each MPM used.

hstages_list — A list giving the used hstages data frames, which identify the correct stage pairing for each row / column in each historical MPM utilized.

agestages_list — A list giving the used agestages data frames, which identify the correct age-stage pairing for each row / column in each age-by-stage MPM utilized.

labels — A small data frame giving the the population and patch identities for each MPM entered.

err_check — An optional list composed of an additional six lists, each of which has the number of elements equal to the number of MPMs utilized. List output include allstages_all, which gives the indices of estimated transitions in MPMs constructed by function invade3() from input vital rate models; allmodels_all, which provides all vital rate models as decomposed and interpreted by function invade3(); equivalence_list, which gives the stage equivalence for density calculations across MPMs; density_list, which gives the density inputs utilized; dens_index_list, which provides indices used to identify matrix elements for density dependence; and density_vr_list, which gives the density_vr inputs utilized.

Notes

The argument var_per_run establishes the style of simulation to run. Entering var_per_run = 1 runs each variant singly. Entering var_per_run = 2 runs pairwise invasibility analysis, trying each pair permutation of variants. Greater values will lead to multiple invasibility analysis with different permutations of groups. For example, var_per_run = 3 runs each permutation of groups of three. The integer set must be positive, and must not be larger than the number of variants.

When optima = TRUE, ESS values for traits that vary in the input adaptAxis data frame are evaluated. The methodology is that originally developed in Benton and Grant (1999, Evolution 53:677-688), as communicated in Roff (2010, Modeling evolution: an introduction to numerical methods, Oxford University Press). In essence, function invade3 determines which traits vary among all traits noted in the input trait axis. A new trait axis is then created with values of variable traits multiplied by 0.995, and this new trait axis is composed entirely of invaders that will be paired against each respective row in the original trait axis. These two trait axis frames are then used to conduct pairwise invasibility elasticity analyses, particularly noting where fitness values and trends invert. Note that this optimization approach really only works with one variable trait.

Examples

```
library(lefko3)
data(cypdata)

sizevector <- c(0, 0, 0, 0, 1, 2.5, 4.5, 8, 17.5)
stagevector <- c("SD", "P1", "SL", "D", "XSm", "Sm", "Md", "Lg", "XLg")
repvector <- c(0, 0, 0, 0, 1, 1, 1, 1, 1)
```

```
obsvector <- c(0, 0, 0, 0, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 0, 1, 1, 1, 1, 1, 1)
immvector <- c(0, 1, 1, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 0, 0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 0, 0, 0.5, 0.5, 1, 1, 2.5, 7)

cypframe_raw <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  propstatus = propvector, immstatus = immvector, indataset = indataset,
  binhalfwidth = binvec)

cypraw_v1 <- verticalize3(data = cypdata, noyears = 6, firstyear = 2004,
  patchidcol = "patch", individcol = "plantid", blocksize = 4,
  sizeacol = "Inf2.04", sizebcol = "Inf.04", sizeccol = "Veg.04",
  repstracol = "Inf.04", repstrbcol = "Inf2.04", fecacol = "Pod.04",
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cypsupp2r <- supplemental(stage3 = c("SD", "P1", "SL", "D",
    "XSm", "Sm", "SD", "P1"),
  stage2 = c("SD", "SD", "P1", "SL", "SL", "SL", "rep",
    "rep"),
  eststage3 = c(NA, NA, NA, "D", "XSm", "Sm", NA, NA),
  eststage2 = c(NA, NA, NA, "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.10, 0.40, 0.25, NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, 1000, 1000),
  type =c(1, 1, 1, 1, 1, 1, 3, 3),
  stageframe = cypframe_raw, historical = FALSE)

cypmatrix2r <- rlefko2(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")
cypmean <- lmean(cypmatrix2r)

cyp_start <- start_input(cypmean, stage2 = c("SD", "P1", "D"),
  value = c(1000, 200, 4))

c2d_4 <- density_input(cypmean, stage3 = c("P1", "P1"), stage2= c("SD", "rep"),
  style = 2, time_delay = 1, alpha = 0.005, beta = 0.000005, type = c(2, 2))

# A simple projection allows us to find a combination of density dependence
# and running time that produces a stable quasi-equilibrium
cyp_proj <- projection3(cypmean, times = 250, start_frame = cyp_start,
  density = c2d_4, integeronly = TRUE)
plot(cyp_proj)

cyp_ta <- trait_axis(stageframe = cypframe_raw,
  stage3 = rep("P1", 15),
  stage2 = rep("rep", 15),
  multiplier = seq(from = 0.1, to = 10.0, length.out = 15),
  type = rep(2, 15))
```

```
cyp_inv <- invade3(axis = cyp_ta, mpm = cypmean, density = c2d_4, times = 350,
  starts = cyp_start, entry_time = c(0, 250), fitness_times = 30,
  var_per_run = 2)
plot(cyp_inv)
```

`plot.adaptInv` *Create Contour Plot of Pairwise Invasibility Analysis Results*

Description

Function `plot.adaptInv` plots pairwise invasibility contour plots. This function is based on code derived from Roff's Modeling Evolution: An Introduction to Numerical Methods (2010, Oxford University Press).

Usage

```
## S3 method for class 'adaptInv'
plot(
  x,
  xlab = "Resident",
  ylab = "Invader",
  res_variant = 1,
  inv_variant = 2,
  repl = 1,
  pip = TRUE,
  elast = FALSE,
  run = 1,
  filled = TRUE,
  plot.title,
  plot.axes,
  axes = TRUE,
  frame.plot = TRUE,
  auto_ylim = TRUE,
  auto_col = TRUE,
  auto_lty = TRUE,
  auto_title = FALSE,
  ...
)
```

Arguments

`x`	An `adaptInv` object, created with function `invade3()`.
`xlab`	The x axis label forthe contour plot. Defaults to `Resident`.
`ylab`	The y axis label forthe contour plot. Defaults to `Invader`.
`res_variant`	The number of the variant representing the resident subpopulation.

inv_variant	The number of the variant representing the mutant subpopulation.
repl	The replicate number to plot, in the fitness data frame within the adaptInv object entered in argument x.
pip	A logical value indicating whether to produce a pairwise invasibility plot. If FALSE, then will produce a diagnostic population size plot. Defaults to TRUE.
elast	A logical value indicating whether to produce an elasticity plot. Such plots can only be produced when trait optimization is performed during invasibility analysis. Defaults to FALSE.
run	An integer giving the run to plot if pip = FALSE.
filled	A logical value indicating whether to produce a filled contour plot, or a standard contour plot. Defaults to TRUE, but reverts if invader fitness is consistently positive, or consistently negative, relative to the resident.
plot.title	A title for the plot.
plot.axes	A generic parameter providing axis information for pairwise invasibility plots.
axes	A logical value indicating whether to include axis lines. Defaults to TRUE.
frame.plot	A logical value indicating whether to frame the plot.
auto_ylim	A logical value indicating whether the maximum of the y axis should be determined automatically. Defaults to TRUE, but reverts to FALSE if any setting for ylim is given. Used only if pip = FALSE.
auto_col	A logical value indicating whether to shift the color of lines associated with each patch automatically. Defaults to TRUE, but reverts to FALSE if any setting for col is given. Used only if pip = FALSE.
auto_lty	A logical value indicating whether to shift the line type associated with each replicate automatically. Defaults to TRUE, but reverts to FALSE if any setting for lty is given. Used only if pip = FALSE.
auto_title	A logical value indicating whether to add a title to each plot. The plot is composed of the concatenated population and patch names. Defaults to FALSE. Used only if pip = FALSE.
...	Other parameters used by functions plot.default().

Value

A contour plot showing the overall fitness dynamics of the invader variant, assuming a pairwise invasibility analysis.

Notes

By default, function plot.adaptInv produces a filled contour plot in which grey regions show where the invader has positive fitness relative to the resident, and white regions show where the invader has negative fitness relative to the resident. Fitness here refers to the Lyapunov coefficient, calculated over the final fitness_times in the original call to function invade3().

Examples

```
library(lefko3)
data(cypdata)

sizevector <- c(0, 0, 0, 0, 1, 2.5, 4.5, 8, 17.5)
stagevector <- c("SD", "P1", "SL", "D", "XSm", "Sm", "Md", "Lg", "XLg")
repvector <- c(0, 0, 0, 0, 1, 1, 1, 1, 1)
obsvector <- c(0, 0, 0, 0, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 0, 1, 1, 1, 1, 1, 1)
immvector <- c(0, 1, 1, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 0, 0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 0, 0, 0.5, 0.5, 1, 1, 2.5, 7)

cypframe_raw <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  propstatus = propvector, immstatus = immvector, indataset = indataset,
  binhalfwidth = binvec)

cypraw_v1 <- verticalize3(data = cypdata, noyears = 6, firstyear = 2004,
  patchidcol = "patch", individcol = "plantid", blocksize = 4,
  sizeacol = "Inf2.04", sizebcol = "Inf.04", sizeccol = "Veg.04",
  repstracol = "Inf.04", repstrbcol = "Inf2.04", fecacol = "Pod.04",
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cypsupp2r <- supplemental(stage3 = c("SD", "P1", "SL", "D",
    "XSm", "Sm", "SD", "P1"),
  stage2 = c("SD", "SD", "P1", "SL", "SL", "SL", "rep",
    "rep"),
  eststage3 = c(NA, NA, NA, "D", "XSm", "Sm", NA, NA),
  eststage2 = c(NA, NA, NA, "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.10, 0.40, 0.25, NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, 1000, 1000),
  type =c(1, 1, 1, 1, 1, 1, 3, 3),
  stageframe = cypframe_raw, historical = FALSE)

cypmatrix2r <- rlefko2(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")
cypmean <- lmean(cypmatrix2r)

cyp_start <- start_input(cypmean, stage2 = c("SD", "P1", "D"),
  value = c(1000, 200, 4))

c2d_4 <- density_input(cypmean, stage3 = c("P1", "P1"), stage2= c("SD", "rep"),
  style = 2, time_delay = 1, alpha = 0.005, beta = 0.000005, type = c(2, 2))

# A simple projection allows us to find a combination of density dependence
# and running time that produces a stable quasi-equilibrium
cyp_proj <- projection3(cypmean, times = 250, start_frame = cyp_start,
```

```
  density = c2d_4, integeronly = TRUE)
plot(cyp_proj)

cyp_ta <- trait_axis(stageframe = cypframe_raw,
  stage3 = rep("P1", 15),
  stage2 = rep("rep", 15),
  multiplier = seq(from = 0.1, to = 10.0, length.out = 15),
  type = rep(2, 15))

cyp_inv <- invade3(axis = cyp_ta, mpm = cypmean, density = c2d_4, times = 350,
  starts = cyp_start, entry_time = c(0, 250), fitness_times = 30,
  var_per_run = 2)
plot(cyp_inv)
```

`plot.adaptProj` *Create Plot of Community Projection*

Description

Function `plot.adaptProj` plots community projections.

Usage

```
## S3 method for class 'adaptProj'
plot(
  x,
  repl = 1,
  auto_ylim = TRUE,
  auto_col = TRUE,
  auto_lty = TRUE,
  auto_title = FALSE,
  ...
)
```

Arguments

`x`	An `adaptProj` object.
`repl`	The replicate to plot. Defaults to `1`, in which case the first replicate is plotted.
`auto_ylim`	A logical value indicating whether the maximum of the y axis should be determined automatically. Defaults to `TRUE`, but reverts to `FALSE` if any setting for `ylim` is given.
`auto_col`	A logical value indicating whether to shift the color of lines associated with each patch automatically. Defaults to `TRUE`, but reverts to `FALSE` if any setting for `col` is given.
`auto_lty`	A logical value indicating whether to shift the line type associated with each replicate automatically. Defaults to `TRUE`, but reverts to `FALSE` if any setting for `lty` is given.

auto_title	A logical value indicating whether to add a title to each plot. The plot is composed of the concatenated population and patch names. Defaults to FALSE.
...	Other parameters used by functions plot.default() and lines().

Value

A plot of the results of a project3() run.

Notes

Output plots are currently limited to time series of population size.

Examples

```
library(lefko3)
data(cypdata)

data(cypa_data)

sizevector <- c(0, 0, 0, 0, 0, 0, 1, 2.5, 4.5, 8, 17.5)
stagevector <- c("SD", "P1", "P2", "P3", "SL", "D", "XSm", "Sm", "Md", "Lg",
  "XLg")
repvector <- c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
obsvector <- c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
immvector <- c(0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 0, 0, 0, 0, 0.5, 0.5, 1, 1, 2.5, 7)

cypframe_raw <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  propstatus = propvector, immstatus = immvector, indataset = indataset,
  binhalfwidth = binvec)

cycaraw_v1 <- verticalize3(data = cypdata, noyears = 6, firstyear = 2004,
  patchidcol = "patch", individcol = "plantid", blocksize = 4,
  sizeacol = "Inf2.04", sizebcol = "Inf.04", sizeccol = "Veg.04",
  repstracol = "Inf.04", repstrbcol = "Inf2.04", fecacol = "Pod.04",
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cyparaw_v1 <- verticalize3(data = cypa_data, noyears = 18, firstyear = 1994,
  individcol = "plant_id", blocksize = 2, sizeacol = "Inf.94",
  sizebcol = "Veg.94", repstracol = "Inf.94", fecacol = "Inf.94",
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cypsupp2r <- supplemental(stage3 = c("SD", "P1", "P2", "P3", "SL", "D",
    "XSm", "Sm", "SD", "P1"),
  stage2 = c("SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "rep",
    "rep"),
```

```
  eststage3 = c(NA, NA, NA, NA, NA, "D", "XSm", "Sm", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.10, 0.20, 0.20, 0.20, 0.25, NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, 0.5, 0.5),
  type =c(1, 1, 1, 1, 1, 1, 1, 3, 3),
  stageframe = cypframe_raw, historical = FALSE)
cyp_supp_list1 <- list(cypsupp2r, cypsupp2r)

cycamatrix2r <- rlefko2(data = cycaraw_v1, stageframe = cypframe_raw,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", indivcol = "individ")

cypamatrix2r <- rlefko2(data = cyparaw_v1, stageframe = cypframe_raw,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", indivcol = "individ")

cyp_mpm_list <- list(cycamatrix2r, cypamatrix2r)

cyca2_start <- start_input(cycamatrix2r, stage2 = c("SD", "P1", "P2"),
  value = c(500, 100, 200))
cypa2_start <- start_input(cypamatrix2r, stage2 = c("SD", "P1", "P2"),
  value = c(5000, 1000, 2000))
cyp_start_list <- list(cyca2_start, cypa2_start)

cyp2_dv <- density_input(cypamatrix2r, stage3 = c("SD", "P1"),
  stage2 = c("rep", "rep"), style = c(1, 1), alpha = c(0.5, 1.2),
  beta = c(1.0, 2.0), type = c(2, 1))
cyp_dv_list <- list(cyp2_dv, cyp2_dv)

cyp_comm_proj <- project3(mpms = cyp_mpm_list, starts = cyp_start_list,
  density = cyp_dv_list, times = 10)

plot(cyp_comm_proj, lwd = 2, bty = "n")
```

`project3` *Project Multiple MPMs With or Without Density Dependence*

Description

Function `project3` uses pre-existing or function-based MPMs to run community projection simulations, in which different populations are run as separate MPMs. Density dependence can be used with individual equivalence vectors specifying Lotka-Volterra coefficients to adjust overall population sizes to make them comparable.

Usage

```
project3(
```

```
  mpms = NULL,
  vrms = NULL,
  stageframes = NULL,
  supplements = NULL,
  equivalence = NULL,
  starts = NULL,
  years = NULL,
  patches = NULL,
  tweights = NULL,
  format = NULL,
  entry_time = NULL,
  sp_density = NULL,
  ind_terms = NULL,
  dev_terms = NULL,
  fb_sparse = NULL,
  firstage = NULL,
  finalage = NULL,
  fecage_min = NULL,
  fecage_max = NULL,
  cont = NULL,
  fecmod = NULL,
  density = NULL,
  density_vr = NULL,
  err_check = NULL,
  stochastic = FALSE,
  integeronly = FALSE,
  substoch = 0L,
  nreps = 1L,
  times = 10000L,
  prep_mats = 20L,
  force_fb = FALSE,
  exp_tol = 700,
  theta_tol = 1e+08
)
```

Arguments

mpms	An optional list of MPMs. Each MPM must be of class `lefkoMat`.
vrms	An optional list of `vrm_input` objects, each corresponding to a distinct MPM that will be created during projection. Each `vrm_input` object requires its own stageframe, entered in the same order via argument `stageframes`.
stageframes	An optional list of stageframes, corresponding in number and order to the MPMs in argument `vrms`. Each stageframe must be of class `stageframe`.
supplements	An optional list of data frames of class `lefkoSD` that provide supplemental data that should be incorporated into function-based MPMs. If used, then should be the same number of data frames as the number of MPMs provided in the list for argument `vrms`. MPMs that do not need supplemental data should be entered as `NULL` in this list. See `supplemental()` for details.

equivalence — An optional numeric vector, list of numeric vectors, data frame of class adaptEq, or list of data frames of class adaptEq. If a numeric vector, then must have the same number of elements as the number of MPMs, with each element giving the effect of an individual of each MPM relative to a reference individual. If a list of vectors, then the list should be composed of as many numeric vectors as MPMs, with each vector giving the effect of each individual in each stage relative to a reference individual. Data frames of class adaptEq, and lists of such data frames, can be made with function equiv_input(). Numeric entries used in these vectors can be thought of as Lotka-Volterra interaction terms, such as are used in multiple species competition models.

starts — An optional list of lefkoSV objects, which are data frames providing the starting numbers of individuals of each stage. If provided, then one is needed per MPM. If not provided, then all projections start with a single individual of each stage per MPM.

years — An optional term corresponding either to a single integer vector of time t values, if all MPMs will use the same time t or set of time t's, or a list of such vectors with each vector corresponding to each MPM in order. In the latter case, a vector composed of a single NA value is interpreted to mean that all time t values in the MPM should be utilized. If a vector shorter than times is supplied, then this vector will be cycled.

patches — An optional string vector with length equal to the number of MPMs, detailing the name of each patch to project for each MPM, in order. Only a single pop-patch may be projected for each MPM given. A value of NA can be supplied to indicate that the population-level matrices should be projected (if argument mpms is used and a population-level set of matrices exist), or that the first patch noted should be used. Defaults to the population-level set or the first patch, depending on whether the former exists.

tweights — An optional list composed of numeric vectors or matrices denoting the probabilities of choosing each matrix in each MPM in a stochastic projection. If an element of the list is a matrix, then a first-order Markovian environment is assumed, in which the probability of choosing a specific annual matrix depends on which annual matrix is currently chosen. If an element of the list is a vector, then the choice of annual matrix is assumed to be independent of the current matrix. Defaults to equal weighting among matrices. If used, then one element per MPM is required, with equal weighting assumed for any element set to NULL.

format — An optional integer vector indicating the kind of function-based MPM to create for each vrm_input object entered in argument vrms. Possible choices include: 1, Ehrlen-format historical MPM; 2, deVries-format historical MPM; 3, ahistorical MPM (default); 4, age-by-stage MPM; and 5, Leslie (age-based) MPM.

entry_time — An optional integer vector giving the entry time for each MPM into the projection. Defaults to a zero vector with the length of the number of MPMs, as given either by argument mpms or vrms.

sp_density — An optional argument for use with vrm_input objects that specifies the spatial density to be used in each time step. If used, may either be a numeric vector giving a single spatial density for each vrm_input object entered in argument vrms (in this case, the value of spatial density given for each vrm_input object will be held constant through the projection), or a list of as many numeric vectors

as vrm_input objects, with the length of each vector giving the spatial density at each time step. If vectors are shorter than specified in times, then these values will be cycled.

ind_terms An optional argument providing values of individual or environmental covariate values for vrm_input objects used in function-based projection. Can be set either to a single data frame with 3 columns giving values for up to 3 covariates across time (rows give the time order of these values), or a list of as many such data frames as vrm_input objects. In the latter case, vrm_input objects that do not use such covariates should have the associated element set to NULL. Unused terms within each data frame must be set to 0 (use of NA will produce errors.) If the number of rows is less than times, then these values will be cycled.

dev_terms An optional list of data frames, one for each vrm_input object. Each should include 14 columns and up to times rows showing the values of the deviation terms to be added to each linear vital rate. The column order should be: 1: survival, 2: observation, 3: primary size, 4: secondary size, 5: tertiary size, 6: reproduction, 7: fecundity, 8: juvenile survival, 9: juvenile observation, 10: juvenile primary size, 11: juvenile secondary size, 12: juvenile tertiary size, 13: juvenile reproduction, and 14: juvenile maturity transition. Unused terms must be set to 0 (use of NA will produce errors). Single or small numbers of values per vital rate model are also allowed, and if the number of rows is less than times, then the terms will be cycled.

fb_sparse A logical vector indicating whether function-based MPMs should be produced in sparse matrix format. Defaults to FALSE for each MPM.

firstage An optional integer vector used for function-based Leslie and age-by-stage MPMs giving the starting ages in such MPMs. Use only if at least one MPM is both function-based and has age structure. Typically, the starting age in such MPMs should be set to 0 if post-breeding and 1 if pre-breeding. All other MPMs should be set to 0. Do not use if no MPM has age structure.

finalage An optional integer vector used for function-based Leslie and age-by-stage MPMs giving the final ages in such MPMs. Use only if at least one MPM is both function-based and has age structure. Do not use if no MPM has age structure.

fecage_min An optional integer vector used for function-based Leslie MPMs giving the first age at which organisms can be reproductive in such MPMs. Use only if at least one MPM is a function-based Leslie MPM. Defaults to the values given in firstage.

fecage_max An optional integer vector used for function-based Leslie MPMs giving the final age at which organisms can be reproductive in such MPMs. Use only if at least one MPM is a function-based Leslie MPM. Defaults to the values given in finalage.

cont An optional vector used for function-based Leslie and age-by-stage MPMs stating whether the MPM should should include a stasis transition within the final age. This should be used only when an organism can maintain the demographic characteristics of the final described age after reaching that age. Can be entered as a logical vector or an integer vector. MPMs without age structure should be entered as 0 or FALSE. Do not use if no MPM has age structure.

`fecmod` An optional vector used for function-based MPMs giving scalar multipliers for fecundity terms, when two fecundity variables are used for a collective fecundity per individual. Each entry refers to each `vrm_input` object in argument `vrms`, in the same order.

`density` An optional list of data frames of class `lefkoDens`, which provide details for density dependence in MPM elements and have been created with function `density_input()`. If used, then one such data frame per MPM is required. MPMs to be run without density dependence should be set to `NULL`.

`density_vr` An optional list of data frames of class `lefkoDensVR`, which provide details for density dependence in vital rate models and have been created with function `link[lefko3]{density_vr}()`. If used, then one such data frame per MPM is required. MPMs to be run without vital describing density dependence relationships in vital rates should be set to `NULL`. Can only be used with function-based projections.

`err_check` A logical value indicating whether to include an extra list of output objects for error checking. Can also be set to the text value `"extreme"`, in which case all `err_check` output plus a multiple level list with each MPM used in each time step will be output.

`stochastic` A logical value indicating whether the projection will be run as a temporally stochastic projection. Defaults to `FALSE`.

`integeronly` A logical value indicating whether to round the number of individuals projected in each stage at each occasion in each MPM to the nearest integer. Defaults to `FALSE`.

`substoch` An integer value indicating whether to force survival- transition matrices to be substochastic in density dependent and density independent simulations. Defaults to `0`, which does not enforce substochasticity. Alternatively, `1` forces all survival-transition elements to range from 0.0 to 1.0, and forces fecundity to be non-negative; and `2` forces all column rows in the survival-transition matrices to total no more than 1.0, in addition to the actions outlined for option `1`. Both settings `1` and `2` change negative fecundity elements to `0.0`.

`nreps` The number of replicate projections. Defaults to `1`.

`times` Number of occasions to iterate per replicate. Defaults to `10000`.

`prep_mats` An integer value for use when creating function-based MPM projections. If using `vrms` input instead of `mpms` input, then this argument determines how many matrices should be used as a limit to develop matrices prior to running the projection. See `Notes` for further details.

`force_fb` A logical value indicating whether to force function-based MPMs to be developed at each time step even if fewer than `prep_mats`. Defaults to `FALSE`.

`exp_tol` A numeric value used to indicate a maximum value to set exponents to in the core kernel to prevent numerical overflow. Defaults to `700`.

`theta_tol` A numeric value used to indicate a maximum value to theta as used in the negative binomial probability density kernel. Defaults to `100000000`, but can be reset to other values during error checking.

Value

A list of class adaptProj, with the following elements:

comm_out — A two-level list with the top level list having number of elements equal to the number of MPMs used as input, and the lower level corresponding to the number of replicates. Each element of the lower level list is a data frame showing the number of individuals in each stage at each time. Rows and columns in the data frames correspond to stages and time steps, respectively.

N_out — A list with the number of elements equal to the number of replicates. Each element within this list is data frame showing the number of individuals of each species or genotype alive at each time. The number of rows are equal to the number of MPMs used, and the columns correspond to the time steps.

stageframe_list — A list in which each element is the stageframe for each MPM used.

hstages_list — A list giving the used hstages data frames, which identify the correct stage pairing for each row / column in each historical MPM utilized.

agestages_list — A list giving the used agestages data frames, which identify the correct age-stage pairing for each row / column in each age-by-stage MPM utilized.

labels — A small data frame giving the the population and patch identities for each MPM entered.

err_check — An optional list composed of an additional six lists, each of which has the number of elements equal to the number of MPMs utilized. List output include allstages_all, which gives the indices of estimatedtransitions in MPMs constructed by function project3() from input vital rate models; allmodels_all, which provides all vital rate models as decomposed and interpreted by function project3(); equivalence_list, which gives the stage equivalence for density calculations across MPMs; density_list, which gives the density inputs utilized; dens_index_list, which provides indices used to identify matrix elements for density dependence; and density_vr_list, which gives the density_vr inputs utilized.

Notes

This function has been optimized in the function-based approach such that if there are relatively few matrices required per MPM to run the projection forward, then these matrices will be made prior to running the projection. This approach saves time, but only if there are relatively few unique matrices required for each MPM. If many or only unique MPMs are required at each time step, then the matrices will be made on the fly during the projection itself. Such a situation will most likely occur if each time step requires a new matrix resulting from a unique individual covariate value, or if the density_vr argument is used. The key argument determining this behavior is prep_mats, which provides the maximum limit for the number of matrices required per MPM in order to create matrices prior to projection.

Examples

```
library(lefko3)
data(cypdata)
```

```
data(cypa_data)

sizevector <- c(0, 0, 0, 0, 0, 0, 1, 2.5, 4.5, 8, 17.5)
stagevector <- c("SD", "P1", "P2", "P3", "SL", "D", "XSm", "Sm", "Md", "Lg",
  "XLg")
repvector <- c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
obsvector <- c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
immvector <- c(0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 0, 0, 0, 0, 0.5, 0.5, 1, 1, 2.5, 7)

cypframe_raw <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  propstatus = propvector, immstatus = immvector, indataset = indataset,
  binhalfwidth = binvec)

cycaraw_v1 <- verticalize3(data = cypdata, noyears = 6, firstyear = 2004,
  patchidcol = "patch", individcol = "plantid", blocksize = 4,
  sizeacol = "Inf2.04", sizebcol = "Inf.04", sizeccol = "Veg.04",
  repstracol = "Inf.04", repstrbcol = "Inf2.04", fecacol = "Pod.04",
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cyparaw_v1 <- verticalize3(data = cypa_data, noyears = 18, firstyear = 1994,
  individcol = "plant_id", blocksize = 2, sizeacol = "Inf.94",
  sizebcol = "Veg.94", repstracol = "Inf.94", fecacol = "Inf.94",
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cypsupp2r <- supplemental(stage3 = c("SD", "P1", "P2", "P3", "SL", "D",
    "XSm", "Sm", "SD", "P1"),
  stage2 = c("SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "rep",
    "rep"),
  eststage3 = c(NA, NA, NA, NA, NA, "D", "XSm", "Sm", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.10, 0.20, 0.20, 0.20, 0.25, NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, 0.5, 0.5),
  type =c(1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  stageframe = cypframe_raw, historical = FALSE)
cyp_supp_list1 <- list(cypsupp2r, cypsupp2r)

cycamatrix2r <- rlefko2(data = cycaraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")

cypamatrix2r <- rlefko2(data = cyparaw_v1, stageframe = cypframe_raw,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")
```

```
cyp_mpm_list <- list(cycamatrix2r, cypamatrix2r)

cyca2_start <- start_input(cycamatrix2r, stage2 = c("SD", "P1", "P2"),
  value = c(500, 100, 200))
cypa2_start <- start_input(cypamatrix2r, stage2 = c("SD", "P1", "P2"),
  value = c(5000, 1000, 2000))
cyp_start_list <- list(cyca2_start, cypa2_start)

cyp2_dv <- density_input(cypamatrix2r, stage3 = c("SD", "P1"),
  stage2 = c("rep", "rep"), style = c(1, 1), alpha = c(0.5, 1.2),
  beta = c(1.0, 2.0), type = c(2, 1))
cyp_dv_list <- list(cyp2_dv, cyp2_dv)

cyp_comm_proj <- project3(mpms = cyp_mpm_list, starts = cyp_start_list,
  density = cyp_dv_list, times = 10)

summary(cyp_comm_proj)
```

`summary.adaptInv` *Summarize adaptInv Objects*

Description

Function `summary.adaptInv()` summarizes `adaptInv` objects.

Usage

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

Arguments

`object`	An `adaptInv` object.
`...`	Other parameters currently not utilized.

Value

This function only produces text summarizing the numbers of variants, time steps, replicates, ESS optima, etc.

Examples

```
library(lefko3)
data(cypdata)

sizevector <- c(0, 0, 0, 0, 1, 2.5, 4.5, 8, 17.5)
stagevector <- c("SD", "P1", "SL", "D", "XSm", "Sm", "Md", "Lg", "XLg")
```

```
repvector <- c(0, 0, 0, 0, 1, 1, 1, 1, 1)
obsvector <- c(0, 0, 0, 0, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 0, 1, 1, 1, 1, 1, 1)
immvector <- c(0, 1, 1, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 0, 0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 0, 0, 0.5, 0.5, 1, 1, 2.5, 7)

cypframe_raw <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  propstatus = propvector, immstatus = immvector, indataset = indataset,
  binhalfwidth = binvec)

cypraw_v1 <- verticalize3(data = cypdata, noyears = 6, firstyear = 2004,
  patchidcol = "patch", individcol = "plantid", blocksize = 4,
  sizeacol = "Inf2.04", sizebcol = "Inf.04", sizeccol = "Veg.04",
  repstracol = "Inf.04", repstrbcol = "Inf2.04", fecacol = "Pod.04",
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cypsupp2r <- supplemental(stage3 = c("SD", "P1", "SL", "D",
    "XSm", "Sm", "SD", "P1"),
  stage2 = c("SD", "SD", "P1", "SL", "SL", "SL", "rep",
    "rep"),
  eststage3 = c(NA, NA, NA, "D", "XSm", "Sm", NA, NA),
  eststage2 = c(NA, NA, NA, "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.10, 0.40, 0.25, NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, 1000, 1000),
  type =c(1, 1, 1, 1, 1, 1, 3, 3),
  stageframe = cypframe_raw, historical = FALSE)

cypmatrix2r <- rlefko2(data = cypraw_v1, stageframe = cypframe_raw,
  year = "all", patch = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", patchcol = "patchid", indivcol = "individ")
cypmean <- lmean(cypmatrix2r)

cyp_start <- start_input(cypmean, stage2 = c("SD", "P1", "D"),
  value = c(1000, 200, 4))

c2d_4 <- density_input(cypmean, stage3 = c("P1", "P1"), stage2= c("SD", "rep"),
  style = 2, time_delay = 1, alpha = 0.005, beta = 0.000005, type = c(2, 2))

# A simple projection allows us to find a combination of density dependence
# and running time that produces a stable quasi-equilibrium
cyp_proj <- projection3(cypmean, times = 250, start_frame = cyp_start,
  density = c2d_4, integeronly = TRUE)
plot(cyp_proj)

cyp_ta <- trait_axis(stageframe = cypframe_raw,
  stage3 = rep("P1", 15),
  stage2 = rep("rep", 15),
  multiplier = seq(from = 0.1, to = 10.0, length.out = 15),
```

```
  type = rep(2, 15))

cyp_inv <- invade3(axis = cyp_ta, mpm = cypmean, density = c2d_4, times = 350,
  starts = cyp_start, entry_time = c(0, 250), fitness_times = 30,
  var_per_run = 2)
summary(cyp_inv)
```

---

summary.adaptProj *Summarize adaptProj Objects*

Description

Function summary.adaptProj() summarizes adaptProj objects.

Usage

```
## S3 method for class 'adaptProj'
summary(
  object,
  threshold = 1,
  inf_alive = TRUE,
  milepost = c(0, 0.25, 0.5, 0.75, 1),
  ext_time = FALSE,
  ...
)
```

Arguments

object	An adaptProj object.
threshold	A threshold population size to be searched for in projections. Defaults to 1.
inf_alive	A logical value indicating whether to treat infinitely large population size as indicating that the population is still extant. If FALSE, then the population is considered extinct. Defaults to TRUE.
milepost	A numeric vector indicating at which points in the projection to assess detailed results. Can be input as integer values, in which case each number must be between 1 and the total number of occasions projected in each projection, or decimals between 0 and 1, which would then be translated into the corresponding projection steps of the total. Defaults to c(0, 0.25, 0.50, 0.75, 1.00).
ext_time	A logical value indicating whether to output extinction times per population-patch. Defaults to FALSE.
...	Other parameters currently not utilized.

Value

Apart from a statement of the results, this function outputs a list with the following elements:

`milepost_sums` A data frame showing the number of replicates at each of the milepost times that is above the threshold population/patch size.

`extinction_times` A dataframe showing the numbers of replicates going extinct (`ext_reps`) and mean extinction time (`ext_time`) per population-patch. If `ext_time` = `FALSE`, then only outputs `NA`.

Notes

The `inf_alive` and `ext_time` options both assess whether replicates have reached a value of `NaN` or `Inf`. If `inf_alive` = `TRUE` or `ext_time` = `TRUE` and one of these values is found, then the replicate is counted in the `milepost_sums` object if the last numeric value in the replicate is above the `threshold` value, and is counted as extant and not extinct if the last numeric value in the replicate is above the extinction threshold of a single individual.

Extinction time is calculated on the basis of whether the replicate ever falls below a single individual. A replicate with a positive population size below 0.0 that manages to rise above 1.0 individual is still considered to have gone extinct the first time it crossed below 1.0.

If the input `lefkoProj` object is a mixture of two or more other `lefkoProj` objects, then mileposts will be given relative to the maximum number of time steps noted.

Examples

```
library(lefko3)
data(cypdata)

data(cypa_data)

sizevector <- c(0, 0, 0, 0, 0, 0, 1, 2.5, 4.5, 8, 17.5)
stagevector <- c("SD", "P1", "P2", "P3", "SL", "D", "XSm", "Sm", "Md", "Lg",
  "XLg")
repvector <- c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
obsvector <- c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
matvector <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
immvector <- c(0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 0, 0, 0, 0, 0.5, 0.5, 1, 1, 2.5, 7)

cypframe_raw <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  propstatus = propvector, immstatus = immvector, indataset = indataset,
  binhalfwidth = binvec)

cycaraw_v1 <- verticalize3(data = cypdata, noyears = 6, firstyear = 2004,
  patchidcol = "patch", individcol = "plantid", blocksize = 4,
  sizeacol = "Inf2.04", sizebcol = "Inf.04", sizeccol = "Veg.04",
  repstracol = "Inf.04", repstrbcol = "Inf2.04", fecacol = "Pod.04",
```

```
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cyparaw_v1 <- verticalize3(data = cypa_data, noyears = 18, firstyear = 1994,
  individcol = "plant_id", blocksize = 2, sizeacol = "Inf.94",
  sizebcol = "Veg.94", repstracol = "Inf.94", fecacol = "Inf.94",
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cypsupp2r <- supplemental(stage3 = c("SD", "P1", "P2", "P3", "SL", "D",
    "XSm", "Sm", "SD", "P1"),
  stage2 = c("SD", "SD", "P1", "P2", "P3", "SL", "SL", "SL", "rep",
    "rep"),
  eststage3 = c(NA, NA, NA, NA, NA, "D", "XSm", "Sm", NA, NA),
  eststage2 = c(NA, NA, NA, NA, NA, "XSm", "XSm", "XSm", NA, NA),
  givenrate = c(0.10, 0.20, 0.20, 0.20, 0.25, NA, NA, NA, NA, NA),
  multiplier = c(NA, NA, NA, NA, NA, NA, NA, NA, 0.5, 0.5),
  type =c(1, 1, 1, 1, 1, 1, 1, 1, 3, 3),
  stageframe = cypframe_raw, historical = FALSE)
cyp_supp_list1 <- list(cypsupp2r, cypsupp2r)

cycamatrix2r <- rlefko2(data = cycaraw_v1, stageframe = cypframe_raw,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", indivcol = "individ")

cypamatrix2r <- rlefko2(data = cyparaw_v1, stageframe = cypframe_raw,
  year = "all", stages = c("stage3", "stage2", "stage1"),
  size = c("size3added", "size2added"), supplement = cypsupp2r,
  yearcol = "year2", indivcol = "individ")

cyp_mpm_list <- list(cycamatrix2r, cypamatrix2r)

cyca2_start <- start_input(cycamatrix2r, stage2 = c("SD", "P1", "P2"),
  value = c(500, 100, 200))
cypa2_start <- start_input(cypamatrix2r, stage2 = c("SD", "P1", "P2"),
  value = c(5000, 1000, 2000))
cyp_start_list <- list(cyca2_start, cypa2_start)

cyp2_dv <- density_input(cypamatrix2r, stage3 = c("SD", "P1"),
  stage2 = c("rep", "rep"), style = c(1, 1), alpha = c(0.5, 1.2),
  beta = c(1.0, 2.0), type = c(2, 1))
cyp_dv_list <- list(cyp2_dv, cyp2_dv)

cyp_comm_proj <- project3(mpms = cyp_mpm_list, starts = cyp_start_list,
  density = cyp_dv_list, times = 10)

summary(cyp_comm_proj)
```

ta_skeleton *Create Skeleton Data Frame for Trait Variation for Invasion Analysis*

Description

Function `ta_skeleton()` creates a core data frame that can be modified by users to provide the core variation in transition elements and vital rates to use in invasion analysis. The resulting data frame should be used as input in function `invade3()`.

Usage

```
ta_skeleton(rows = 10L)
```

Arguments

`rows`	The number of rows needed in the data frame. Defaults to 10.

Value

A data frame of class `adaptAxis`, with the following columns:

`variant`	Denotes each variant in order, with each row corresponding to a novel variant.
`stage3`	Stage at occasion *t*+1 in the transition to be replaced.
`stage2`	Stage at occasion *t* in the transition to be replaced.
`stage1`	Stage at occasion *t*-1 in the transition to be replaced.
`age3`	Age at occasion *t*+1 in the transition to be replaced.
`age2`	Age at occasion *t* in the transition to be replaced.
`eststage3`	Stage at occasion *t*+1 in the transition to replace the transition designated by `stage3`, `stage2`, and `stage1`.
`eststage2`	Stage at occasion *t* in the transition to replace the transition designated by `stage3`, `stage2`, and `stage1`.
`eststage1`	Stage at occasion *t*-1 in the transition to replace the transition designated by `stage3`, `stage2`, and `stage1`.
`estage3`	Age at occasion *t*+1 in the transition to replace the transition designated by `age2`.
`estage2`	Age at occasion *t* in the transition to replace the transition designated by `age2`.
`givenrate`	A constant to be used as the value of the transition.
`offset`	A constant value to be added to the transition or proxy transition.
`multiplier`	A multiplier for proxy transitions or for fecundity.
`convtype`	Designates whether the transition from occasion *t* to occasion *t*+1 is a survival transition probability (1), a fecundity rate (2), or a fecundity multiplier (3).
`convtype_t12`	Designates whether the transition from occasion *t*-1 to occasion *t* is a survival transition probability (1), a fecundity rate (2).

surv_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for survival probability.
obs_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for observation probability.
size_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for primary size transition.
sizeb_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for secondary size transition.
sizec_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for tertiary size transition.
repst_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for reproduction probability.
fec_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for fecundity.
jsurv_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for juvenile survival probability.
jobs_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for juvenile observation probability.
jsize_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for juvenile primary size transition.
jsizeb_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for juvenile secondary size transition.
jsizec_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for juvenile tertiary size transition.
jrepst_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for juvenile reproduction probability.
jmatst_dev	A numeric vector giving the deviations to the y-intercept of the vital rate model for maturity status.

Examples

```
current_traits <- ta_skeleton(4)
current_traits$stage3 <- c("Dorm", "Dorm", "Sdl1", NA)
current_traits$stage2 <- c("cut", "V0r", "rep", NA)
current_traits$convtype <- c(1, 1, 2, NA)
current_traits$offset <- c(0.1, 0.2, 0.3, NA)
current_traits$surv_dev <- c(NA, NA, NA, 0.1)
```

trait_axis *Create a Data Frame of Trait Data for Invasion Analysis*

Description

Function `trait_axis()` provides all necessary data for invasion analysis. It lists the specific variations to MPMs for each variant run. Variants can be given via overwritten matrix elements, proxy matrix elements, additive offsets on matrix elements, matrix element multipliers, and additive offsets to y-intercepts in vital rate models.

Usage

```
trait_axis(
  historical = NULL,
  stagebased = NULL,
  agebased = NULL,
  stageframe = NULL,
  stage3 = NULL,
  stage2 = NULL,
  stage1 = NULL,
  age3 = NULL,
  age2 = NULL,
  eststage3 = NULL,
  eststage2 = NULL,
  eststage1 = NULL,
  estage3 = NULL,
  estage2 = NULL,
  givenrate = NULL,
  offset = NULL,
  multiplier = NULL,
  type = NULL,
  type_t12 = NULL,
  surv_dev = NULL,
  obs_dev = NULL,
  size_dev = NULL,
  sizeb_dev = NULL,
  sizec_dev = NULL,
  repst_dev = NULL,
  fec_dev = NULL,
  jsurv_dev = NULL,
  jobs_dev = NULL,
  jsize_dev = NULL,
  jsizeb_dev = NULL,
  jsizec_dev = NULL,
  jrepst_dev = NULL,
  jmatst_dev = NULL
)
```

Arguments

historical	A single logical value indicating whether the MPMs intended will be historical or ahistorical. Defaults to TRUE.
stagebased	A single logical value indicating whether the MPM will be stage-based or age-by-stage. Defaults to TRUE.
agebased	A single logical value indicating whether the MPM will be age-based or age-by-stage. Defaults to FALSE.
stageframe	The stageframe used to produce the MPM. Required if producing any stage-based or age-by-stage MPM. Must be omitted for purely age-based MPMs.
stage3	String vector of stage names in occasion *t*+1 in the transition to be affected. Abbreviations for groups of stages are also usable (see Notes). Required in all stage-based and age-by-stage MPMs.
stage2	String vector of stage names in occasion *t* in the transition to be affected. Abbreviations for groups of stages are also usable (see Notes). Required in all stage-based and age-by-stage MPMs.
stage1	String vector of stage names in occasion *t*-1 in the transition to be affected. Only needed if a historical matrix is to be produced. Abbreviations for groups of stages are also usable (see Notes). Required for historical stage-based MPMs.
age3	An integer vector of the ages in occasion *t*+1 to use in transitions to be affected. Required for all age- and age-by-stage MPMs.
age2	An integer vector of the ages in occasion *t* to use in transitions to be affected. Required for all age- and age-by-stage MPMs.
eststage3	String vector of stage names to replace stage3 in a proxy transition. Only needed if a transition will be replaced by another estimated transition, and only in stage-based and age-by-stage MPMs.
eststage2	String vector of stage names to replace stage2 in a proxy transition. Only needed if a transition will be replaced by another estimated transition, and only in stage-based and age-by-stage MPMs.
eststage1	String vector of stage names to replace stage1 in a proxy historical transition. Only needed if a transition will be replaced by another estimated transition, and the matrix to be estimated is historical and stage-based. Stage NotAlive is also possible for raw hMPMs as a means of handling the prior stage for individuals entering the population in occasion *t*.
estage3	Integer vector of age at time *t*+1 to replace age3 in a proxy transition. Only needed if a transition will be replaced by another estimated transition, and only in age-based and age-by-stage MPMs.
estage2	Integer vector of age at time *t* to replace age2 in a proxy transition. Only needed if a transition will be replaced by another estimated transition, and only in age-based and age-by-stage MPMs.
givenrate	A numeric vector of fixed rates or probabilities to replace for the transition described by stage3, stage2, stage1, and/or age2.
offset	A numeric vector of fixed numeric values to add to the transitions described by stage3, stage2, stage1, and/or age2.

`multiplier` A numeric vector of multipliers for the transition described by `stage3`, `stage2`, `stage1`, and/or `age2`, or for the proxy transitions described by `eststage3`, `eststage2`, `eststage1`, and/or `estage2`. Defaults to `1`.

`type` Integer vector denoting the kind of transition between occasions *t* and *t*+1 to be replaced. This should be entered as `1`, `S`, or `s` for the replacement of a survival transition; `2`, `F`, or `f` for the replacement of a fecundity transition; or `3`, `R`, or `r` for a fecundity set value / general multiplier. If empty or not provided, then defaults to `1` for survival transition.

`type_t12` An optional integer vector denoting the kind of transition between occasions *t*-1 and *t*. Only necessary if a historical MPM in deVries format is desired. This should be entered as `1`, `S`, or `s` for a survival transition; or `2`, `F`, or `f` for a fecundity transitions. Defaults to `1` for survival transition, with impacts only on the construction of deVries-format hMPMs.

`surv_dev` An optional vector of numeric deviations to the y-intercept of the survival model used in function-based MPM creation. Defaults to `NA` for all values.

`obs_dev` An optional vector of numeric deviations to the y-intercept of the observation model used in function-based MPM creation. Defaults to `NA` for all values.

`size_dev` An optional vector of numeric deviations to the y-intercept of the primary size model used in function-based MPM creation. Defaults to `NA` for all values.

`sizeb_dev` An optional vector of numeric deviations to the y-intercept of the secondary size model used in function-based MPM creation. Defaults to `NA` for all values.

`sizec_dev` An optional vector of numeric deviations to the y-intercept of the tertiary size model used in function-based MPM creation. Defaults to `NA` for all values.

`repst_dev` An optional vector of numeric deviations to the y-intercept of the reproduction model used in function-based MPM creation. Defaults to `NA` for all values.

`fec_dev` An optional vector of numeric deviations to the y-intercept of the fecundity model used in function-based MPM creation. Defaults to `NA` for all values.

`jsurv_dev` An optional vector of numeric deviations to the y-intercept of the juvenile survival model used in function-based MPM creation. Defaults to `NA` for all values.

`jobs_dev` An optional vector of numeric deviations to the y-intercept of the juvenile observation model used in function-based MPM creation. Defaults to `NA` for all values.

`jsize_dev` An optional vector of numeric deviations to the y-intercept of the juvenile primary size model used in function-based MPM creation. Defaults to `NA` for all values.

`jsizeb_dev` An optional vector of numeric deviations to the y-intercept of the juvenile secondary size model used in function-based MPM creation. Defaults to `NA` for all values.

`jsizec_dev` An optional vector of numeric deviations to the y-intercept of the juvenile tertiary size model used in function-based MPM creation. Defaults to `NA` for all values.

`jrepst_dev` An optional vector of numeric deviations to the y-intercept of the juvenile reproduction model used in function-based MPM creation. Defaults to `NA` for all values.

`jmatst_dev` An optional vector of numeric deviations to the y-intercept of the juvenile maturity model used in function-based MPM creation. Defaults to `NA` for all values.

Value

A data frame of class `adaptAxis`. This object can be used as input in function `invade3()`.

Variables in this object include the following:

`variant`	Denotes each variant in order, with each row corresponding to a novel variant.
`stage3`	Stage at occasion *t*+1 in the transition to be replaced.
`stage2`	Stage at occasion *t* in the transition to be replaced.
`stage1`	Stage at occasion *t*-1 in the transition to be replaced.
`age3`	Age at occasion *t*+1 in the transition to be replaced.
`age2`	Age at occasion *t* in the transition to be replaced.
`eststage3`	Stage at occasion *t*+1 in the transition to replace the transition designated by `stage3`, `stage2`, and `stage1`.
`eststage2`	Stage at occasion *t* in the transition to replace the transition designated by `stage3`, `stage2`, and `stage1`.
`eststage1`	Stage at occasion *t*-1 in the transition to replace the transition designated by `stage3`, `stage2`, and `stage1`.
`estage3`	Age at occasion *t*+1 in the transition to replace the transition designated by `age3`.
`estage2`	Age at occasion *t* in the transition to replace the transition designated by `age2`.
`givenrate`	A constant to be used as the value of the transition.
`offset`	A constant value to be added to the transition or proxy transition.
`multiplier`	A multiplier for proxy transitions or for fecundity.
`convtype`	Designates whether the transition from occasion *t* to occasion *t*+1 is a survival transition probability (1), a fecundity rate (2), or a fecundity multiplier (3).
`convtype_t12`	Designates whether the transition from occasion *t*-1 to occasion *t* is a survival transition probability (1), a fecundity rate (2).
`surv_dev`	Numeric deviations to the y-intercept of the vital rate model of survival.
`obs_dev`	Numeric deviations to the y-intercept of the vital rate model of observation.
`size_dev`	Numeric deviations to the y-intercept of the vital rate model of primary size.
`sizeb_dev`	Numeric deviations to the y-intercept of the vital rate model of secondary size.
`sizec_dev`	Numeric deviations to the y-intercept of the vital rate model of tertiary size.
`repst_dev`	Numeric deviations to the y-intercept of the vital rate model of reproduction.
`fec_dev`	Numeric deviations to the y-intercept of the vital rate model of fecundity.
`jsurv_dev`	Numeric deviations to the y-intercept of the vital rate model of juvenile survival.
`jobs_dev`	Numeric deviations to the y-intercept of the vital rate model of juvenile observation.
`jsize_dev`	Numeric deviations to the y-intercept of the vital rate model of juvenile primary size.
`jsizeb_dev`	Numeric deviations to the y-intercept of the vital rate model of juvenile secondary size.

jsizec_dev	Numeric deviations to the y-intercept of the vital rate model of juvenile tertiary size.
jrepst_dev	Numeric deviations to the y-intercept of the vital rate model of juvenile reproduction.
jmatst_dev	Numeric deviations to the y-intercept of the vital rate model of juvenile maturity.

Notes

Negative values are not allowed in `givenrate` and `multiplier` input, but are allowed in `offset`, if values are to be subtracted from specific estimated transitions. Stage entries should not be used for purely age-based MPMs, and age entries should not be used for purely stage-based MPMs.

Entries in `stage3`, `stage2`, and `stage1` can include abbreviations for groups of stages. Use `rep` if all reproductive stages are to be used, `nrep` if all mature but non-reproductive stages are to be used, `mat` if all mature stages are to be used, `immat` if all immature stages are to be used, `prop` if all propagule stages are to be used, `npr` if all non-propagule stages are to be used, `obs` if all observable stages are to be used, `nobs` if all unobservable stages are to be used, and leave empty or use `all` if all stages in stageframe are to be used. Also use `groupX` to denote all stages in group X (e.g. `group1` will use all stages in the respective stageframe's group 1).

Type 3 conversions are referred to as fecundity set values, or general fecundity multipliers. These set the transitions to be used as fecundity transitions. Transitions set here will be interpreted as being generally reproductive, meaning that the from and to stages will be used to determine the general fecundity transitions to incorporate into stage-based MPMs, while the age portion of the input will be used to incorporate the actual multiplier(s) specified. If only stage transitions at certain ages are expected to be the sole contributors to fecundity, then type 2 conversions should also be included in the supplement (Type 1 and 2 conversions can be purely age-specific, and do not set reproductive transitions in MPM creation). For example, if all stage 2 to stage 3 transitions above age 2 yield fecundity, then stage 2 to stage 3 can be set to `multiplier = 1.0` with `convtype = 3`, and the same transition for `age2 = c(1, 2)` can be set to `multiplier = c(0, 0)`.

Several operations may be included per transition. Operations on the same row of the resulting data frame are generally handled with given rate substitutions first, then with proxy transitions, then by additive offsets, and finally by multipliers. This order can be manipulated by ordering operations across rows, with higher numbered rows in the data frame being performed later.

See Also

`ta_skeleton()`

Examples

```
library(lefko3)

data(cypa_data)

sizevector <- c(0, 0, 0, 0, 0, 0, 1, 2.5, 4.5, 8, 17.5)
stagevector <- c("SD", "P1", "P2", "P3", "SL", "D", "XSm", "Sm", "Md", "Lg",
  "XLg")
repvector <- c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
obsvector <- c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
```

```
matvector <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
immvector <- c(0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0)
propvector <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
indataset <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)
binvec <- c(0, 0, 0, 0, 0, 0.5, 0.5, 1, 1, 2.5, 7)

cypframe_raw <- sf_create(sizes = sizevector, stagenames = stagevector,
  repstatus = repvector, obsstatus = obsvector, matstatus = matvector,
  propstatus = propvector, immstatus = immvector, indataset = indataset,
  binhalfwidth = binvec)

cypraw_v1 <- verticalize3(data = cypa_data, noyears = 18, firstyear = 1994,
  individcol = "plant_id", blocksize = 2, sizeacol = "Inf.94",
  sizebcol = "Veg.94", repstracol = "Inf.94", fecacol = "Inf.94",
  stageassign = cypframe_raw, stagesize = "sizeadded", NAas0 = TRUE,
  NRasRep = TRUE)

cypa_ta <- trait_axis(stageframe = cypframe_raw,
  stage3 = c("P1", "P1", "P1", NA, NA, NA),
  stage2 = c("rep", "rep", "rep", NA, NA, NA),
  multiplier = c(0.5, 2.0, 10., NA, NA, NA), type = c(2, 2, 2, NA, NA, NA),
  obs_dev = c(NA, NA, NA, 0.5, 2.0, 50), fec_dev = c(NA, NA, NA, -1000, 0, 1000))
```

Index